I302 - Aprendizaje Automático
y Aprendizaje Profundo

Trabajo Práctico 2:

Clasificación y Ensemble Learning

Martín Bianchi

14 de abril de 2025

Ingeniería en Inteligencia Artificial

1. Diagnóstico de Cáncer de Mama

En este trabajo se abordó el problema de diagnóstico de cáncer de mama como una tarea de clasificación binaria, buscando predecir si un tumor es benigno o maligno a partir de datos clínicos y genéticos. Para ello, se realizó un exhaustivo preprocesamiento de los datos, incluyendo imputación de valores faltantes, transformación de variables y normalización. Se entrenó un modelo de regresión logística penalizada, cuya performance fue evaluada mediante métricas estándar como F-score, AUC-ROC y AUC-PR. Además, se probaron distintas estrategias de rebalanceo (undersampling, oversampling, SMOTE y cost-sensitive learning) para mitigar el desbalance de clases. El modelo con aprendizaje sensible al costo mostró el mejor equilibrio entre recall y precisión, reduciendo falsos negativos sin un aumento significativo de falsos positivos, lo que lo convierte en una herramienta prometedora para apoyar el diagnóstico médico.

1.1. Introducción

El cáncer de mama es uno de los tipos de cáncer más comunes y una de las principales causas de mortalidad en mujeres a nivel mundial. La detección temprana resulta fundamental para mejorar el pronóstico y la supervivencia, por lo que en los últimos años se han desarrollado herramientas basadas en inteligencia artificial para asistir en su diagnóstico.

En este trabajo abordamos el problema como una **tarea de clasificación binaria**, en la que se busca predecir si un tumor es *benigno* o *maligno* a partir de información clínica y genética de cada paciente. El enfoque se centra en el uso de modelos de *machine learning*, capaces de aprender patrones a partir de los datos y realizar predicciones con alta precisión.

Los datos utilizados presentan desafíos típicos de entornos médicos: múltiples variables numéricas y categóricas, valores faltantes, presencia de outliers y un marcado desbalance entre clases. El desarrollo del modelo requirió un preprocesamiento cuidadoso, incluyendo la transformación de variables, la imputación de valores faltantes y la normalización de datos.

Una vez preparado el conjunto de datos, se entrenó un modelo de regresión logística penalizada, evaluando su desempeño mediante métricas estándar. Además, se exploraron diferentes estrategias de rebalanceo para mejorar su comportamiento ante clases desbalanceadas. El objetivo final es construir un modelo capaz de generalizar correctamente y que minimice errores críticos, como los falsos negativos, que en contextos médicos pueden tener consecuencias graves.

1.2. Métodos

1.2.1. Datos y preprocesamiento

Conjunto de datos. Se utilizó un dataset clínico para diagnóstico de cáncer de mama, dividido originalmente en dos subconjuntos: `dev` y `test`. A partir de `dev`, se realizó una partición en **80 % para entrenamiento** y **20 % para validación**, mientras que el conjunto `test` se mantuvo separado hasta la evaluación final.

Codificación de variables. Las variables categóricas fueron transformadas para su uso en modelos

numéricos. La variable `cell_type`, de tipo multiclase sin orden, fue codificada mediante **one-hot encoding**. En cambio, `genetic_mutation`, de tipo binario, fue simplemente **binarizada** (presencia o ausencia de mutación).

Tratamiento de outliers. Inicialmente se intentó detectar valores atípicos con el **rango intercuartílico** (IQR), definiendo outliers como aquellos fuera del rango $[Q1 - 1{,}5 \cdot \text{IQR},\ Q3 + 1{,}5 \cdot \text{IQR}]$. Sin embargo, este criterio eliminaba demasiados valores plausibles, por lo que se optó por una estrategia más conservadora: se definieron **límites manuales** para cada variable a partir de la inspección visual de los `boxplots`. Los valores considerados fuera de rango fueron reemplazados por `NaN` para ser imputados más adelante.

