



GRUPO#3

Integrantes:

- Karem Huacota
- Karen Torrico
- Yesika Luna
- Ivan Mamani
- Elvis Miranda

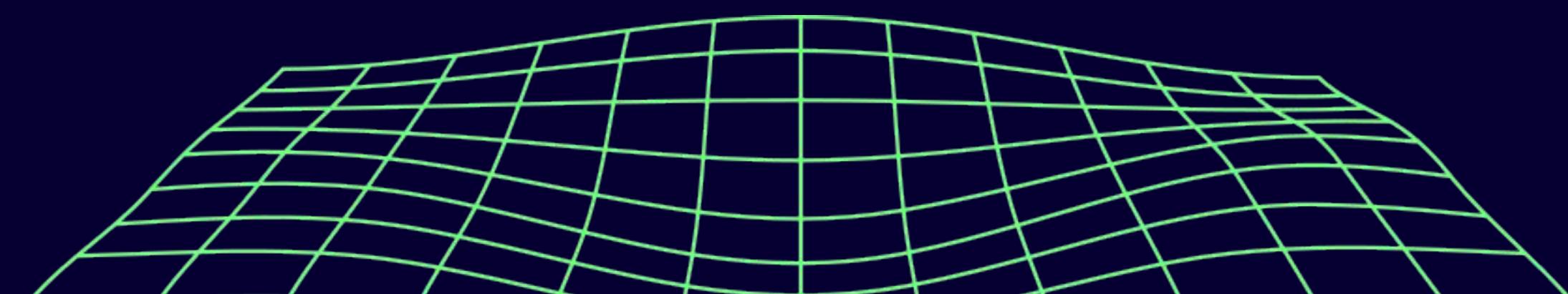
MACHINE LEARNING AND DEEP LEARNING

Julio 18-07, 2025

Proyecto de Clasificación Supervisada

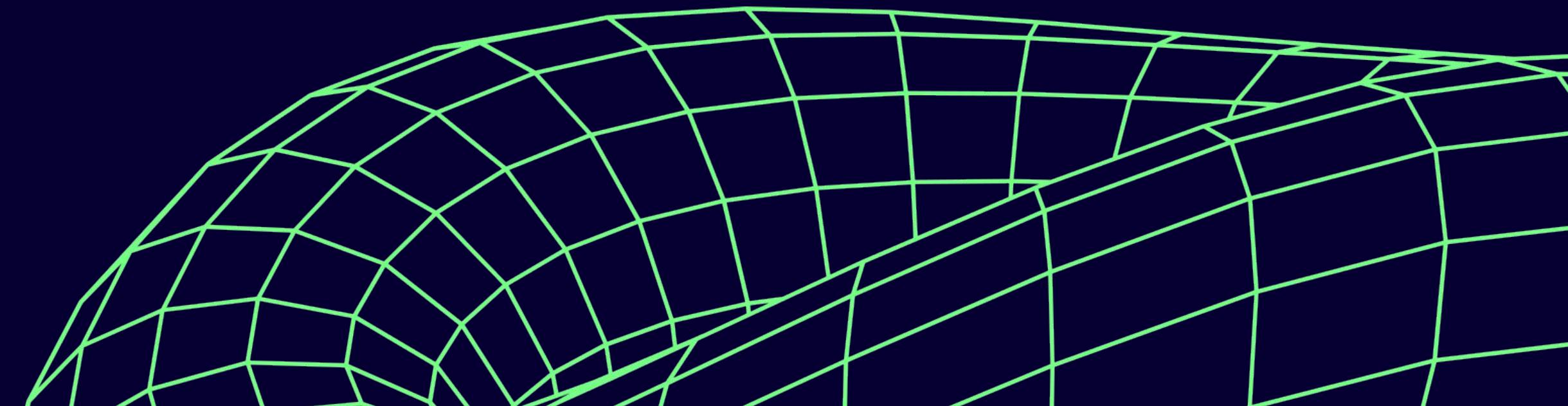
Introducción

En esta presentación, cubriremos los aspectos más relevantes del tratamiento de datos en proyectos de Machine Learning y Deep Learning. Analizaremos los modelos utilizados, la configuración de hiperparámetros y las métricas aplicadas, así como los desafíos enfrentados y las lecciones aprendidas.



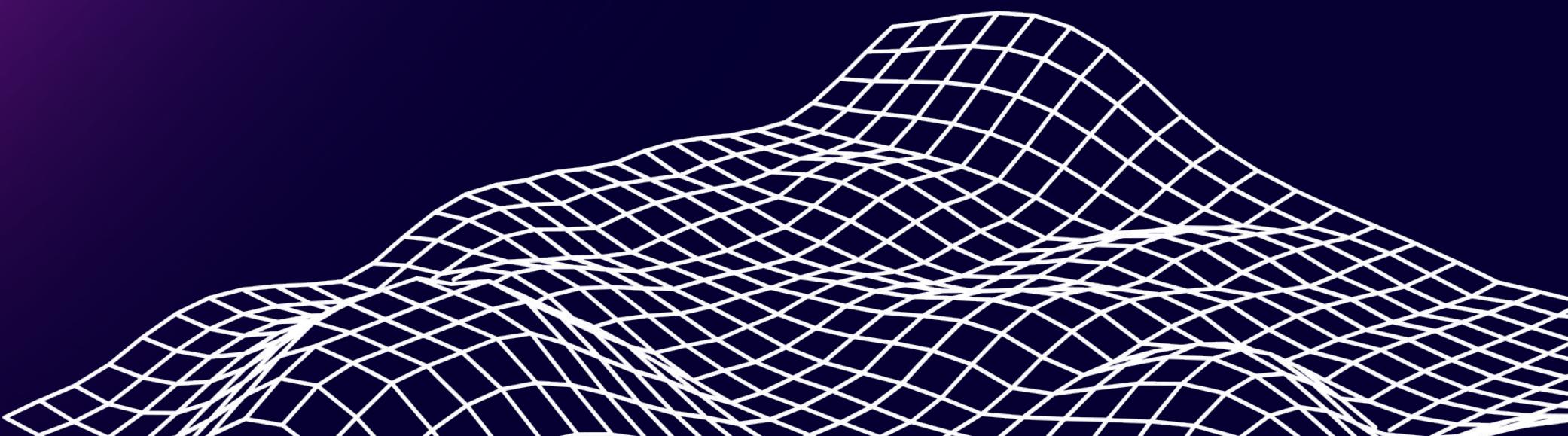
Contenido

- 1 Descripción del problema**
- 2 Tratamiento de Datos**
- 3 Modelos y Configuración**
- 4 Retos y Soluciones**
- 5 Conclusión**



