



Universidad de  
**SanAndrés**

**I302 - Aprendizaje Automático  
y Aprendizaje Profundo**

**Trabajo Práctico 2:  
Clasificación y Ensemble Learning**

Matteo Musacchio

15 de abril de 2025

Ingeniería en Inteligencia Artificial

# 1. Diagnóstico de Cáncer de Mama

## Resumen

En esta primera parte del trabajo se desarrolló un modelo de regresión logística binaria para predecir el diagnóstico de cáncer de mama a partir de diversas características celulares. Se trabajó con dos versiones del dataset: una balanceada y otra desbalanceada. En ambos casos, se aplicó un esquema de preprocessamiento que incluyó la imputación de valores faltantes mediante KNN, el tratamiento de valores atípicos como NaNs, codificación de variables categóricas y normalización por Min-Max scaling.

En el dataset balanceado, se realizó un ajuste del hiperparámetro de regularización  $\lambda$  mediante un barrido logarítmico y se evaluó el modelo final sobre el conjunto de test, obteniendo un F1-score de 0.882, un AUC-ROC de 0.939 y una precisión del 89.7 %.

En el caso del dataset desbalanceado, se implementaron y compararon cinco técnicas de rebalanceo: sin rebalanceo, undersampling, oversampling por duplicación, SMOTE y cost re-weighting. SMOTE resultó ser la técnica más efectiva, logrando el mejor compromiso entre precisión, recall y capacidad de discriminación, con un AUC-ROC de 0.909 y F1-score de 0.836.

## 1.1. Introducción

El diagnóstico temprano de tumores es una herramienta clave en la medicina, especialmente en el contexto del cáncer de mama, donde una detección precisa puede mejorar significativamente el pronóstico de los pacientes. En este trabajo se aborda un problema de clasificación binaria cuyo objetivo es predecir si un tumor es benigno o maligno a partir de características celulares.

El conjunto de datos utilizado fue generado a partir de imágenes histopatológicas de biopsias mamarias, de las cuales se extrajeron variables morfológicas y moleculares. Estas variables incluyen el tamaño y la forma de las células, la densidad nuclear, la tasa de mitosis, la saturación de oxígeno y la presencia de mutaciones genéticas.

La entrada del algoritmo está compuesta por estas características numéricas y categóricas, y la salida es una predicción binaria: 0 si el tumor es benigno y 1 si es maligno. Para modelar esta relación se utilizó un algoritmo de regresión logística binaria con regularización L2. El modelo fue entrenado sobre un conjunto balanceado y posteriormente evaluado en un conjunto desbalanceado, aplicando diversas estrategias de rebalanceo como undersampling, oversampling, SMOTE y reponderación de la función de costo.

## 1.2. Métodos

Se abordó el problema de clasificación binaria utilizando un conjunto de datos con características morfológicas y moleculares de células. Las variables fueron normalizadas con min-max scaling y usando el conjunto que ya estaba dividido en 90 % dev y

10 % test, se dividió el dev en 80 % entrenamiento, 20 % validación.

### 1.2.1. Modelos implementados

- Regresión logística binaria con regularización L2

El entrenamiento se realizó minimizando la función de pérdida logarítmica regularizada con penalización L2:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \left[ y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \|\theta\|^2$$

Este modelo estima la probabilidad de que una observación pertenezca a la clase positiva mediante la función sigmoidal:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

El vector de parámetros  $\theta$  se actualizó utilizando descenso por gradiente.

### 1.2.2. Preprocesamiento de los datos

Antes de entrenar los modelos, se aplica a cada split un pipeline de preprocessamiento a los datos con el objetivo de mejorar la calidad de la información y la performance de los clasificadores. Las etapas fueron las siguientes:

**Detección de outliers:** se identificaron valores atípicos en las variables numéricas utilizando los cuantiles 25 ( $Q_1$ ) y 75 ( $Q_3$ ). Un valor se consideró

outlier si estaba fuera del rango intercuartílico extendido:

$$x < Q_1 - 1,5 \cdot IQR \quad \text{o} \quad x > Q_3 + 1,5 \cdot IQR$$

donde  $IQR = Q_3 - Q_1$ . Los valores detectados como outliers fueron reemplazados por valores nulos (`NaN`) para que pudieran ser tratados en la siguiente etapa.

**Imputación de valores faltantes:** los valores nulos, incluyendo los generados en la etapa anterior, fueron imputados utilizando el algoritmo

**K-Nearest Neighbors Imputer (KNN).** Para cada muestra con un valor faltante, se buscaron los  $k$  vecinos más cercanos (en este caso, en el espacio de características completas) y se imputó el valor faltante con el promedio de los valores correspondientes en dichos vecinos.

$$x_i^{(j)} = \frac{1}{k} \sum_{n \in \mathcal{N}_k(i)} x_n^{(j)}$$

donde  $x_i^{(j)}$  es la característica faltante en la muestra  $i$ , y  $\mathcal{N}_k(i)$  es el conjunto de los  $k$  vecinos más cercanos.

**Codificación de variables categóricas:** se aplicó *One-Hot Encoding* a las columnas categóricas, convirtiendo cada categoría en una columna binaria.

**Normalización:** finalmente, se aplicó una normalización Min-Max a cada característica para llevar sus valores al rango  $[0, 1]$ . Dado un valor  $x$ , se normaliza como:

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Esto se realizó utilizando estadísticas calculadas a partir del conjunto de entrenamiento (mínimo y máximo por columna), y luego se aplicaron de forma consistente a los conjuntos de validación o prueba.

### 1.2.3. Rebalanceo de clases

Dado el desbalance entre las clases (mayoría de ejemplos benignos), se implementaron diversas técnicas de rebalanceo sobre el conjunto de entrenamiento para mejorar la capacidad del modelo de detectar correctamente la clase minoritaria (maligna).

A continuación se describen las estrategias aplicadas:

**Undersampling:** se eliminan muestras de la clase mayoritaria para igualar la cantidad de la minoritaria. Esto puede llevar a la pérdida de información valiosa, pero ayuda a equilibrar el conjunto de datos.

**Oversampling por duplicación:** se duplican muestras de la clase minoritaria hasta igualar el tamaño de la clase mayoritaria. Esto puede llevar a un sobreajuste, ya que el modelo puede aprender patrones específicos de las muestras duplicadas.

**SMOTE (Synthetic Minority Over-sampling Technique):** se generan nuevos ejemplos sintéticos interpolando entre una muestra de la clase minoritaria y uno de sus  $k$  vecinos más cercanos. Dado un punto  $x$  y su vecino  $x_{\text{nn}}$ , se crea un nuevo punto:

$$x_{\text{new}} = x + \delta \cdot (x_{\text{nn}} - x), \quad \delta \sim \mathcal{U}(0, 1)$$

Esto permite crear puntos intermedios realistas que ayudan a densificar la región de la clase minoritaria.