Imputación y normalización. Para completar los valores faltantes se utilizó un **KNN Imputer**, aprovechando la similitud entre muestras. Para estabilizar el proceso, los `NaN` se reemplazaron temporalmente por la mediana, y luego se imputó cada variable en función de sus vecinos más cercanos en el espacio normalizado. La **normalización** se aplicó previamente utilizando `RobustScaler` para variables con valores extremos y `MinMaxScaler` para aquellas con rangos bien definidos, siempre tomando como referencia el conjunto de entrenamiento para evitar *data leakage*.

1.2.2. Modelo y entrenamiento

Algoritmo de clasificación. Se entrenó un modelo de **regresión logística** para clasificación binaria. Este modelo estima la probabilidad de pertenencia a la clase positiva mediante la función sigmoidea:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad \text{donde } z = \mathbf{w}^\top \mathbf{x} + b$$

La pérdida se calculó con **binary cross-entropy** regularizada:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] + \lambda \|\mathbf{w}\|_2^2$$

donde λ es el coeficiente de regularización L2.

Selección de hiperparámetros. Se exploraron distintas combinaciones de `learning rate`, `regularización L2` y `umbral de clasificación`. No se utilizó validación cruzada, sino una validación simple con el conjunto reservado (*hold-out*), ya que la validación cruzada aumentaba considerablemente los tiempos de entrenamiento e impedía evaluar una mayor cantidad de hiperparámetros.

En particular, se evaluaron tasas de aprendizaje en `[0.1, 0.01, 0.001]`, coeficientes de regularización L2 en `[0, 1e-4, 1e-3, 1e-2, 0.1, 1, 10, 100]` y umbrales de clasificación en 100 valores equiespaciados entre 0 y 1. Los mejores resultados se obtuvieron con la combinación que maximizó el *F-score* sobre el conjunto de validación. Estos fueron:

```
lr = 0.01, L2 = 0.1, threshold = 0.87
```

Métricas de evaluación. Se utilizaron las siguientes métricas sobre `validation` y `test`:

- **F-score**: promedio armónico entre precisión y recall.
- **Recall (sensibilidad)**: proporción de positivos correctamente identificados.
- **Precisión**: proporción de positivos predichos que son correctos.
- **Accuracy**: proporción de clasificaciones correctas.
- **AUC-ROC**: área bajo la curva ROC, mide discriminación general.
- **AUC-PR**: área bajo la curva precisión-recall, más informativa con clases desbalanceadas.

1.2.3. Métodos de balanceo

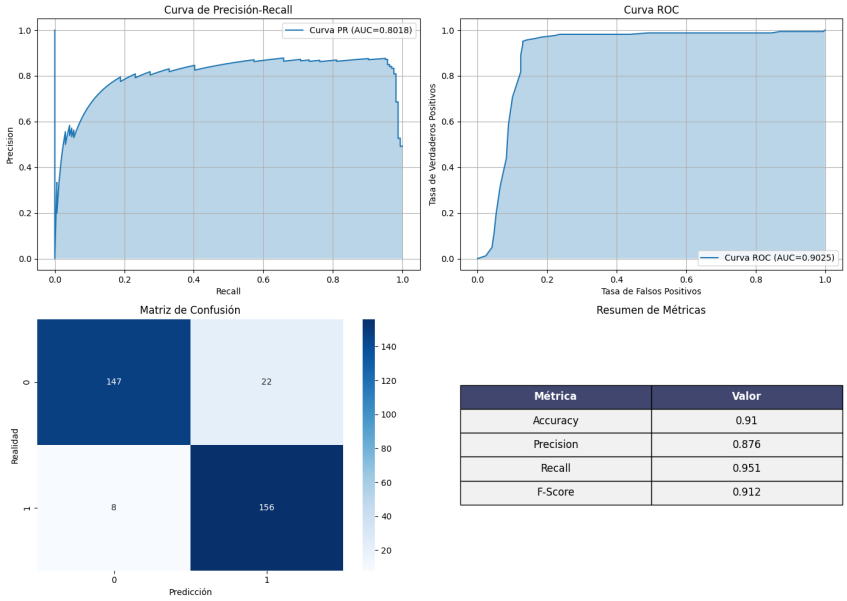
Dataset desbalanceado. Se repitió el mismo pipeline sobre una versión artificialmente desbalanceada del conjunto. Para abordar el desbalance, se aplicaron cinco técnicas:

- **Sin balanceo**: se entrena directamente con los datos tal como están, lo que suele favorecer la clase mayoritaria.
- **Undersampling**: se eliminan muestras de la clase mayoritaria para igualar la proporción con la minoritaria.
- **Oversampling**: se duplican (o se agregan con reemplazo) muestras de la clase minoritaria para balancear.
- **SMOTE**: se generan nuevas muestras sintéticas de la clase minoritaria interpolando entre vecinos cercanos.
- **Cost-sensitive learning**: se asigna mayor penalización a los errores sobre la clase minoritaria mediante pesos en la función de pérdida.