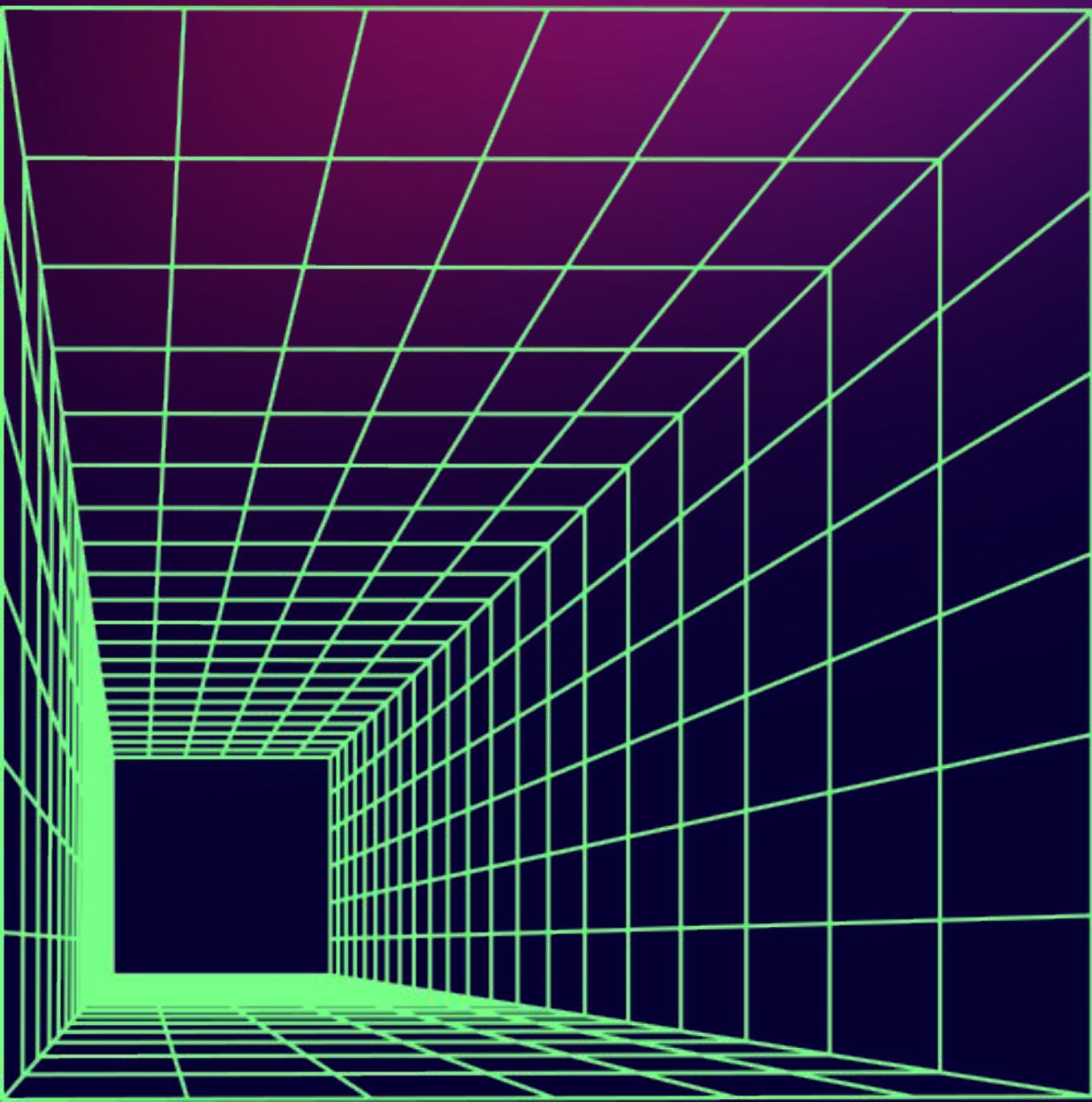
Descripción del problema

1



DESCRIPCION

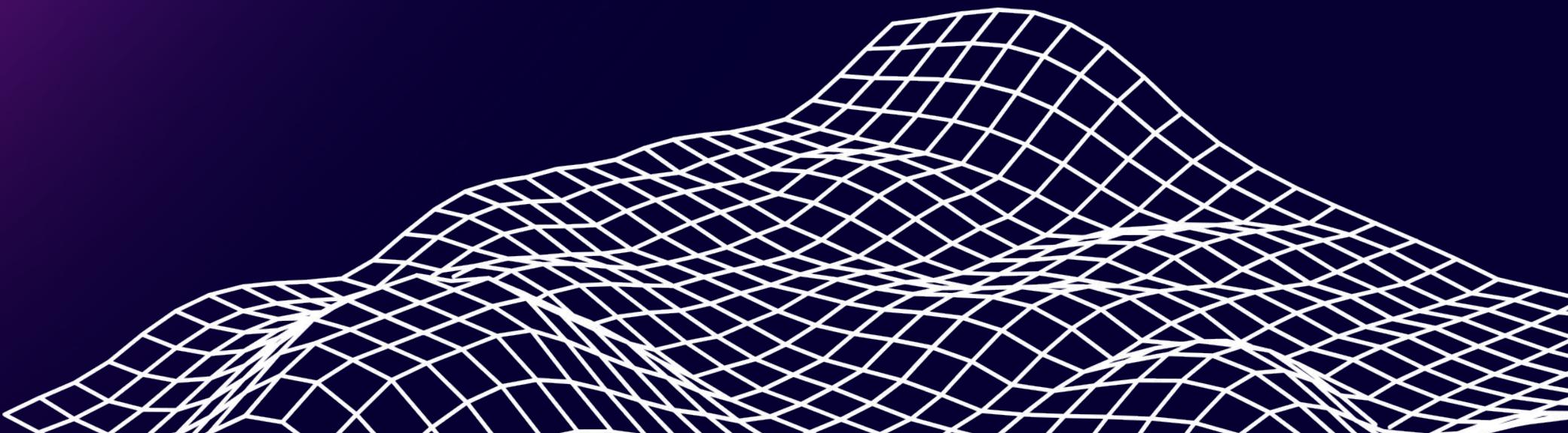
El Proyecto aborda un problema de clasificación supervisada utilizando un conjunto de datos cuyo objetivo es predecir la variable **OBJ**, de naturaleza binaria.