**Cost re-weighting:** en lugar de modificar la cantidad de muestras de cada clase, esta estrategia ajusta la función de pérdida para penalizar de forma distinta los errores de clasificación. En particular, se multiplica la contribución de las muestras de la clase minoritaria por un factor de peso definido como:

$$C = \frac{\pi_2}{\pi_1}$$

donde  $\pi_1$  es la probabilidad a priori de la clase minoritaria y  $\pi_2$  la de la clase mayoritaria. La función de pérdida ponderada queda entonces como:

$$\begin{aligned} J(\theta) = & -\frac{1}{m} \sum_{i=1}^m \alpha^{(i)} (y^{(i)} \log(h_{\theta}(x^{(i)})) \\ & + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))) \end{aligned}$$

donde  $\alpha^{(i)} = C$  si  $y^{(i)} = 1$  (clase minoritaria), y  $\alpha^{(i)} = 1$  si  $y^{(i)} = 0$ . De este modo, se compensa el desbalance sin modificar los datos, incentivando al modelo a prestar más atención a la clase minoritaria.

#### 1.2.4. Selección de hiperparámetros

Para seleccionar el valor óptimo del hiperparámetro de regularización  $\lambda$  en el modelo de regresión logística binaria, se utilizó la técnica de *cross-validation* y se la comparó con un barrido clásico sobre un conjunto de validación fijo.

La validación cruzada consiste en dividir el conjunto de entrenamiento en  $k$  *folds* de igual tamaño. Luego, se realizan  $k$  iteraciones. En cada iteración de cross validation se siguió el siguiente procedimiento, se partió del conjunto crudo de entrenamiento, sin aplicar transformaciones previas, y se lo dividió en  $k = 5$  folds. Para cada iteración, se utilizaron  $k - 1$  folds como entrenamiento y uno como validación. El preprocesamiento se aplicó únicamente sobre los folds de entrenamiento, y luego se aplicó al fold de validación utilizando los mismos parámetros.

Este procedimiento se repitió para cada valor de  $\lambda$  en un rango logarítmico entre  $10^{-4}$  y  $10^2$ . En cada caso, se entrenó el modelo y se calculó el F1-score sobre el fold de validación. Finalmente, se promedió el F1-score de cada modelo entre los 5 folds y se seleccionó el valor de  $\lambda$  que obtuvo el mejor desempeño promedio:

$$\lambda^* = \arg \max_{\lambda} \left( \frac{1}{k} \sum_{i=1}^k F1_{\lambda}^{(i)} \right)$$

Luego, para validar la robustez del método, también se realizó un barrido de valores de  $\lambda$  utilizando un conjunto fijo de validación. Se entrenaron modelos con distintos valores de regularización sobre el conjunto de entrenamiento preprocesado, y se evaluaron directamente sobre el conjunto de validación.

Se compararon los valores óptimos obtenidos por ambos enfoques (cross-validation vs. validación fija) en términos de F1-score. Finalmente, se eligió el mejor de los dos para utilizarlo en el resto de las evaluaciones posteriores del trabajo.

#### 1.2.5. Métricas de evaluación

Para evaluar el rendimiento de los modelos clasificadores se utilizaron diversas métricas que permiten analizar tanto el desempeño global como el comportamiento en cada clase. En todos los casos se trató de un problema de clasificación binaria (tumor benigno o maligno), por lo que las métricas se definieron sobre la matriz de confusión:

	Positivo	Negativo
Positivo	TP	FN
Negativo	FP	TN

Donde las **columnas** representan las predicciones del modelo, mientras que las **filas** corresponden a los valores reales.

**Accuracy:** proporción de predicciones correctas sobre el total. Mide el rendimiento global, pero puede ser engañosa en casos de clases desbalanceadas.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

**Precision:** mide qué fracción de los casos predichos como positivos fueron efectivamente positivos.

$$\text{Precision} = \frac{TP}{TP + FP}$$

**Recall:** indica qué fracción de los casos positivos reales fueron correctamente identificados por el modelo.

$$\text{Recall} = \frac{TP}{TP + FN}$$

**F1-score:** es la media armónica entre precisión y recall. Se utiliza cuando se quiere un balance entre ambas métricas, especialmente útil en presencia de clases desbalanceadas.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

**AUC-ROC:** la curva ROC representa la relación entre la Tasa de Verdaderos Positivos (TPR) y la Tasa de Falsos Positivos (FPR):

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}$$

El AUC-ROC mide el área bajo esta curva, y representa la probabilidad de que el modelo asigne una mayor probabilidad a un positivo que a un negativo elegido al azar. Un valor de 0.5 indica un clasificador aleatorio, mientras que 1.0 indica clasificación perfecta.

**AUC-PR:** alternativa al AUC-ROC que resulta más informativa cuando hay fuerte desbalance de clases. Representa el área bajo la curva que grafica *Precision* vs *Recall* para distintos umbrales de decisión.

## 1.3. Resultados

### 1.3.1. Análisis exploratorio

Durante el análisis exploratorio se decidió usar el dataset balanceado, el cual consta con 55 % de instancias con diagnóstico 0 y un 45 % con diagnóstico 1, observamos varios aspectos que motivaron decisiones clave de preprocesamiento. En particular, identificamos la presencia de valores atípicos, valores faltantes y distribuciones altamente sesgadas en muchas variables.

En primer lugar, decidimos tratar los outliers como valores faltantes. Esta decisión se basó en que aproximadamente el 5 % de los datos por columna presentaban valores que se alejaban considerablemente del resto de la distribución, lo cual podía distorsionar las imputaciones y los modelos posteriores si se los mantenía sin tratamiento.

Luego, al evaluar los valores faltantes (incluyendo los que surgieron al tratar los outliers como NaNs), encontramos que cerca del 18 % de los datos por columna estaban ausentes. Dada esta magnitud, consideramos que métodos de imputación simples como la media o la mediana no serían suficientemente robustos, por lo que optamos por utilizar *KNN Imputer*, que permite aprovechar relaciones entre variables para realizar imputaciones más informadas.

Además, aplicamos *one-hot encoding* a las variables categóricas con el fin de prepararlas para los algoritmos de machine learning, que generalmente requieren entradas numéricas. Finalmente, normalizamos las variables numéricas utilizando Min-Max scaling, ya que los modelos empleados (especialmente aquellos sensibles a las escalas como regresión logística o KNN) se benefician de trabajar con datos acotados en un mismo rango.

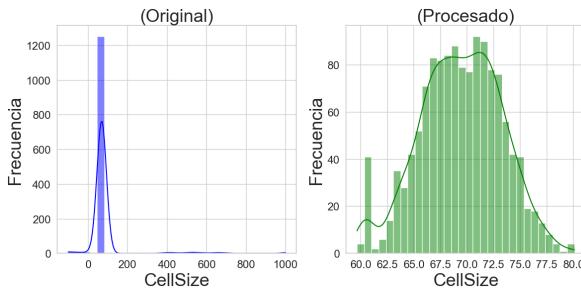


Figura 1: Distribución de la variable `CellSize` antes y después del preprocesamiento.

