Comparación de Técnicas de Aprendizaje Automático Supervisado Aplicados a Datos Cardiólogos

Autor: Nicolás Seivane
Tutora: Andrea Rey

Fecha
Universidad Nacional de Hurlingham

Índice General

Índice de Figuras

Índice de Tablas

Resumen

Chapter 1

Introducción

El objetivo general de este trabajo es comparar el rendimiento de diversas técnicas de Aprendizaje Automático Supervisado con el fin de recomendar aquella que presente el mejor desempeño al aplicarse sobre un conjunto de datos cardiológicos. Se expondrán las técnicas empleadas y las métricas utilizadas para poder determinar cuál de ellas obtiene los valores más favorables y, en consecuencia, resulta más adecuada para el problema planteado.

1.1 Motivación

El proceso de diagnóstico médico puede ser extenso, incluso contando con la mejor disposición del personal de salud, ya que con frecuencia requiere la recopilación y análisis de datos provenientes de distintos estudios. El propósito de este trabajo es contribuir a agilizar dicho proceso, identificando técnicas que puedan facilitar el diagnóstico médico. No solo resulta fundamental la posibilidad de obtener diagnósticos más ágiles, sino también la de reconocer qué atributos o características de los estudios resultan más significativos que otros para un diagnóstico determinado.

1.2 Estado del Arte

Las técnicas de Aprendizaje Automático (ML) se utilizan cada vez más en la investigación cardiovascular. El trabajo de Isaksen et al. [7](2025) presenta recomendaciones y orientaciones para la evaluación adecuada de modelos de aprendizaje automático supervisado en cardiología, destacando los problemas específicos asociados con estas técnicas, como la fuga de datos (*data leakage*) y el desequilibrio de clases. Por su parte, el documento de Kumar y Kumar [8](2021) revisa las metodologías de ML para el diagnóstico de cardiopatías utilizando métodos no invasivos (NI), un área crucial dada la alta mortalidad anual (17.9 millones de personas) asociada con los problemas cardíacos.

1.3 Conjuntos de datos Utilizados

Para la realización de este trabajo se exploraron diversas plataformas en busca de conjuntos de datos reales que resultaran relevantes para el estudio mediante técnicas de aprendizaje automático. Durante esta búsqueda se identificaron múltiples *datasets* de distinta naturaleza: algunos correspondientes a problemas de **clasificación binaria**, donde las observaciones se asocian a dos posibles clases, y otros de **clasificación multiclase**, con más de dos categorías posibles.

Se observó, además, una marcada predominancia de conjuntos de datos provenientes del ámbito **médico**, dentro de los cuales se seleccionaron aquellos considerados más adecuados para las pruebas de los métodos de aprendizaje automático, abarcando tanto casos binarios como multiclase.

Todos estos *datasets* fueron procesados anteriormente a las pruebas realizadas, en donde se eliminaron registros duplicados o con datos faltantes y también no se tuvieron en cuenta aquellos con datos atípicos a *priori*, como un caso donde un paciente tiene colesterol 0.

1.3.1 Dataset Binario: Insuficiencia Cardíaca Predicción

Las enfermedades cardiovasculares son la causa número uno de muerte globalmente, con un estimado de 17.9 millones de vidas cada año, aproximadamente 31% de todas las muertes globales. La idea central de este trabajo es encontrar la técnica de aprendizaje automático mas óptima para poder realizar predicciones de si un paciente tiene altas probabilidades de tener insuficiencia cardíaca.

Este dataset [5], llamado *HeartFailure* fue creado mediante la combinación de cinco datasets independientes en 11 atributos comunes, logrando el dataset más grande de información de enfermedades cardiovasculares utilizado para investigación. Los cinco datasets utilizados son:

- Cleveland: 303 observaciones
- Hungarian: 294 observaciones
- Switzerland: 123 observaciones
- Long Beach VA: 200 observaciones
- Stalog (Heart) Data Set: 270 observaciones

La cantidad de registros y los tipos de atributos utilizados para este trabajo fueron los siguientes, considerando que estos registros ya fueron previamente procesados para poder ser utilizados en las técnicas de aprendizaje automático. Se informara la distribución de los valores si el atributo es categórico, en caso de ser numérico, se informara la media de los valores, junto al valor máximo y mínimo que poseen.

En la Tabla 1.1 se muestran que tipo de datos son los atributos del dataset que serán utilizados en este trabajo.
Cantidad de registros: 918
Cantidad de atributos: 11
Atributos Categóricos: 5
Atributos Numéricos: 6

Descripción atributos:

A continuación se realizara una pequeña descripción de cada atributo anteriormente señalado. Si el atributo es categórico, se informara la distribución de los valores que posee dicho atributo y en caso de resultar ser un atributo numérico, se informara la media de los valores de dicho atributo, junto al valor máximo y mínimo que posee. Se realiza lo anterior para contextualizar los atributos y ver los rangos de valores con que se trabajara.

Age: Este atributo refiera a la edad de los pacientes. Tiene media: 53 años, valor máximo: 77 y valor mínimo: 28, con proporciones de edad bastante bien distribuidas, siendo la menor de 0.11% para algunas

Tabla 1.1: Tipo de atributo del conjunto Binario.