- Variable objetivo : La variable OBJ adopta valores **0 o 1**, indicando la clase o categoría correspondiente a cada observación.
- Variable Independiente
 - ✓ Variables Absolutas:
V1, V2, V3, ..., V30
Corresponden a valores numéricos en escala absoluta,
 - ✓ Variables Relativas:
V1_P, V2_P, V3_P, ..., V30_P
Representan los valores absolutos normalizados respecto al total de las 30 variables originales,

TRATAMIENTO DE DATOS

2



Análisis Exploratorio

- ✓ El dataset contiene **10,000 filas** y 66 columnas, incluyendo identificadores (ALEAT, ID, VT), la variable objetivo (OBJ), y características (V1 a V30 y sus versiones normalizadas V1_P a V30_P).

```
# Inspeccionar todas las columnas
print(df.columns)

→ Index(['ALEAT', 'ID', 'OBJ', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8',
       'V9', 'V10', 'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18',
       'V19', 'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28',
       'V29', 'V30', 'V31_P', 'VT', 'V1_P', 'V2_P', 'V3_P', 'V4_P', 'V5_P',
       'V6_P', 'V7_P', 'V8_P', 'V9_P', 'V10_P', 'V11_P', 'V12_P', 'V13_P',
       'V14_P', 'V15_P', 'V16_P', 'V17_P', 'V18_P', 'V19_P', 'V20_P', 'V21_P',
       'V22_P', 'V23_P', 'V24_P', 'V25_P', 'V26_P', 'V27_P', 'V28_P', 'V29_P',
       'V30_P', 'V31_P.1'],
      dtype='object')

[ ] # Observando el nombre de las columnas podemos observar que:
# ALEAT, ID -> Muy probablemente identificadores únicos.
# V31_P, V31_P.1 -> No forman parte del bloque V1-V30; sospechosas de ser duplicadas.
# VT -> Sospecha de identificador o columna técnica auxiliar.

# Entonces comprobamos si son identificadores (Si el número de valores únicos es igual al número de filas, mas probable que sea identificador)
for col in ['ALEAT', 'ID', 'VT']:
    if col in df.columns:
        print(col, "→ Valores únicos:", df[col].nunique(), "/", len(df))

→ ALEAT → Valores únicos: 9999 / 10000
ID → Valores únicos: 10000 / 10000
VT → Valores únicos: 9994 / 10000

[ ] # Eliminar columnas irrelevantes es para evitar que el modelo use datos que NO aportan valor predictivo
# (ejemplo: IDs, nombres aleatorios). Podrían introducir ruido o incluso fugas de información.
df = df.drop(columns=['ALEAT', 'ID', 'VT'])
```

LIMPIEZA DE DATOS

1. Eliminación de columnas irrelevantes o identificadores

Se eliminaron las columnas ALEAT, ID, y VT por ser identificadores únicos o casi únicos (valores únicos cercanos al número de filas), lo que evita ruido y fugas de información en el modelo.

2. Eliminación de columnas residuales con alta cardinalidad y valores residuales numéricos

- Las columnas V31_P y V31_P.1 se eliminaron debido a su alta cardinalidad, predominancia de ceros, y valores residuales que sugieren ruido técnico sin valor predictivo.
- Las columnas V14 y V14_P se eliminaron por tener valor

3. Verificación y confirmación de ausencia de valores nulos:

Se confirmó que no había valores nulos en las columnas restantes, eliminando la necesidad de imputación.

4. Selección de Variables:

Se optó por usar las variables originales (V1 a V30, excluyendo V14) en lugar de las normalizadas (V1_P a V30_P) para el modelado,

CODIFICACION

Codificación: Variable objetivo '**OBJ**' codificada como binaria (Sí=1, NO=0).

Balanceo: Aplicado **SMOTE** para corregir desbalance de clases.

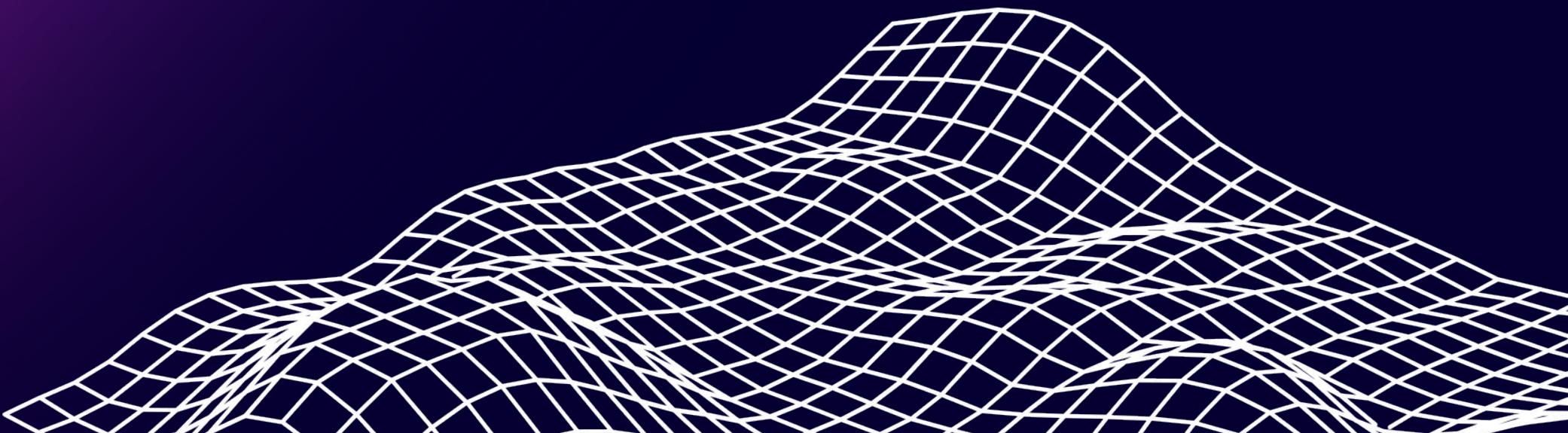
Variables: Usadas variables originales ('V1' a 'V30', sin 'V14').

Justificación:

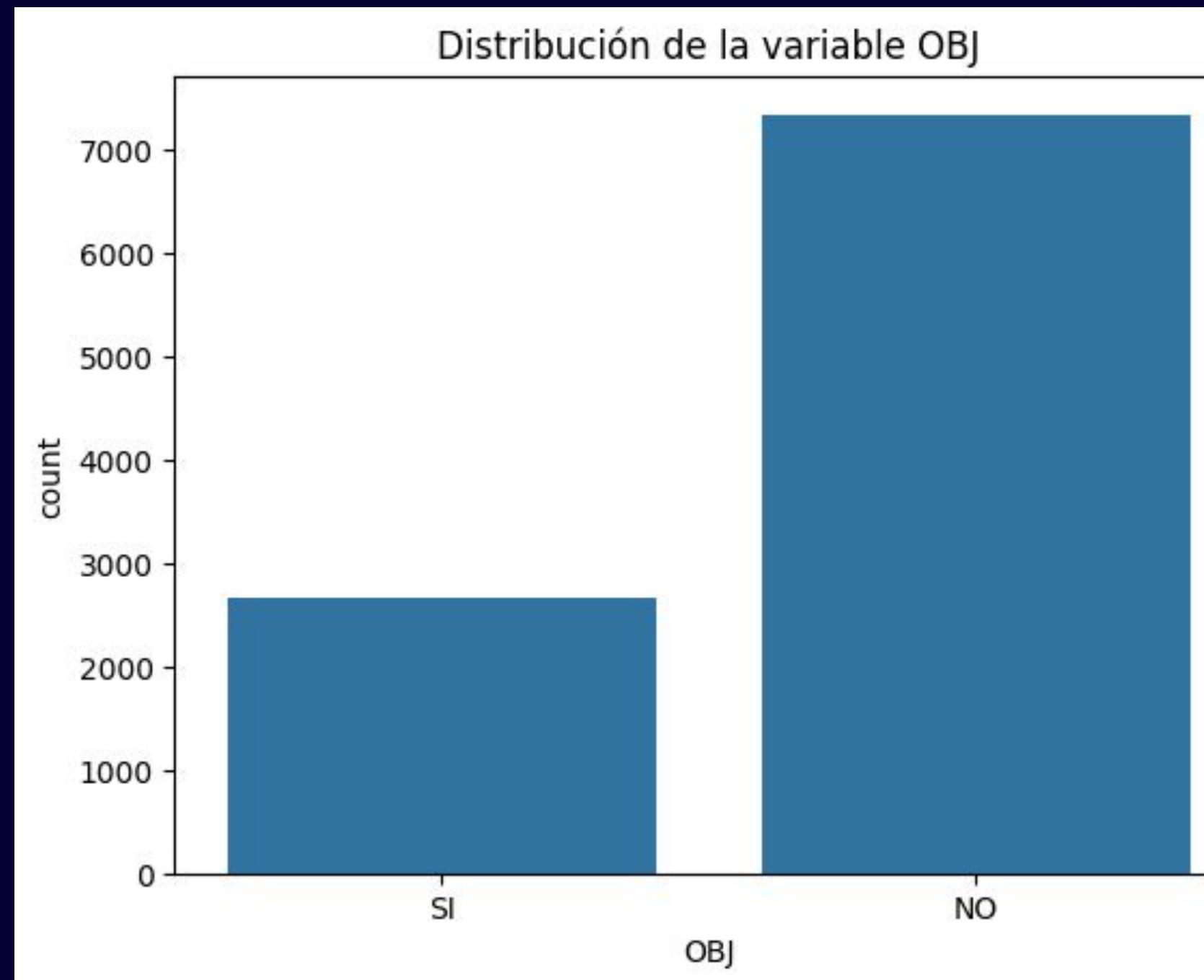
Estas transformaciones aseguran que los datos sean relevantes, consistentes y libres de ruido, reduciendo el riesgo de sobreajuste y mejorando la calidad del entrenamiento

3

Análisis preliminar de los datos.



Distribución de la variable objetivo OBJ

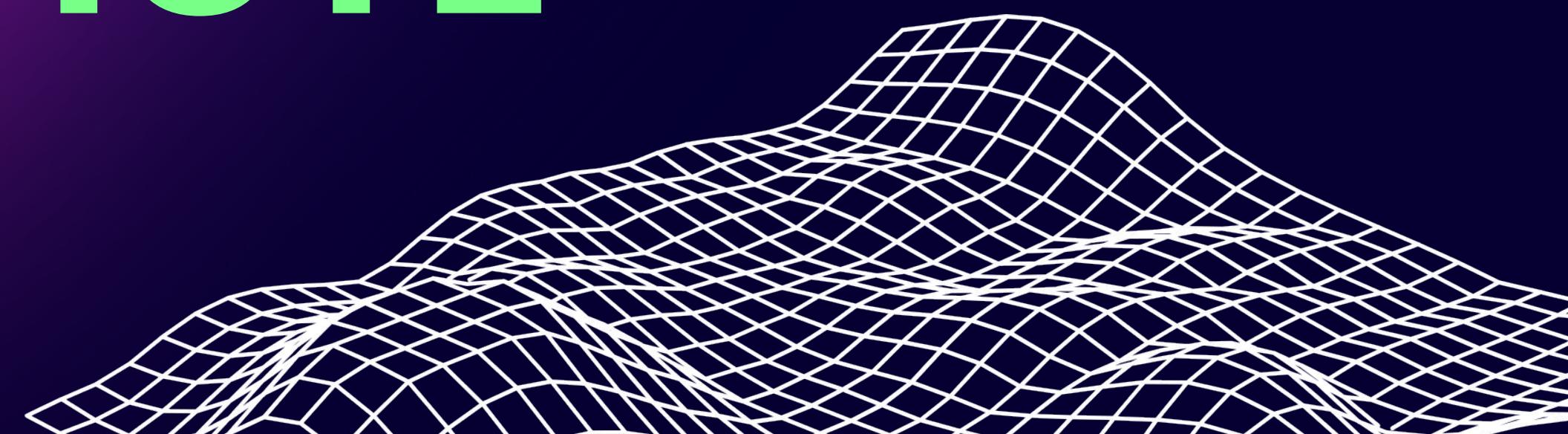


Se utilizó `sns.countplot` para visualizar la cantidad de registros pertenecientes a cada clase de la variable OBJ.