En la Figura 1 se observa claramente cómo estos

pasos de preprocesamiento tuvieron el efecto deseado: la distribución original de la variable `CellSize` presenta una fuerte concentración en un rango muy acotado y una cola larga de outliers, mientras que la distribución resultante luego del procesamiento es mucho más simétrica y apropiada para el entrenamiento de un modelo de regresión. Cabe destacar que esta es solo una feature que se seleccionó para tener de ejemplo, en la sección A.1 se hace un análisis del dataset completo procesado.

Adicionalmente, se analizó la matriz de correlación entre variables numéricas, la cual se incluye en el Apéndice (Sección A.2)

### 1.3.2. Selección de hiperparámetros

Si bien el procedimiento de validación cruzada es una estrategia robusta para la selección de hiperparámetros, en este caso que el mejor valor de  $\lambda$  encontrado mediante el barrido simple fue  $\lambda = 0.579$  con un F1-score de 0.890, superando al obtenido por validación cruzada  $\lambda = 0.287$ , con F1-score de 0.7685. Cabe destacar que estos resultados se obtuvieron evaluando directamente sobre el conjunto de test, luego de entrenar los modelos con el conjunto de desarrollo del dataset balanceado. Dado que el objetivo principal es maximizar el F1-score y que el valor hallado por barrido ofrece un mejor desempeño, decidimos utilizar solamente el resultado del barrido simple para el resto de las evaluaciones.

### 1.3.3. Evaluación del modelo en el dataset balanceado

Una vez finalizado el proceso de selección de hiperparámetros y entrenamiento del modelo de regresión logística, se evaluó su rendimiento sobre el conjunto de test balanceado provisto. Para ello, se aplicó el mismo preprocesamiento utilizado sobre los datos de desarrollo y se entrenó un modelo final con el mejor valor de regularización  $\lambda$  obtenido mediante barrido de hiperparámetros ( $\lambda = 0.579$ ).

Métrica	Train/Val	Dev/Test
<b>Accuracy</b>	0.930	0.902
<b>Precision</b>	0.894	0.901
<b>Recall</b>	0.96	0.879
<b>F1-Score</b>	0.926	0.890
<b>AUC-ROC</b>	0.949	0.927
<b>AUC-PR</b>	0.887	0.902

Cuadro 1: Métricas de evaluación del modelo en el set de test.

	Positivo	Negativo
Positivo	TP = 94	FN = 8
Negativo	FP = 10	TN = 73

Cuadro 2: Matriz de confusión del modelo en el set de test.

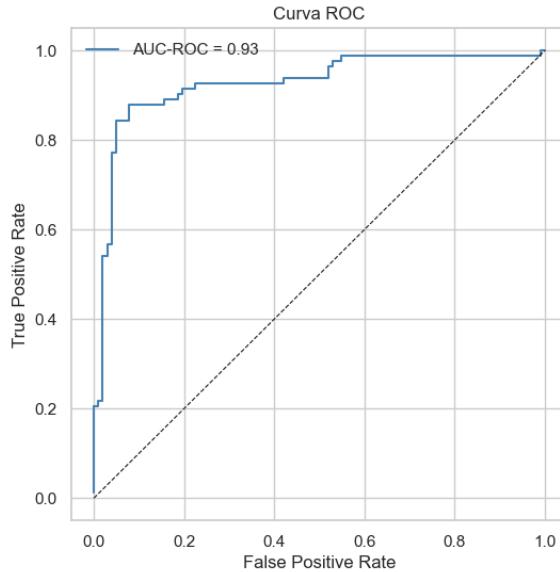


Figura 2: Curva ROC del modelo en el dataset de test.

En la Tabla 1 se presentan las métricas de evaluación del modelo sobre el conjunto de test balanceado. Se observa un buen desempeño en todas las métricas, con un F1-score de 0.882 y un AUC-ROC de 0.939, lo que indica una buena capacidad de discriminación entre las clases. Además, al comparar con las métricas obtenidas en el conjunto de entrenamiento/validación, se observa una performance muy similar, lo cual sugiere que el modelo generaliza correctamente y no está sobreajustado.

La matriz de confusión en la Tabla 2 muestra que el modelo logró clasificar correctamente 71 de los 78 casos positivos y 95 de los 107 casos negativos, con un total de 12 falsos positivos y 7 falsos negativos.

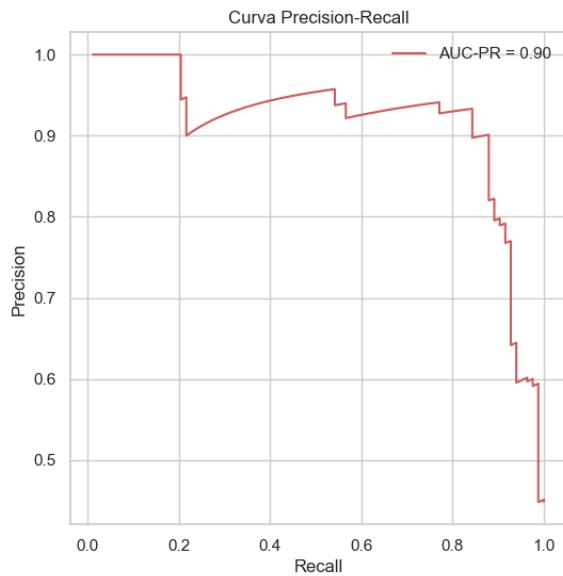


Figura 3: Curva PR del modelo en el dataset de test.

Finalmente, en las Figuras 2 y 3 se muestran las curvas ROC y Precision-Recall, que complementan cuantitativa y visualmente el análisis del modelo. Ambas curvas confirman su buen rendimiento, con un área bajo la curva alta y una clara separación entre las clases predichas.

#### 1.3.4. Evaluación del modelo en el dataset desbalanceado

El dataset desbalanceado presenta una distribución de clases desigual, con aproximadamente un 75 % de diagnósticos clase 0 y un 25 % de clase 1. Además, contiene alrededor de un 20 % de valores faltantes y cerca de un 5 % de outliers por columna, lo cual resulta muy similar a las características observadas en el dataset balanceado. Por este moti-

vo, se aplicó el mismo esquema de preprocesamiento previamente definido. Una vez ajustado el modelo de regresión logística y seleccionado el mejor valor de  $\lambda$ , se evaluó su rendimiento sobre el conjunto de test desbalanceado. Para esta evaluación, se compararon cinco estrategias distintas de manejo del desbalance de clases: sin rebalanceo, undersampling, oversampling por duplicación, SMOTE y cost re-weighting. En todos los casos, se aplicó el pre-

procesamiento ya mencionado y, además, el proceso de selección del mejor valor de  $\lambda$  mediante barrido se repitió luego de aplicar cada técnica de rebalanceo,

asegurando así que el modelo esté optimizado específicamente para los datos generados por cada estrategia.

Modelo	Accuracy	Precision	Recall	F1-Score	AUC-ROC	AUC-PR
Sin Rebalanceo	0.897	0.833	0.735	0.781	0.906	0.772
Undersampling	0.912	0.824	0.824	0.824	0.915	0.804
Duplicación	0.912	0.824	0.824	0.824	0.911	0.795
SMOTE	0.919	0.848	0.824	0.836	0.909	0.782
Cost Re-weighting	0.912	0.824	0.824	0.824	0.914	0.794

Cuadro 3: Métricas de evaluación del modelo sobre el dataset desbalanceado.