1.3. Resultados

1.3.1. Evaluación del modelo base

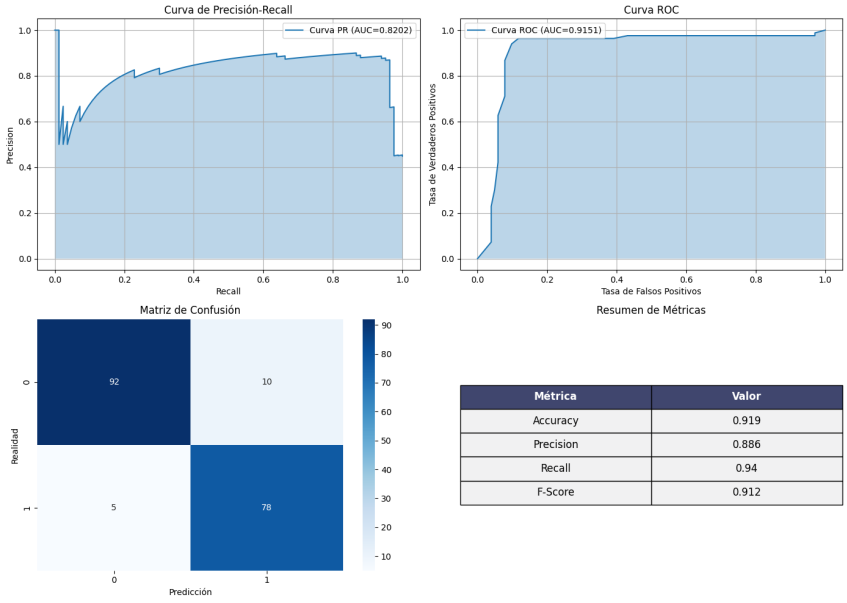
El primer modelo entrenado fue una regresión logística penalizada (con regularización L2), aplicada sobre los datos *sin rebalanceo*. En la **Figura 1 y 2** se presentan los resultados obtenidos en validación y test, incluyendo curvas ROC y PR, matrices de confusión y métricas relevantes.



Métrica	Valor
Accuracy	0.91
Precision	0.876
Recall	0.951
F-Score	0.912

Figura 1: Resultados del modelo base sin rebalanceo: conjunto de validacion.

El desempeño fue muy similar en ambos conjuntos, lo que indica una buena capacidad de generalización. Las curvas muestran que el modelo distingue correctamente entre las clases, y el alto valor de **recall** refleja su efectividad para detectar casos positivos. Esto es esencial en un contexto clínico, donde los **falsos negativos** pueden tener consecuencias graves al omitir un diagnóstico de cáncer.



Métrica	Valor
Accuracy	0.919
Precision	0.886
Recall	0.94
F-Score	0.912

Figura 2: Resultados del modelo base sin rebalanceo: conjunto de test.

La matriz de confusión también evidencia este comportamiento: hay muy pocos falsos negativos en ambos conjuntos. Sin embargo, el modelo comete una cantidad moderada de **falsos positivos**, lo que implica una menor precisión. En este tipo de problema, ese compromiso es aceptable: es preferible generar una alarma falsa que dejar pasar un caso real. El F-score elevado y la consistencia entre validación y test

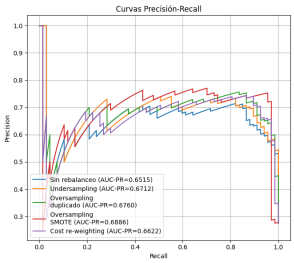
refuerzan la solidez del modelo base.