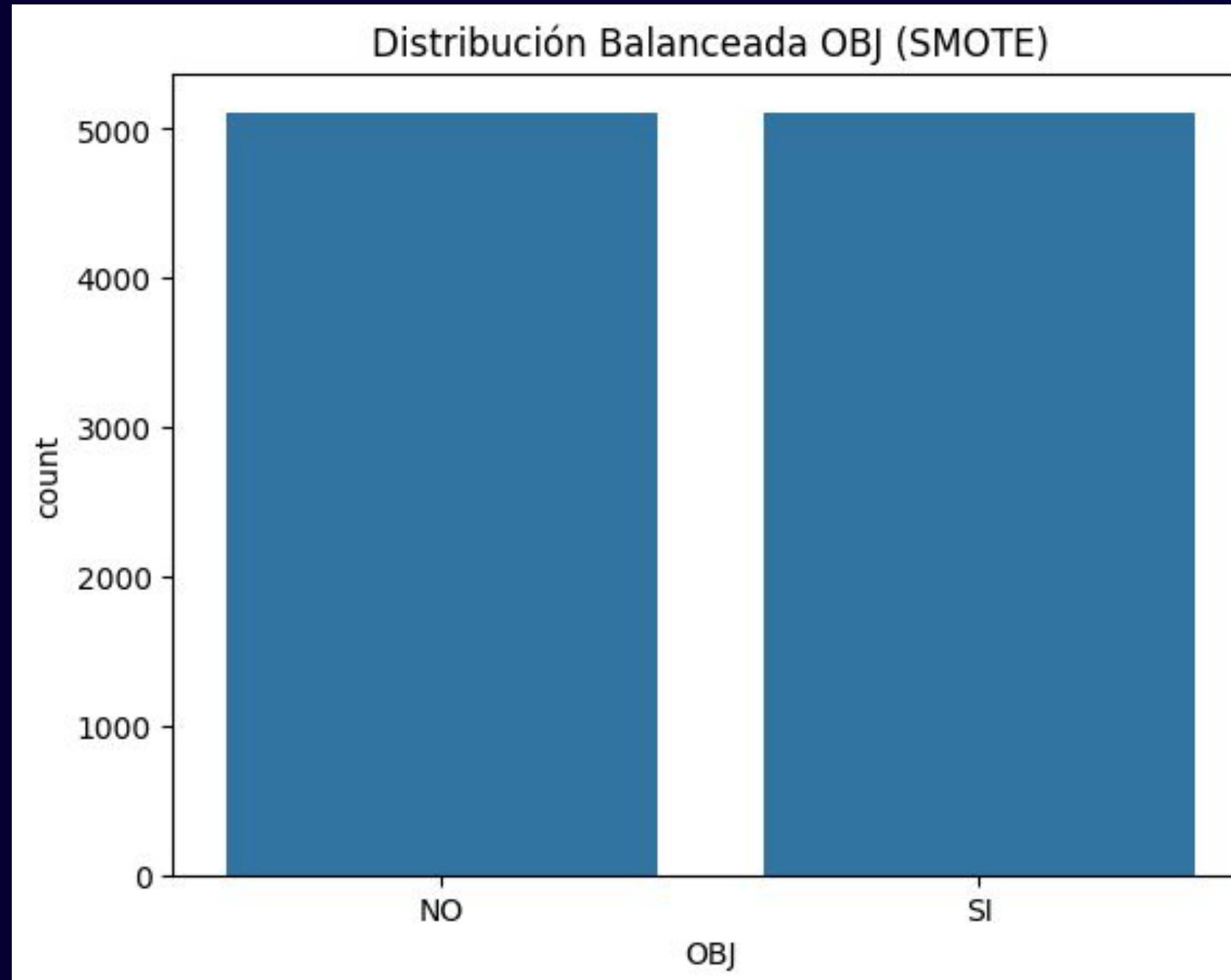
El dataset presenta un claro desbalance de clases, lo cual puede afectar el rendimiento de los modelos de clasificación. Será necesario considerar técnicas como el ajuste de pesos o el uso de balanceo de clases como SMOTE.

Division de Datos y Balance de Clases con SMOTE

4



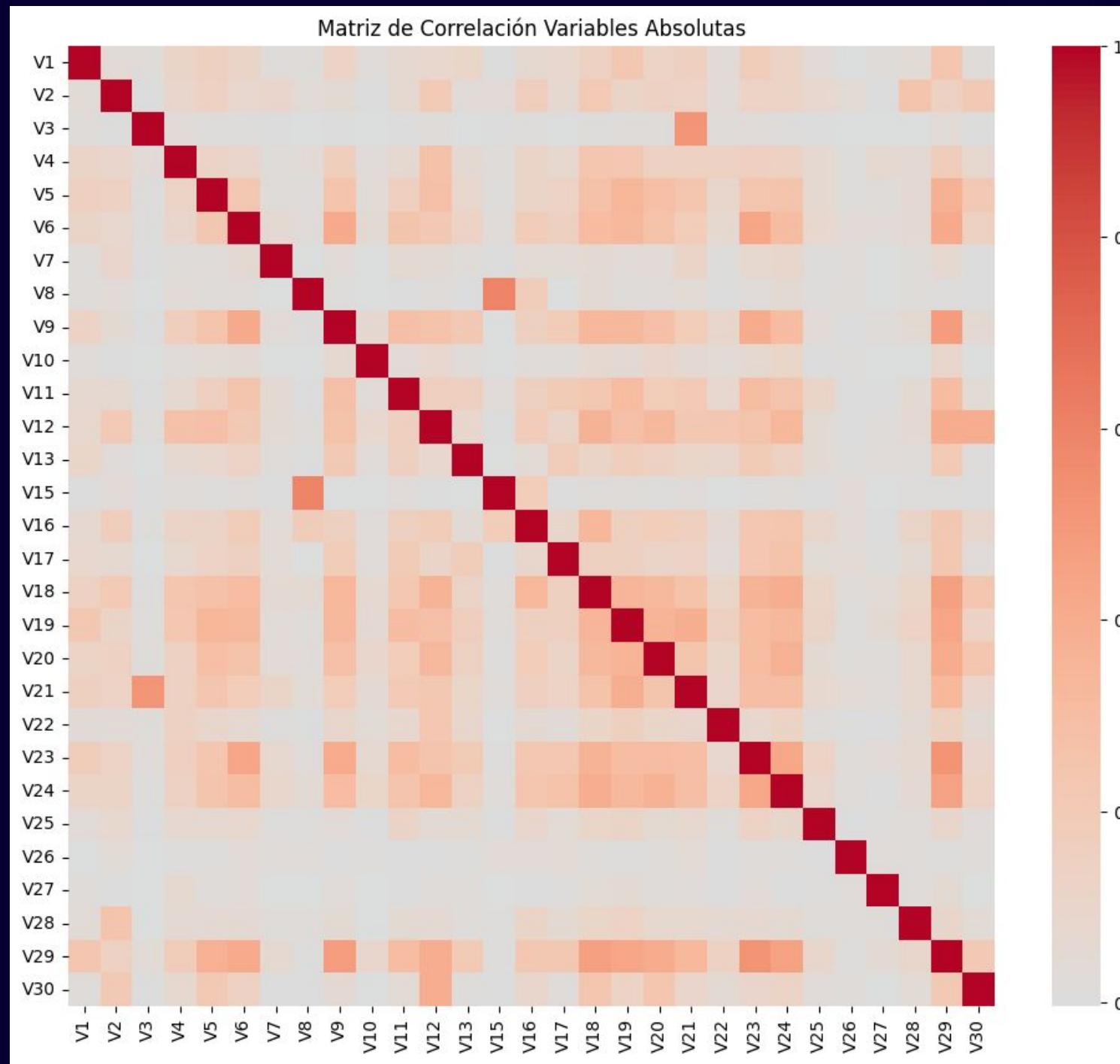
Motivo del balanceo - SMOTE



SMOTE genera nuevas muestras sintéticas de la clase minoritaria, creando puntos intermedios entre registros existentes, sin simplemente duplicar datos.