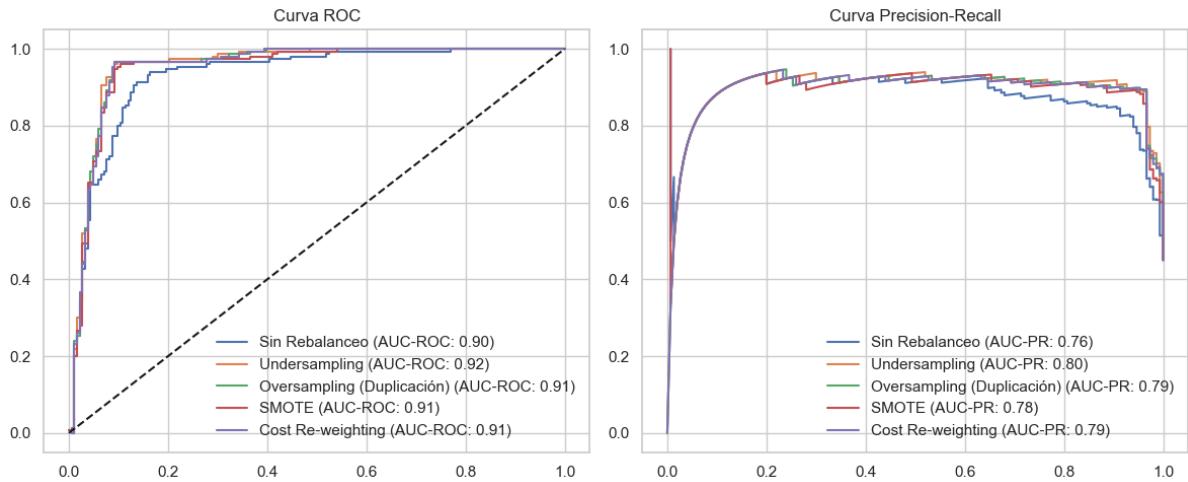


Figura 4: Curvas ROC y PR del modelo sobre el dataset desbalanceado.

En la Tabla 3 se observan los valores obtenidos para cada métrica. Si bien el modelo sin rebalanceo obtuvo un AUC-ROC competitivo (0.906), su F1-score fue muy bajo (0.781), mientras que las estrategias de rebalanceo mejoraron tanto la F1-score como el AUC-PR, métricas más sensibles al desbalance.

Las Figuras en 4 muestran el impacto visual de las técnicas de rebalanceo. Si bien en la curva ROC las diferencias son sutiles, en la curva Precision-Recall se observa una mejora clara, especialmente en la región de alto recall.

A la hora de seleccionar un modelo para un entorno de producción, es importante considerar el contexto y los objetivos específicos del problema. En este caso particular, se optaría por la estrategia de Oversampling mediante SMOTE, ya que fue la que obtuvo las mejores métricas en términos de F1-score y AUC-PR, que son especialmente relevantes frente al desbalance de clases. Además, SMOTE presenta una ventaja conceptual respecto al oversampling por duplicación: en lugar de replicar instancias existentes, genera nuevas muestras sintéticas interpolando entre los vecinos más cercanos de la clase minoritaria, lo que permite una mejor generalización del modelo al evitar el sobreajuste.

## 2. Predicción de Rendimiento de Jugadores de Basketball

### Resumen

En la segunda parte del trabajo se abordó un problema de clasificación multiclasa para predecir el impacto de jugadores de basketball a partir de métricas individuales, utilizando como variable objetivo la clase `WAR class`. Se implementaron desde cero tres algoritmos: regresión logística softmax con regularización L2, análisis discriminante lineal (LDA) y Random Forest con entropía como criterio de división.

El dataset presentaba una distribución de clases relativamente balanceada y no contenía valores faltantes ni variables categóricas. El único preprocessamiento necesario fue la eliminación de algunas entradas inválidas y la normalización de los datos numéricos mediante z-score.

El mejor desempeño lo obtuvo el Random Forest, con un accuracy del 96.1% y un F1-score de 0.961 sobre el conjunto de test. Cabe destacar que todos los modelos presentaron un rendimiento competitivo, lo que evidencia la capacidad de estas arquitecturas para capturar relaciones significativas entre las variables.

### 2.1. Introducción

En esta segunda parte del trabajo práctico se abordó un problema de clasificación multiclasa con el objetivo de predecir el impacto de jugadores de basketball profesionales a partir de métricas individuales. Puntualmente, se buscó estimar la categoría de rendimiento de cada jugador en función de su valor de WAR (Wins Above Replacement), una métrica ampliamente utilizada en deportes para cuantificar el aporte de un jugador en términos de victorias sobre el rendimiento esperado de un suplente promedio.

Los archivos provistos tienen los datos de distintos jugadores de basketball recopilados a lo largo de varias temporadas, donde cada observación representa un jugador en una temporada particular. Las variables de entrada incluyen estadísticas de juego como cantidad de minutos, rebotes, asistencias, eficiencia ofensiva y defensiva, entre otras. La variable objetivo `WAR class` toma valores en 1, 2, 3, correspondientes a las clases *Negative WAR*, *Null WAR* y *Positive WAR*, respectivamente.

Para resolver esta tarea se implementaron y evaluaron tres arquitecturas de clasificación supervisada: Análisis Discriminante Lineal (LDA), Regresión Logística Multiclasa y Random Forest. Se analizaron las ventajas y limitaciones de cada modelo, tanto desde el punto de vista del desempeño predictivo como de su comportamiento ante datos ruidosos, desbalanceados y correlacionados. Se utilizó un esquema de validación sobre el conjunto de desarrollo para ajustar los modelos, y luego se realizó una evaluación final utilizando el conjunto de test.

### 2.2. Métodos

#### 2.2.1. Modelos implementados

Se implementaron tres modelos de clasificación multiclasa, adecuados para la tarea de predicción del impacto de jugadores de basketball según la clase `WAR class`:

**Análisis Discriminante Lineal (LDA):** este modelo asume que las observaciones de cada clase siguen una distribución normal multivariada con diferente media pero igual matriz de covarianza. El modelo estima las medias  $\mu_k$  y la matriz de covarianza común  $\Sigma$  sobre los datos de entrenamiento. La probabilidad de pertenecer a una clase  $k$  se obtiene evaluando la función discriminante lineal:

$$g_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k)$$

donde  $\pi_k$  es la probabilidad a priori de la clase  $k$ . El modelo predice la clase para la cual  $g_k(x)$  es máximo.