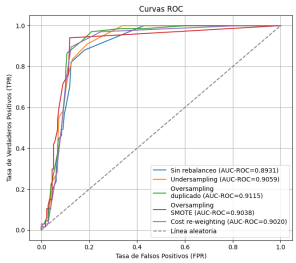
1.3.2. Comparación de estrategias de rebalanceo

Con el objetivo de mejorar la detección de la clase minoritaria, se aplicaron cinco estrategias de rebalanceo: sin rebalanceo, *undersampling*, *oversampling* por duplicación, *SMOTE*, y *cost-sensitive learning*. Los resultados se presentan en las Figuras 3 (validación) y 4 (test).

En validación, se observa que técnicas como SMOTE y reponderación por costo lograron mejorar notablemente el **recall**, reduciendo aún más los falsos negativos. Esto sugiere que estos enfoques logran una mayor sensibilidad, lo cual es prioritario en problemas de diagnóstico. A medida que se aplican estrategias más sofisticadas, el F-score mejora de manera consistente, aunque con cierto costo en la precisión debido al incremento de falsos positivos. Este compromiso puede ser válido si el objetivo principal es minimizar omisiones de casos reales.

En test, se confirma la robustez de estas estrategias: el modelo con **cost-sensitive learning** no solo mantiene un F-score elevado, sino que además presenta un excelente equilibrio entre sensibilidad y precisión. Esto indica que no solo aprendió a detectar mejor los positivos, sino que también logró controlar el número de alarmas falsas, lo que lo convierte en una opción sólida para implementación real.

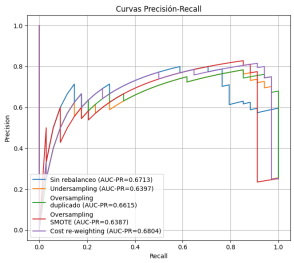


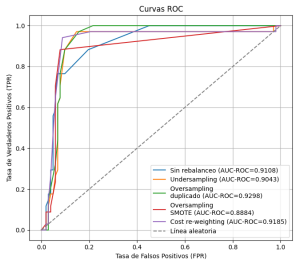


Resumen de Métricas

Modelo	Accuracy	Precision	Recall	F-Score	AUC-ROC	AUC-PR
Sin rebalanceo	0.861	0.714	0.821	0.764	0.893	0.652
Undersampling	0.861	0.709	0.836	0.767	0.906	0.671
Oversampling duplicado	0.885	0.753	0.866	0.806	0.911	0.676
Oversampling SMOTE	0.898	0.75	0.94	0.834	0.904	0.689
Cost re-weighting	0.881	0.732	0.896	0.805	0.902	0.662

Figura 3: Comparación de estrategias de rebalanceo sobre el conjunto de validación.





Resumen de Métricas

Modelo	Accuracy	Precision	Recall	F-Score	AUC-ROC	AUC-PR
Sin rebalanceo	0.868	0.722	0.765	0.743	0.911	0.671
Undersampling	0.897	0.75	0.882	0.811	0.904	0.64
Oversampling duplicado	0.897	0.75	0.882	0.811	0.93	0.662
Oversampling SMOTE	0.912	0.789	0.882	0.833	0.888	0.639
Cost re-weighting	0.919	0.78	0.941	0.853	0.919	0.68

Figura 4: Desempeño en test de los modelos con distintas estrategias de rebalanceo.

En resumen, la estrategia que mostró el mejor desempeño general fue **cost-sensitive learning**. No solo obtuvo el mayor F-score en test, sino que también logró un excelente equilibrio entre precisión y recall, con valores altos de AUC-ROC y AUC-PR. Esto indica que el modelo aprendió a detectar los casos positivos sin disparar en exceso la cantidad de falsos positivos, lo cual es ideal en contextos clínicos. Por el contrario, el modelo sin rebalanceo fue el que tuvo el peor desempeño relativo. Si bien mostró buen rendimiento, fue superado sistemáticamente por los demás en recall y sensibilidad. Esto puede atribuirse

al desbalance de clases original, que lleva al modelo a favorecer la clase mayoritaria y, por lo tanto, a omitir casos positivos. Las estrategias de rebalanceo logran contrarrestar este sesgo y mejorar la detección de la clase minoritaria.