Después de aplicar SMOTE, la distribución de clases quedó perfectamente balanceada:
-5000 registros SI
-5000 registros NO

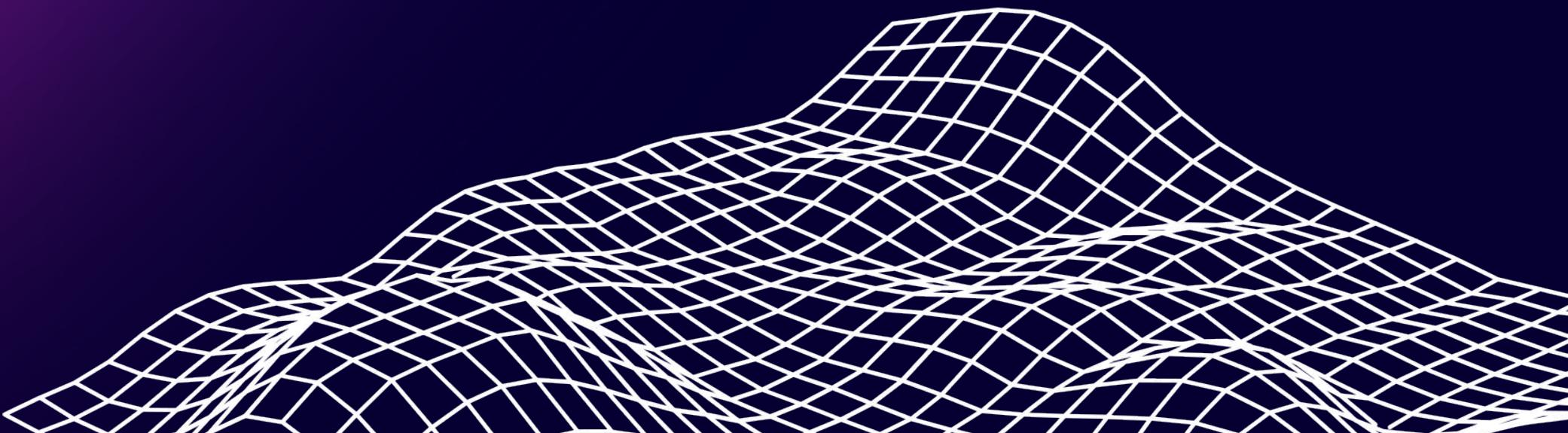
Matriz de Correlación entre Variables Absolutas (V1 a V30)



No se detectan problemas graves de multicolinealidad.
Las variables relativas no son redundantes entre sí, por lo que todas podrían aportar información al modelo.

5

MODELOS Y CONFIGURACION



MODELOS EVALUADOS

Se evaluaron cuatro modelos:

REGRESION LINEAL

Un modelo lineal para clasificación binaria, adecuado para relaciones lineales entre características y la variable objetivo.

RANDOM FOREST

Un modelo de ensamble basado en árboles de decisión, robusto frente a datos ruidosos y capaz de capturar relaciones no lineales.

SVN

✓ Un modelo que maximiza el margen entre clases, útil para datasets con separaciones claras

XGBoost

Un modelo de ensamble basado en gradient boosting, optimizado para alta precisión y manejo de dataset complejos

Configuración hiperparámetros

```
modelos = [
    # 'LogisticRegression': {
    #     'model': LogisticRegression(max_iter=5000, class_weight='balanced'),
    #     'params': {'C': [0.01, 0.1, 1, 10]}
    # },
    'RandomForest': {
        # 'model': RandomForestClassifier(random_state=42, class_weight='balanced', n_estimators=1000, max_depth=None),
        'model': RandomForestClassifier(random_state=42, n_estimators=1000, max_depth=None, min_samples_split=2, min_samples_leaf=1, max_features=10),
        # 'params': {'n_estimators': [100, 150], 'max_depth': [5, 10, None]}
        'params': {'n_estimators': [100, 150], 'max_depth': [5, 10, None]}
    },
    # 'SVM_RBF': {
    #     'model': SVC(kernel='rbf', probability=True, class_weight='balanced'),
    #     'params': {'C': [0.1, 1, 10], 'gamma': ['scale', 0.01, 0.001]}
    # },
    # 'XGBoost': {
    #     'model': XGBClassifier(use_label_encoder=False, eval_metric='logloss'),
    #     'params': {'n_estimators': [100, 150], 'max_depth': [3, 5, 7], 'scale_pos_weight': [1, 2, 5]}
    # }
]
```

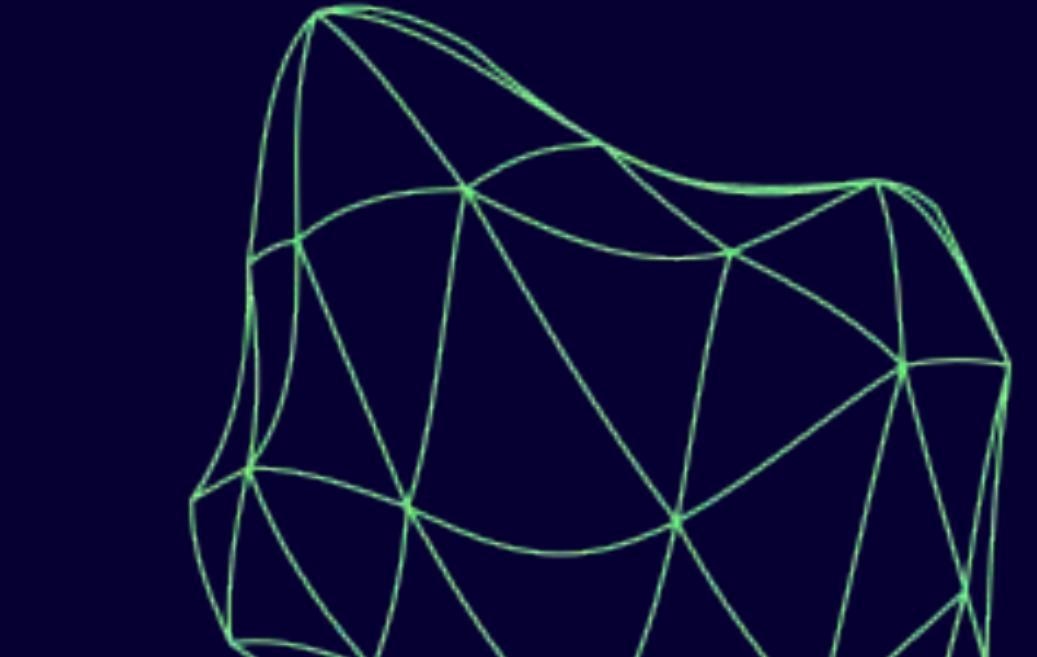
resultados = []

SVN : El kernel RBF es versátil para capturar relaciones no lineales, y un valor estándar de C es un punto de partida razonable.