**Regresión Logística Multiclasa:** se utilizó una generalización de la regresión logística binaria mediante la función softmax. Dado un vector de parámetros  $\theta_k$  para cada clase  $k$ , la probabilidad de que una muestra  $x$  pertenezca a la clase  $k$  se calcula como:

$$P(y = k | x) = \frac{\exp(\theta_k^T x)}{\sum_{j=1}^K \exp(\theta_j^T x)}$$

El modelo fue entrenado utilizando descenso por gradiente sobre la función de pérdida de entropía cruzada con regularización L2:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K \mathbf{1}\{y^{(i)} = k\} \log P(y^{(i)} = k | x^{(i)})$$

$$+ \frac{\lambda}{2} \sum_{k=1}^K \|\theta_k\|^2$$

donde  $\mathbf{1}\{\cdot\}$  es la función indicadora,  $m$  es la cantidad de muestras,  $K$  el número de clases y  $\lambda$  el coeficiente de regularización. El entrenamiento se realizó con descenso por gradiente estándar, actualizando todos los vectores  $\theta_k$  en cada iteración.

**Random Forest:** Cada árbol se entrena sobre una muestra bootstrap del conjunto de entrenamiento. En cada nodo del árbol, se selecciona aleatoriamente un subconjunto de features y se busca el mejor umbral de división según la ganancia de información (entropía). La función de entropía se define como:

$$H(y) = - \sum_{k=1}^K p_k \log(p_k)$$

donde  $p_k$  es la proporción de ejemplos de clase  $k$  en el conjunto. Para una división en subconjuntos izquierdo  $S_L$  y derecho  $S_R$ , la ganancia de información es:

$$IG(S, A) = H(S) - \left( \frac{|S_L|}{|S|} H(S_L) + \frac{|S_R|}{|S|} H(S_R) \right)$$

La predicción final del bosque se obtiene por votación mayoritaria entre todos los árboles del ensemble.

## 2.2.2. Conjunto de Datos y Preprocesamiento

El conjunto de desarrollo fue dividido en un 80 % para entrenamiento y un 20 % para validación, mientras que el conjunto de test se reservó para la evaluación final de los modelos. Se aplicó normalización mediante *z-score* sobre todas las variables numéricas. La fórmula utilizada para normalizar cada variable  $x$  fue:

$$x'_i = \frac{x_i - \mu}{\sigma}$$

donde  $\mu$  es la media y  $\sigma$  el desvío estándar de la variable en el conjunto de entrenamiento. Los

parámetros de normalización fueron calculados únicamente sobre los datos de entrenamiento y luego aplicados tanto al conjunto de validación como al conjunto de test, evitando así cualquier fuga de información.

### 2.2.3. Información Mutua

Durante el análisis exploratorio, se calculó la información mutua entre cada feature y el target **WAR class** con el objetivo de cuantificar el grado de dependencia no lineal entre ellas. La información mutua  $I(X; Y)$  mide cuánta información comparte una variable aleatoria  $X$  con otra  $Y$ , y se define como:

$$I(X; Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x, y) \log_2 \left( \frac{P(x, y)}{P(x)P(y)} \right)$$

donde  $P(x, y)$  es la probabilidad conjunta de los valores  $x$  e  $y$ , y  $P(x)$ ,  $P(y)$  son las probabilidades marginales.

### 2.2.4. Selección de Hiperparámetros

El único hiperparámetro ajustado en forma sistemática fue el coeficiente de regularización  $\lambda$  para la regresión logística. Se exploraron distintos valores en una escala logarítmica, y se utilizó un esquema simple de validación, utilizando el F1-score como métrica objetivo. El valor de  $\lambda$  que obtuvo el mejor desempeño fue luego empleado para entrenar el modelo final sobre el conjunto completo de desarrollo.

En el caso de Random Forest, se optó por una configuración manual de hiperparámetros, ya que se observó empíricamente que el dataset presentaba una estructura que este modelo podía capturar con facilidad. Por este motivo, no se consideró necesario realizar una búsqueda exhaustiva como *grid search*, aunque dicha estrategia podría emplearse si se deseara un ajuste más fino o si el conjunto de datos fuera más complejo.

### 2.2.5. Métricas de Evaluación

Se emplearon las siguientes métricas para comparar el desempeño de los modelos: Accuracy, Precision, Recall, F1-Score, AUC-ROC (One-vs-Rest), AUC-PR (One-vs-Rest), Matriz de Confusión.

Las métricas de **accuracy**, **matriz de confusión**, **precision**, **recall** y **F1-score** fueron generalizadas al caso multiclase utilizando el promedio

macro: se calcula la métrica por separado para cada clase y luego se promedia. Por ejemplo, la precisión macro se define como:

$$\text{Precision}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^K \frac{TP_k}{TP_k + FP_k}$$

donde  $K$  es la cantidad de clases y  $TP_k$ ,  $FP_k$  son los verdaderos positivos y falsos positivos de la clase  $k$  en la matriz binarizada correspondiente.

Las métricas de **AUC-ROC** y **AUC-PR** también fueron extendidas al caso multiclase utilizando un esquema *One-vs-Rest*. Para cada clase  $k$ , se construyó una curva ROC (o PR) tratando esa clase como positiva y todas las demás como negativas. Luego, se calculó el área bajo cada curva y se promedió:

$$\text{AUC-ROC}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^K \text{AUC-ROC}_k$$

$$\text{AUC-PR}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^K \text{AUC-PR}_k$$

Las curvas ROC y PR se construyeron evaluando la probabilidad predicha para cada clase y variando el umbral de decisión.

## 2.3. Resultados

### 2.3.1. Análisis Exploratorio

Se comenzó el análisis examinando la distribución del target. Este dataset se encuentra razonablemente balanceado, con una distribución de clases del 30 % para la clase 1, 37 % para la clase 2, y el restante 33 % para la clase 3. Esto indica que no es necesario aplicar técnicas de rebalanceo para el entrenamiento de los modelos.

Respecto a la calidad de los datos, no se identificaron valores faltantes ni duplicados. Las variables presentes son todas numéricas, por lo que no fue necesario realizar codificación adicional como *one-hot encoding*. Tampoco se observaron valores atípicos significativos. Sin embargo, se detectaron algunas observaciones con valores negativos en las columnas `mp` (minutos jugados) y `poss` (posesiones), que carecen de interpretación práctica. Estas filas fueron eliminadas directamente del conjunto de datos.

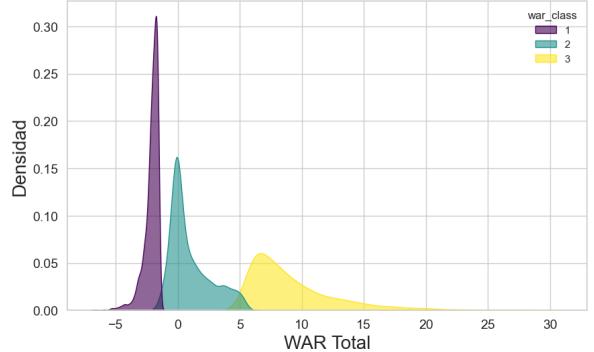


Figura 5: Distribución de la variable `WAR total` dividiendo por colores cada clase.