2. Rendimiento de Jugadores de Basketball

En este trabajo se abordó un problema de clasificación multiclase con el objetivo de predecir el impacto de jugadores de basketball a partir de métricas individuales. Para ello, se utilizaron registros históricos con variables numéricas que describen el rendimiento de cada jugador, y se entrenaron tres modelos: Regresión Logística Multiclase, Análisis Discriminante Lineal (LDA) y Random Forest. Durante el preprocesamiento se eliminaron valores inválidos, se descartaron variables redundantes, se normalizaron los datos con RobustScaler y se aplicó SMOTE para balancear las clases. Luego de ajustar los hiperparámetros mediante validación simple, se evaluó el desempeño de los modelos con métricas como precisión, recall, F1-score y curvas ROC/PR. Los resultados mostraron que Random Forest fue el modelo con mejor capacidad de generalización, alcanzando un F1-score de 0.963 y un AUC cercano a 1 sobre el conjunto de test.

2.1. Introducción

En este trabajo se aborda un problema de **clasificación multiclase**, cuyo objetivo es predecir el rendimiento de jugadores de basketball a partir de métricas individuales registradas a lo largo de varias temporadas. La variable objetivo es `WAR_class`, que representa una categorización del *Wins Above Replacement* (WAR), una métrica que estima el impacto de un jugador en términos de victorias aportadas por encima de lo que brindaría un suplente promedio.

Las clases posibles son tres: *Negative WAR*, *Null WAR* y *Positive WAR*, correspondientes a las clases 1, 2 y 3 respectivamente.

La entrada del algoritmo está compuesta por diversas variables numéricas que describen el rendimiento de cada jugador, como posesiones, puntos, rebotes, asistencias, entre otras. La salida es una predicción de clase que indica el nivel de impacto que tuvo el jugador según su desempeño.

Durante la etapa de preprocesamiento, se detectaron algunos valores imposibles (como posesiones negativas) que fueron eliminados, y se descartó la variable `war_total` al detectar que contenía información derivada de la clase objetivo. Para manejar el desbalance entre clases, se aplicó la técnica *SMOTE*, que genera ejemplos sintéticos de las clases minoritarias. Además, los datos fueron normalizados utilizando *RobustScaler*, dada la presencia de valores atípicos asociados a jugadores con rendimientos extraordinarios.

Se entrenaron tres modelos de *machine learning*: **Random Forest**, **Regresión Logística Multiclase** y **Análisis Discriminante Lineal (LDA)**. Luego de ajustar hiperparámetros y evaluar el desempeño en un conjunto de validación, se concluyó que Random Forest fue el modelo más robusto y preciso, mostrando un rendimiento equilibrado en términos de precisión, recall y F1-score. Finalmente, los modelos fueron evaluados sobre un conjunto de test reservado exclusivamente para medir su capacidad de generalización.

2.2. Métodos

2.2.1. Datos y preprocesamiento

Conjunto de datos. Se trabajó con un dataset de jugadores de basketball con métricas individuales recolectadas a lo largo de varias temporadas. Se dividió el conjunto `dev` en un **80 % para entrenamiento** y **20 % para validación**, mientras que el conjunto `test` se utilizó exclusivamente para la evaluación final.

Análisis exploratorio. No se encontraron valores nulos, pero sí registros inválidos (como posesiones negativas), que fueron eliminados. Se generaron gráficos de correlación, `boxplots` y `pairplots` para comprender mejor la estructura del dataset. Se eliminó la variable `war_total` ya que presenta una correspondencia directa con la variable objetivo `war_class`. Al contener implícitamente la información de la clase, su inclusión como predictor introduce una fuga de información (*data leakage*), afectando la validez del modelo.

Tratamiento de outliers. Se decidió conservar valores extremos razonables, asumiendo que reflejan rendimientos excepcionales, y eliminar solo aquellos que eran físicamente imposibles.

Balanceo y normalización. Debido al desbalance en las clases, se utilizó **SMOTE** para generar muestras sintéticas de las clases minoritarias. Luego se aplicó `RobustScaler` para normalizar las variables, ya que este método es más robusto frente a outliers.