XGBoost:

Regresión Logística: La regularización L2 ayuda a prevenir el sobreajuste, y un valor moderado de C balancea la complejidad del modelo.

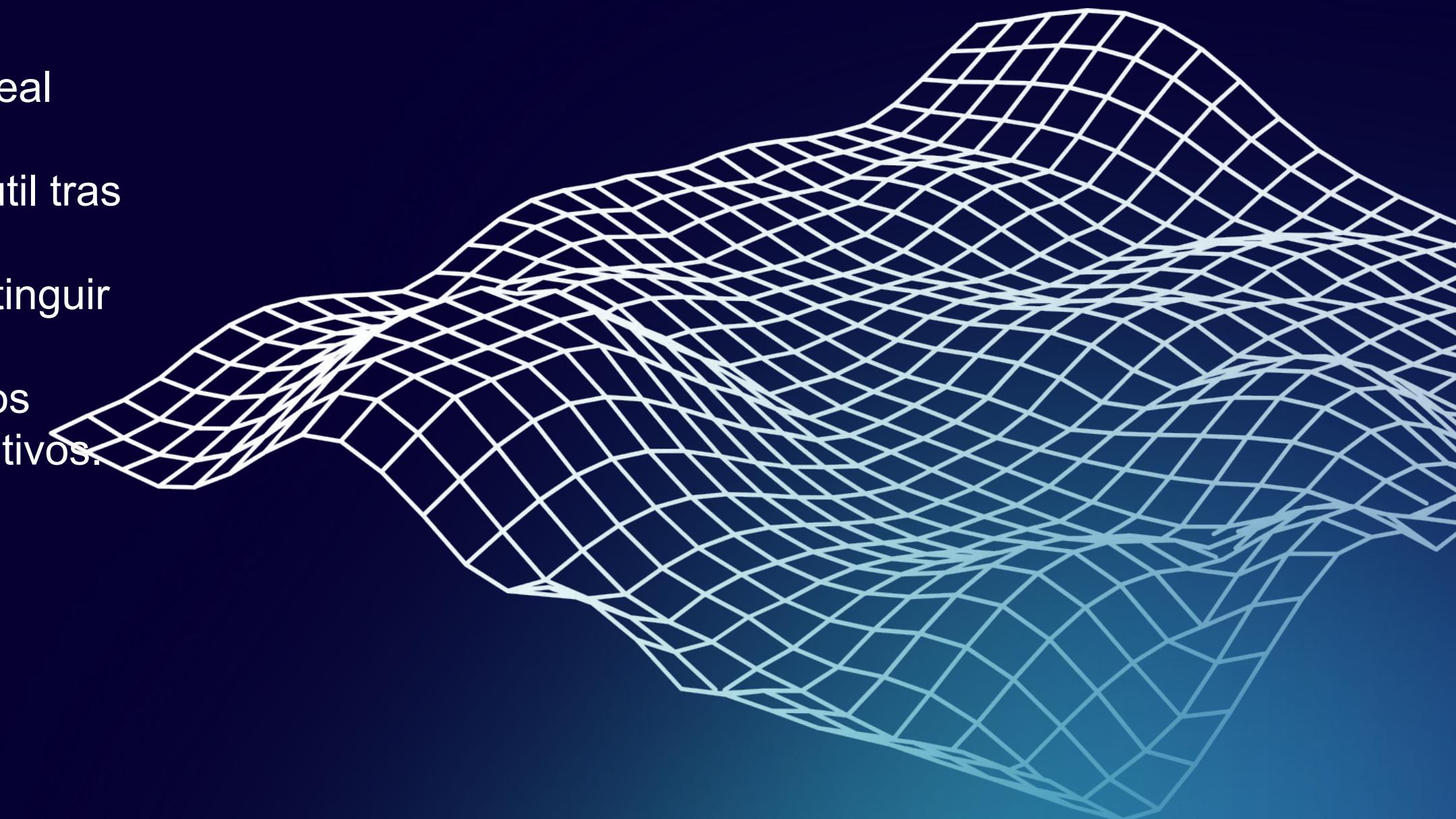
Random Forest: Un número moderado de árboles (n_estimators) y una profundidad limitada (max_depth) reducen el sobreajuste mientras mantienen un buen rendimiento.



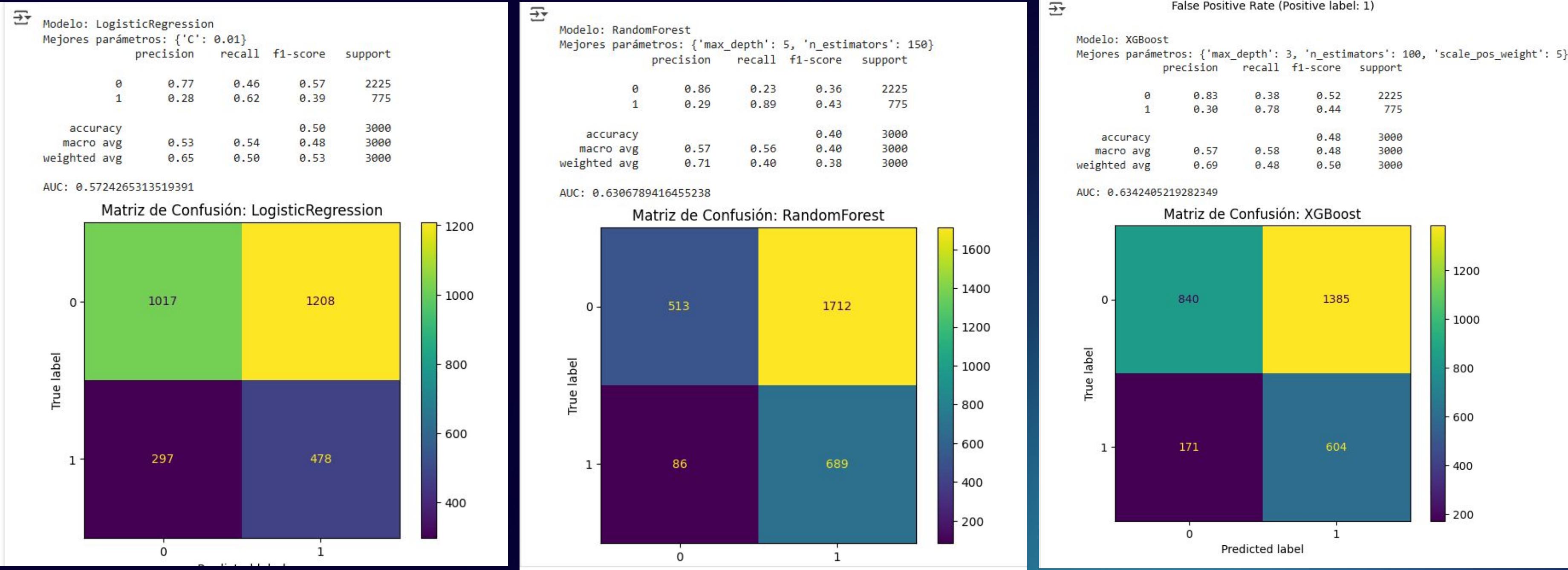
Métricas Utilizadas

Las métricas evaluadas son:

1. **F1-Score**: Media armónica de precisión y recall, ideal para datasets desbalanceados.
2. **Accuracy**: Proporción de predicciones correctas, útil tras aplicar SMOTE.
3. **AUC-ROC**: Mide la capacidad del modelo para distinguir entre clases.
4. **Matriz de confusión**: Evalúa falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos.