En la Figura 5 puede observarse que la variable `war total` presenta una separación muy marcada entre las clases de la variable objetivo `WAR class`. Esta relación tan directa facilita la predicción de la clase de forma casi trivial, ya que los valores de `war total` caen en rangos claramente diferenciables para cada clase. Esto fue respaldado también por la información mutua (Sección A.3), que mostró una dependencia extremadamente fuerte con la variable objetivo. Este es un caso claro de *data leakage*, ya que el modelo podría simplemente aprender esta regla sin necesidad de capturar relaciones más complejas o relevantes del resto de las variables.

Por esta razón, se decidió eliminar la variable `war total` del conjunto de atributos antes de entrenar los modelos, con el fin de evitar un sobreajuste artificial y permitir una evaluación más honesta del poder predictivo real de las demás variables.

Una vez finalizado el entrenamiento y la selección de hiperparámetros, se procedió a evaluar el rendimiento final de los modelos sobre el conjunto de test provisto. Para la regresión logística multiclase se utilizó el mejor valor de regularización obtenido mediante barrido simple, con  $\lambda = 0.0001$ , y para nuestra implementación final de Random Forest utilizamos como hiperparámetros: cantidad de árboles:  $n_{\text{trees}} = 10$ , la cantidad mínima de muestras por nodo para dividir: `min_samples_split = 2`, y no usamos un límite explícito en la profundidad de los árboles ni en la cantidad de features considerados por `split`

Estos parámetros resultaron en un excelente desempeño, y dada la simplicidad del ajuste, favorecieron además la velocidad de entrenamiento y evaluación del modelo. En todos los casos, se aplicó el mismo preprocesamiento definido durante el desarrollo

Métrica	LDA	Regresión Logística	Random Forest
Accuracy	0.91	0.89	0.96
Precision	0.92	0.90	0.96
Recall	0.92	0.90	0.96
F1 Score	0.91	0.89	0.96
AUC ROC	0.98	0.96	0.99
AUC PR	0.93	0.92	0.98

Cuadro 4: Comparación de métricas de validación para los tres modelos evaluados en Test.

La Tabla 4 muestra que el modelo Random Forest obtuvo el mejor desempeño en todas las métricas. Esto se debe a su capacidad para capturar relaciones no lineales y realizar múltiples particiones del espacio de entrada, lo que le permite adaptarse mejor a las complejidades del dataset.

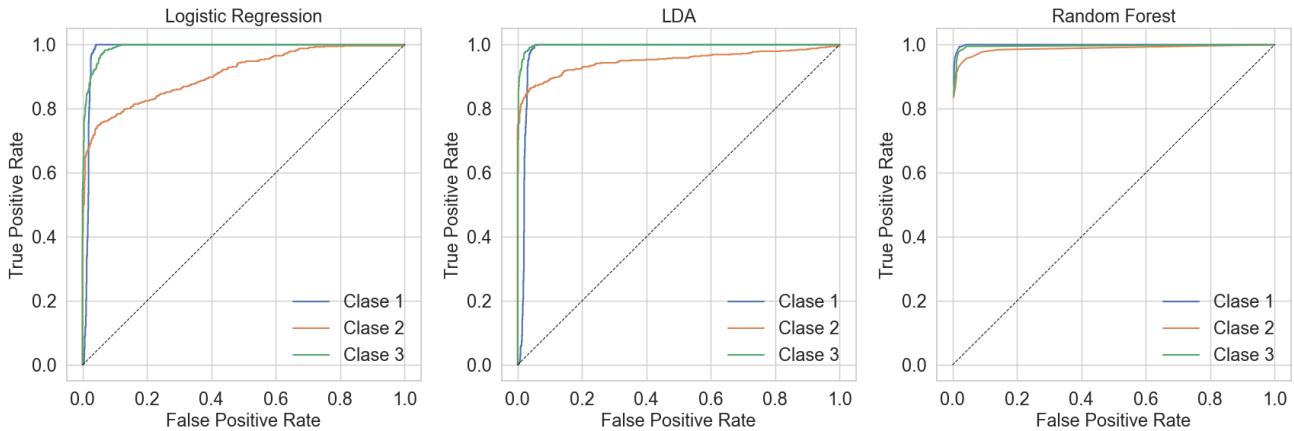


Figura 6: Curvas ROC One-vs-Rest para Regresión Logística, LDA y Random Forest.

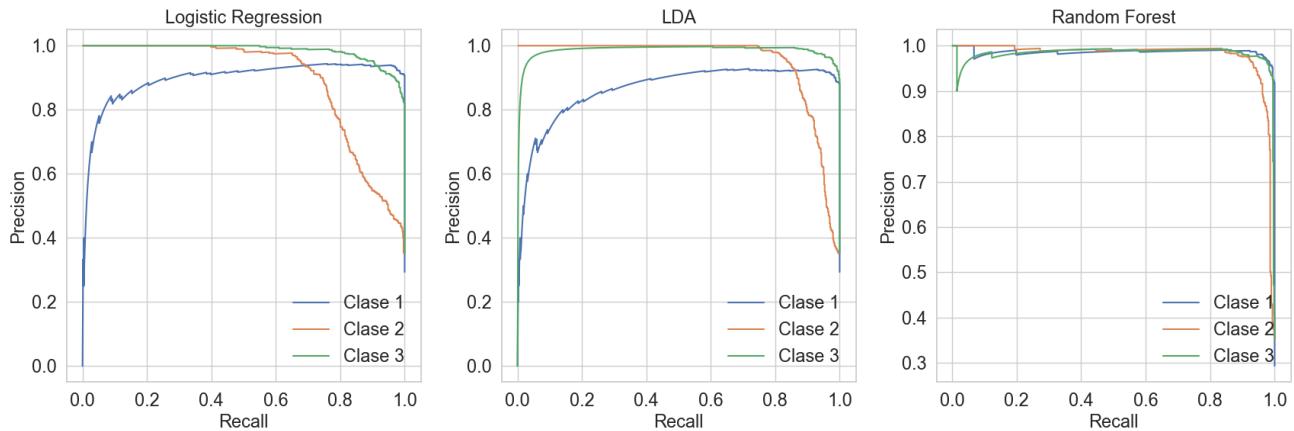


Figura 7: Curvas Precision-Recall One-vs-Rest para Regresión Logística, LDA y Random Forest.

Como se puede apreciar en las Figuras 6 y 7 las curvas muestran que Random Forest no sólo domina en términos de área bajo la curva (AUC), sino que mantiene una alta precisión incluso para niveles elevados de recall. Esto es especialmente importante en contextos donde todas las clases tienen relevancia.