2.2.2. Modelos implementados

Se implementaron tres modelos de clasificación multiclase:

1. Regresión Logística Multiclase. Este modelo estima probabilidades mediante la función softmax:

$$\hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, \quad z_k = \mathbf{w}_k^\top \mathbf{x} + b_k$$

donde $K = 3$ es la cantidad de clases. La pérdida utilizada fue la **cross-entropy multiclase** con regularización L2:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log \hat{y}_{ik} + \lambda \sum_{k=1}^{K} \|\mathbf{w}_k\|_2^2$$

2. Random Forest. Es un modelo de **ensamble** que combina múltiples árboles de decisión independientes. Cada árbol se entrena sobre una muestra distinta del dataset generada mediante *bootstrap* (muestreo con reemplazo), y en cada nodo del árbol se considera solo un subconjunto aleatorio de features para realizar la partición. Esto introduce diversidad entre los árboles, lo que mejora la generalización del modelo y reduce el sobreajuste. Durante el entrenamiento, los árboles dividen el espacio de decisiones eligiendo la mejor partición según una medida de **impureza**, como la **entropía**. La predicción final se obtiene por **votación mayoritaria** entre todos los árboles del bosque. Random Forest no requiere normalización de los datos y maneja bien tanto variables numéricas como categóricas, además de ser robusto frente a outliers y datos faltantes.

3. Análisis Discriminante Lineal (LDA). Este modelo asume que las clases se distribuyen normalmente con igual matriz de covarianza. Busca proyectar los datos en un subespacio que maximiza la separación entre clases:

$$\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2)$$

2.2.3. Selección de hiperparámetros

Se realizó búsqueda por grilla sobre los siguientes hiperparámetros:

- **Regresión logística:** `learning_rate` $\in \{0{,}1, 0{,}01, 0{,}001\}$, λ (L2) $\in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 0{,}1, 1, 10\}$
- **Random Forest:** `n_trees` $\in \{5, 10, 20\}$, `max_depth` $\in \{5, 10, 15\}$, `min_samples_split` $\in \{2\}$

No se utilizó validación cruzada, sino una validación simple con el conjunto reservado (*hold-out*), ya que la validación cruzada aumentaba considerablemente los tiempos de entrenamiento e impedía evaluar una mayor cantidad de hiperparámetros. Los mejores resultados se obtuvieron con:

- **Logistic Regression:** `lr = 0.1`, `lambda = 0`, con un F-score de 0.913
- **Random Forest:** `n_trees = 20`, `max_depth = 15`, `min_samples_split = 2`, con un F-score de 0.955

2.2.4. Métricas de evaluación

Los modelos fueron evaluados según las siguientes métricas:

- **Accuracy**: proporción de instancias correctamente clasificadas.
- **Precision, Recall y F-score**: promedio no ponderado entre clases.
- **Matriz de confusión**: para observar el tipo de errores más frecuentes.
- **Curva ROC**: muestra la relación entre tasa de verdaderos positivos y tasa de falsos positivos para diferentes umbrales.
- **Curva PR**: compara precisión y recall, especialmente útil en datasets desbalanceados.

Los resultados sobre el conjunto de test confirmaron que Random Forest fue el modelo más robusto y preciso, manteniendo métricas consistentes con las obtenidas en validación.

2.2.5. Resultados y evaluación de los modelos

El análisis de desempeño sobre el conjunto de validación revela claras diferencias entre los modelos evaluados. **Random Forest** se destaca notablemente frente a LDA y Logistic Regression, como se muestra en la Figura 5. En las curvas de precisión-recall y ROC, mantiene valores más altos de AUC, lo que indica mejor capacidad de discriminación en todas las clases. El resumen de métricas confirma esta tendencia, con mayores valores en todas las categorías evaluadas.

Las matrices de confusión muestran que Random Forest comete menos errores de clasificación, especialmente en la clase 2, donde LDA y Logistic presentan mayores confusiones. Si bien Logistic supera

levemente a LDA en recall y F1, ambos modelos quedan por debajo de Random Forest en todas las métricas.