Comparación Modelos



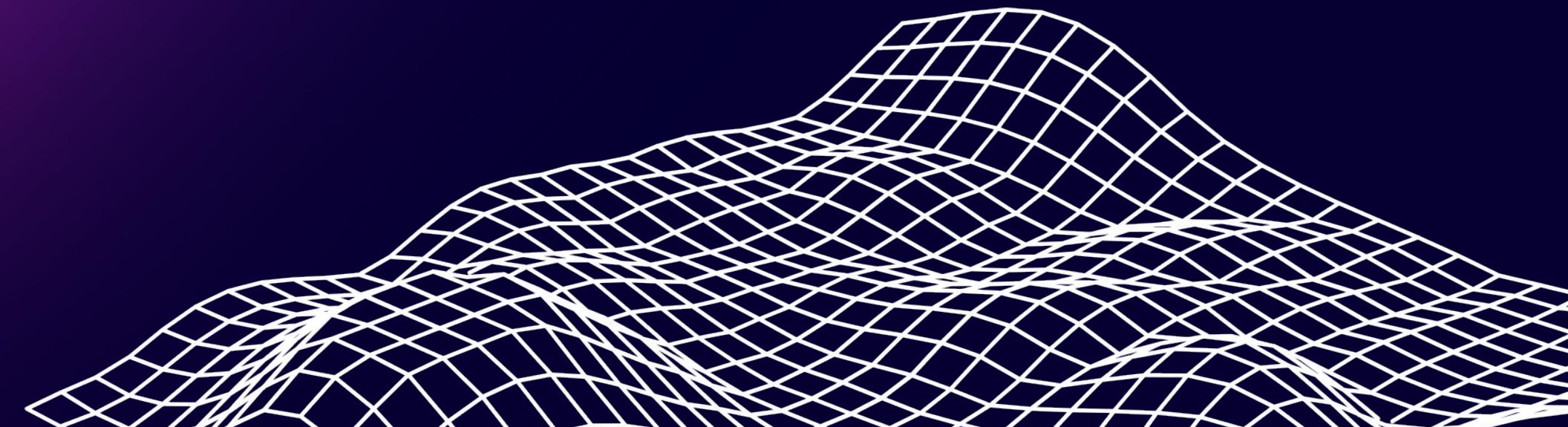
Colocar texto

Colocar texto

Colocar texto

6

Demo





SCHOOL OF
ENGINEERING
UNIDAD DE POSTGRADO
UAGRM - FICCT

GRUPO 3

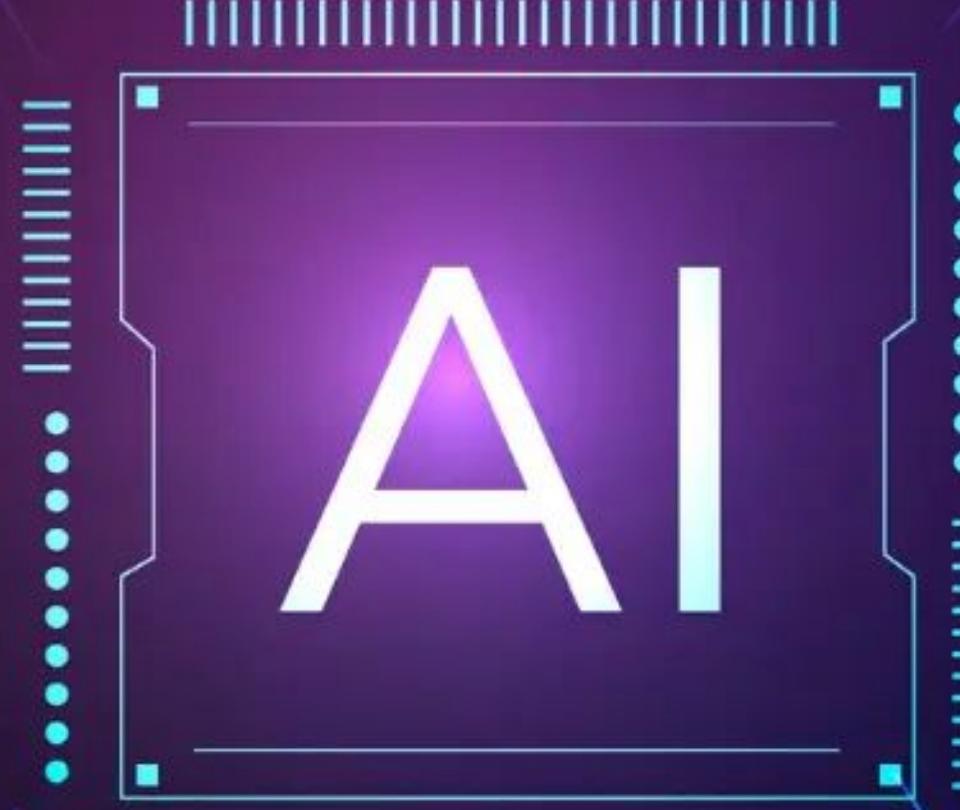
Proyecto de Aprendizaje Supervisado - Clasificación Binaria



Notebook

Link del notebook

Retos encontrados



AI

A graphic of a central processing unit (CPU) or artificial intelligence chip. It features a dark purple background with white circuit lines forming a grid pattern. In the center, the letters "AI" are written in a large, white, sans-serif font. The chip has a rectangular outline with small blue squares at the corners and vertical lines extending from the sides.

- Problemas de sobreajuste.
- Dificultades en la limpieza de datos.
- Desbalanceo de clases.



SCHOOL OF
ENGINEERING
UNIDAD DE POSTGRADO
UAGRM - FICCT

GRUPO 3

Proyecto de Aprendizaje Supervisado - Clasificación Binaria



GRACIAS!