Por último, en la Figura 8 se incluye una comparación visual de las matrices de confusión de los tres modelos, que permiten identificar con mayor claridad los patrones de error, donde se observa que tanto LDA como Random Forest logran una correcta clasificación en la mayoría de los casos, pero es Random Forest el que presenta menos errores de confusión entre las clases 2 y 3, que eran las más difíciles de

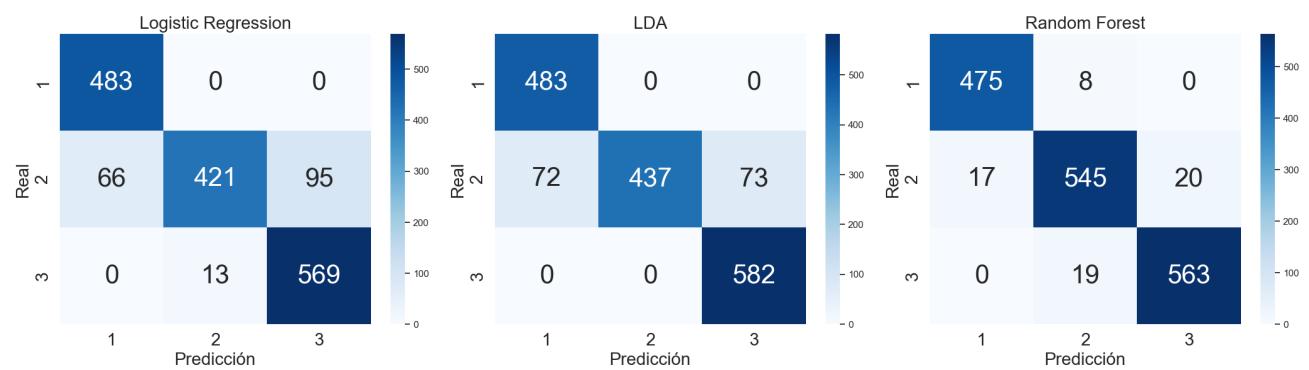


Figura 8: Matrices de confusión normalizadas para los tres modelos.

distinguir. Esto refuerza su superioridad como modelo final seleccionado.

La elección del modelo a producción recae entonces de forma natural en Random Forest. Además de su excelente performance, tiene la ventaja de ser relativamente robusto frente a outliers y no requiere un gran preprocesamiento, lo cual simplifica su integración en un pipeline automatizado. A pesar de que es más costoso computacionalmente que la Regresión Logística o LDA, este aspecto no se considera limitante en este caso debido al tamaño moderado del dataset y a que no se requiere inferencia en tiempo real.

## A. Apéndice

### A.1. Pairplot dataset procesado

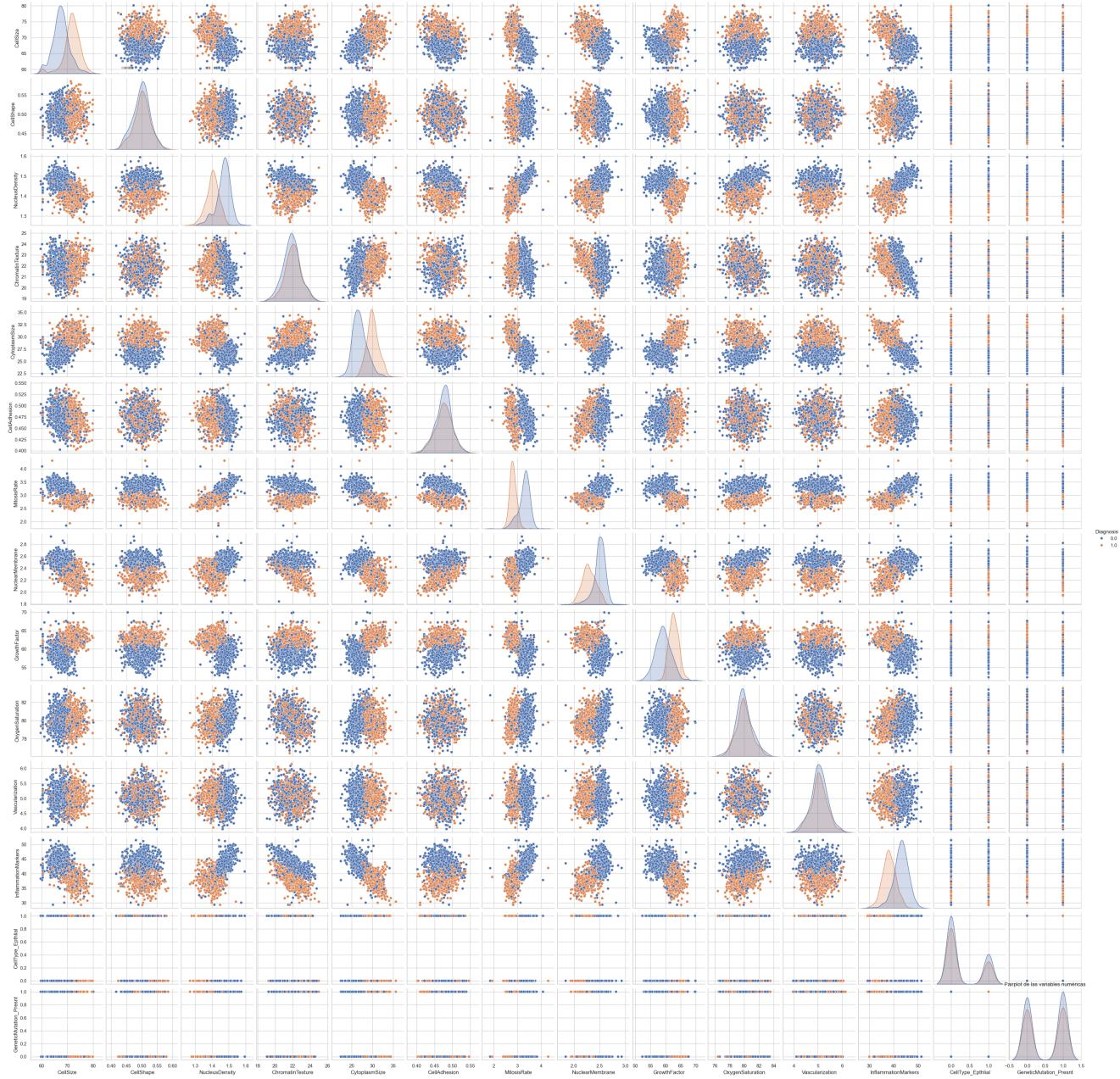


Figura 9: Pairplot del dataset completo procesado.

Como se observa en la Figura 9, los datos presentan un buen nivel de preprocesamiento, con variables ya estandarizadas y sin presencia evidente de outliers. Además, se puede notar que las dos clases (representadas por los colores azul y naranja) resultan claramente separables en varios pares de variables, lo cual sugiere que el problema es linealmente separable en buena medida. Esta propiedad facilita el trabajo de los clasificadores y es indicativa de un conjunto de datos bien estructurado.