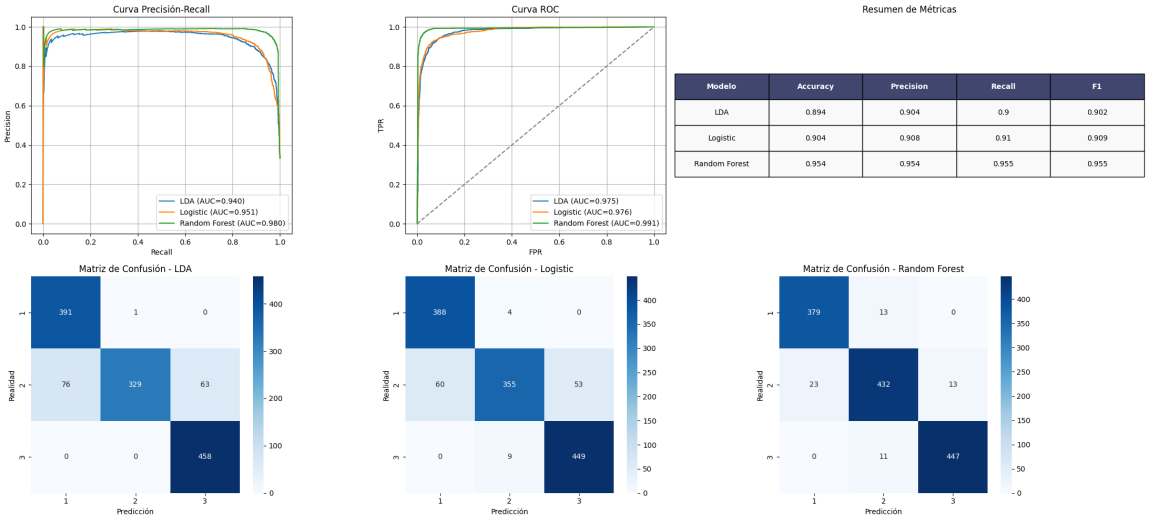


Figura 5: Desempeño de los modelos en validación: métricas de precisión, recall y F1.

Al evaluar el desempeño sobre el conjunto de test, se mantiene la superioridad del modelo **Random Forest** frente a LDA y Logistic Regression. En la Figura 6, sus curvas ROC y precisión-recall muestran los mayores valores de AUC, y el resumen de métricas confirma su ventaja en todas las categorías: precisión, recall, F1 y exactitud.

Las matrices de confusión reflejan una menor cantidad de errores, especialmente en la clase 2, donde los otros modelos presentan mayores confusiones. Logistic mejora levemente respecto a LDA, pero ambos se mantienen por debajo del rendimiento alcanzado por Random Forest.

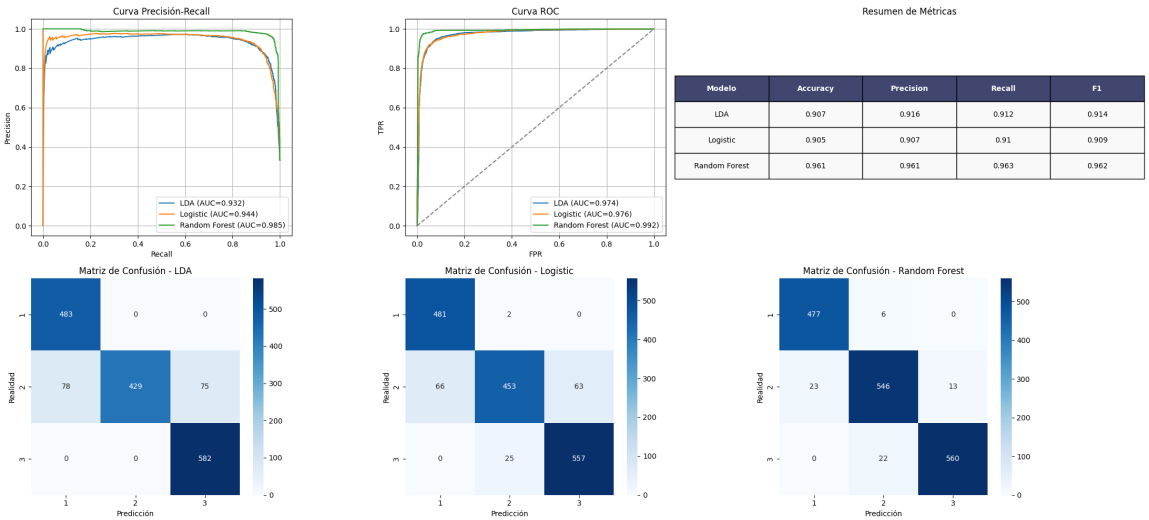


Figura 6: Resultados de los modelos en test antes del reentrenamiento.

La Figura 7 presenta el rendimiento final sobre el conjunto de test, luego de entrenar con la combinación de train y validación. El modelo **Random Forest** sigue siendo el más robusto, con métricas consistentemente

superiores: **Accuracy** de 0.962, **F1** de 0.963 y valores de **AUC** cercanos a 1 en ambas curvas ROC y precisión-recall.

Las matrices de confusión confirman una menor tasa de error en todas las clases, especialmente en la clase 2, donde los modelos LDA y Logistic muestran más confusiones. Aunque los tres modelos se desempeñan razonablemente bien, Random Forest destaca por su capacidad para minimizar errores incluso en clases más difíciles de predecir.

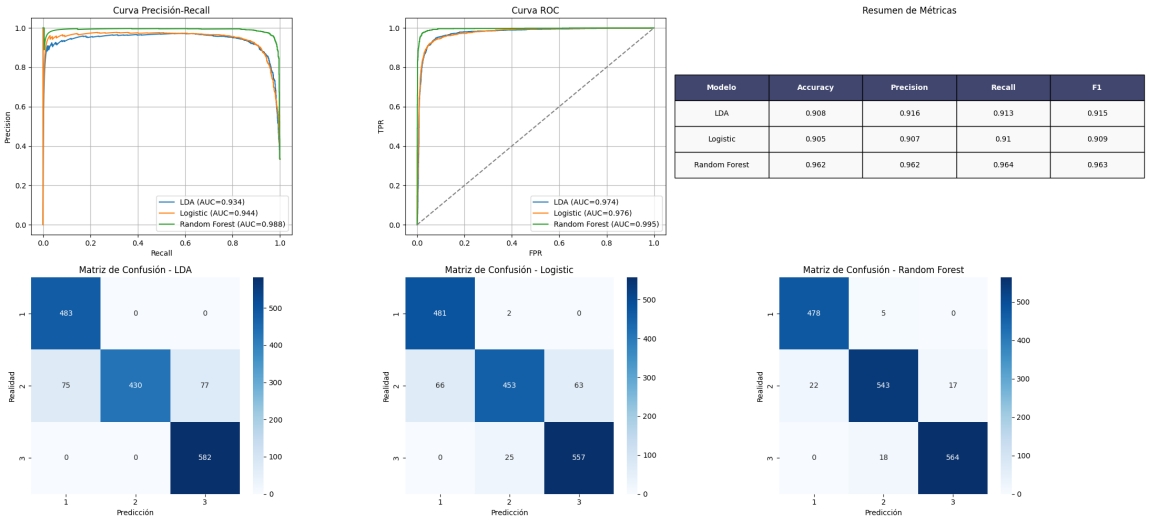


Figura 7: Resultados finales en test tras reentrenar con train+validation.

El rendimiento superior de Random Forest puede atribuirse a su capacidad para manejar relaciones no lineales y capturar interacciones complejas entre las características, lo cual es crucial en problemas de clasificación multiclase. Por otro lado, el desempeño moderado de Logistic Regression y LDA se explica por su naturaleza más lineal y sus limitaciones al trabajar con datos más complejos.

El modelo que funcionó mejor fue **Random Forest**, alcanzando los mejores resultados en todas las métricas evaluadas, incluyendo accuracy, precisión, recall, F1 y AUC, tanto en validación como en test. Por el contrario, el modelo que tuvo el peor desempeño fue **Logistic Regression**, con métricas levemente inferiores, especialmente en precisión y F1, posiblemente debido a su naturaleza lineal, que limita su capacidad para modelar patrones complejos en los datos. LDA se ubicó en un punto intermedio, mostrando un rendimiento razonable, aunque menos competitivo que Random Forest frente a clases con mayor solapamiento.