## A.2. Matriz de Correlación

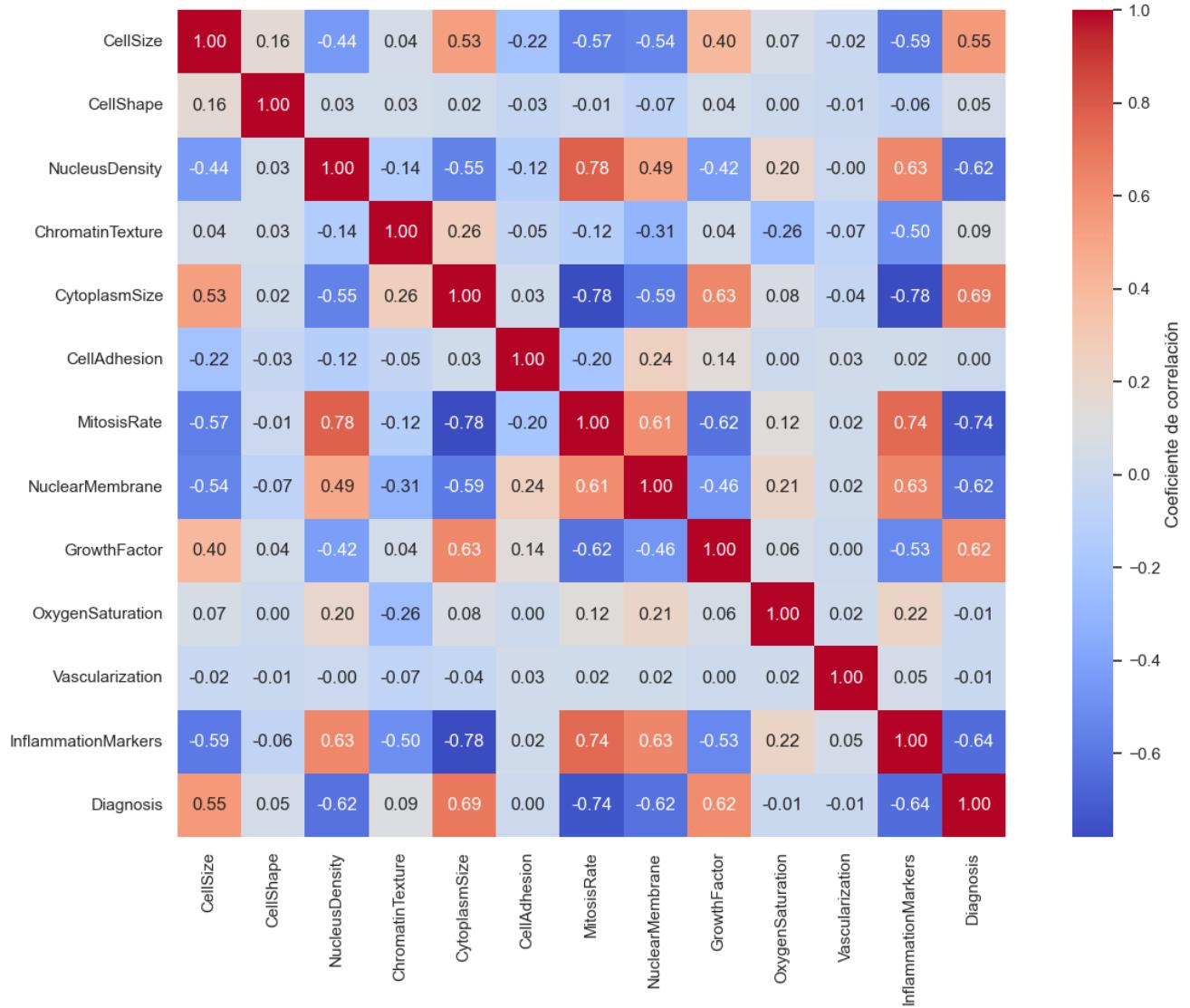


Figura 10: Matriz de correlación de Pearson entre las variables numéricas del dataset balanceado.

En la Figura 10 se pueden identificar algunas relaciones fuertes entre variables, como por ejemplo una alta correlación positiva entre *MitosisRate* e *InflammationMarkers* ( $\rho = 0,74$ ), y una correlación negativa entre *CytoplasmSize* y *MitosisRate* ( $\rho = -0,78$ ). También se observa que varias variables están moderadamente correlacionadas con la variable objetivo *Diagnosis*, como *CytoplasmSize* ( $\rho = 0,69$ ), *MitosisRate* ( $\rho = -0,74$ ) e *InflammationMarkers* ( $\rho = -0,64$ ), lo cual sugiere que podrían tener un impacto importante en el proceso de predicción.

### A.3. Análisis de la variable war\_total

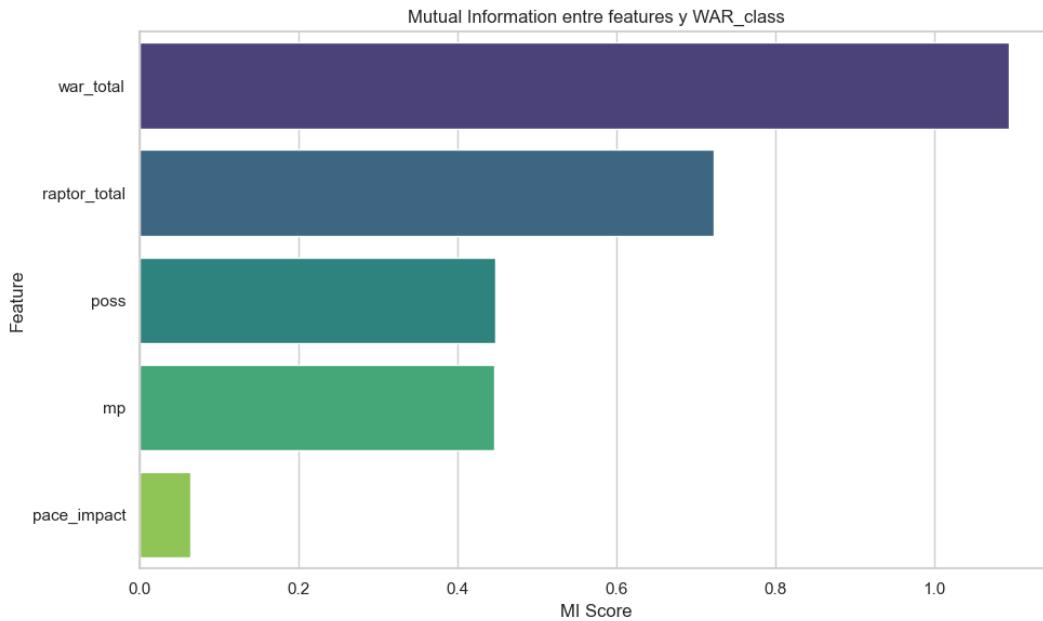


Figura 11: Mutual Information entre cada feature y la clase objetivo WAR class.

Analizamos la información mutua entre cada variable y el target. Como puede observarse en la Figura 11, la variable `war total` presenta una dependencia muy fuerte con el target. Esto se debe a que la clase `WAR class` fue construida directamente a partir del valor de `war total`: si `war total` es negativo, `WAR class` toma el valor 1; si es igual a 0, toma el valor 2; y si es positivo, toma el valor 3. Esto representa un caso claro de *data leakage*, ya que el modelo podría aprender de forma directa la regla de asignación de clase sin necesidad de capturar relaciones más profundas entre las variables.

Por esta razón, se decidió eliminar la variable `war total` del conjunto de atributos antes de entrenar los modelos, permitiendo así una evaluación más realista del poder predictivo del resto de las variables.