Atributo	**Tipo de dato**	**¿Esta codificado?**	**Unidad**
Age	Numérico (int)	No	Años
Sex	Categorico (string)	No	-
ChestPainType	Categorico (string)	No	-
RestingBP	Numérico (int)	No	mm Hg
Cholesterol	Numérico (int)	No	mm/dl
FastingBS	Numérico (int)	Si	mg/dl
RestingECG	Categorico (string)	No	-
MaxHR	Numérico (int)	No	-
ExerciseAngina	Categorico (string)	No	-
Oldpeak	Numérico (float)	No	ST en depresión
ST_Slope	Categorico (string)	No	-
HeartDisease	Numérico (int)	Si	-

edades y la mayor de 4.14% para otras edades, teniendo otras distribuciones entre estos dos rangos

Sex: Refiere al Sexo de los pacientes; hay una distribución 78.98% M (masculinos) y hay 21.02% F (femeninos)

ChestPainType: Tipo del dolor en el pecho, del caul hay varias clasificaciones; Tiene una distribución 18.85% ATA, hay 22.11% NAP, hay 54.03% ASY, hay 5.01% TA. [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]

RestingBP: Se esta describiendo la Presión sanguínea en reposo, donde hay una distribución de 51.09% de mujeres, codificadas en 1 y 48.91% de hombres, codificados en 0.

Cholesterol: Este atributo es el Colesterol serico, la medida total de colesterol en sangre; tiene media: 199.02, valor máximo: 603.00 y valor mínimo: 0.00. Miligramos por decilitro

FastingBS: Es la Glucosa en sangre en ayuno; hay 76.66% Glucosa en sangre $<$ 120 mg/dl codificado en 0 y hay 23.34% Glucosa en sangre $>$ 120 mg/dl codificado en 1

RestingECG: Son los Resultados de electrocardiogramas en reposo; hay 60.09% codificado en Normal, hay 19.41% codificado en ST y hay 20.50% codificado en LVH [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of $>$ 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]

MaxHR: Este atributo es el Máximo ritmo cardíaco registrado, tiene media: 136.79, valor máximo: 202.00 y valor mínimo: 60.00

ExerciseAngina: Es la Angina producido por ejercicio, dolor en el pecho; hay 59.54% No codificado en N y hay 40.46% Si codificado en Y

Oldpeak: Valor máximo de depresión del segmento ST (en milímetros) registrado en todas las derivaciones contiguas durante una prueba de esfuerzo. Forma parte del cálculo del riesgo de un paciente de isquemia o infarto de miocardio; valores más altos indican un mayor riesgo de enfermedad coronaria; tiene media:

0.90, valor máximo: 6.20 y valor mínimo: -0.10

ST_Slope: The slope of the peak exercise ST segment; hay 43.08% Up, hay 50.05% Flat y hay 6.87% Down [Up: upsloping, Flat: flat, Down: downsloping]

HeartDisease: Variable de salida de si posee una enfermedad cardíaca; hay 44.71% No codificado en 0 y hay 55.29% Si codificado en 1. Siendo esta la **variables objetivo**.

1.3.2 Dataset Multiclase: Cardiotocografía Predicción

Descripción: La cardiotocografía (CTG) es un registro continuo de la frecuencia cardíaca fetal que se obtiene mediante un transductor de ultrasonidos colocado en el abdomen materno. La CTG se utiliza ampliamente durante el embarazo como método para evaluar el bienestar fetal, sobre todo en embarazos con mayor riesgo de complicaciones.

En el dataset [2] utilizado se procesaron automáticamente 2126 cardiotocogramas fetales (CTG) y se midieron sus características diagnósticas. Tres obstetras expertos clasificaron los CTG y se les asignó una etiqueta de clasificación consensuada. La clasificación se realizó tanto con respecto a un patrón morfológico (A, B, C...) como al estado fetal (N, S, P).

En la Tabla 1.2 se muestran los tipos de las variables utilizadas en este trabajo. Luego la cantidad de registros y los tipos de atributos utilizados para este trabajo fueron los siguientes, considerando que estos registros ya fueron previamente procesados para poder ser utilizados en las técnicas de aprendizaje automático.
Cantidad de registros: 2115
Cantidad de atributos: 21
Atributos Categóricos: 0
Atributos Numéricos: 21

Descripción atributos:

A continuación se realizara una pequeña descripción de cada atributo anteriormente señalado. Si el atributo es categórico, se informara la distribución de los valores que posee dicho atributo y en caso de resultar ser un atributo numérico, se informara la media de los valores de dicho atributo, junto al valor máximo y mínimo que posee. Se realiza lo anterior para contextualizar los atributos y ver los rangos de valores con que se trabajará.

LB:Frecuencia cardíaca fetal basal (latidos por minuto). Tiene media: 133.30, valor máximo: 160.00 y valor mínimo: 106.00

AC: Número de aceleraciones por segundo. Tiene media: 0.00, valor máximo: 0.02 y valor mínimo: 0.00

FM:Número de movimientos fetales por segundo. Tiene media: 0.01, valor máximo: 0.48 y valor mínimo: 0.00

UC: Número de contracciones uterinas por segundo. Tiene media: 0.00, valor máximo: 0.01 y valor mínimo: 0.00

Tabla 1.2: Tipo de atributo del conjunto Multiclae.

Atributo	**Tipo de dato**
LB	Numérico (int)
AC	Numérico
FM	Numérico (float)
UC	Numérico (float)
DL	Numérico (float)
DS	Numérico (float)
DP	Numérico (float)
ASTV	Numérico (int)
MSTV	Numérico (float)
ALTV	Numérico (int)
MLTV	Numérico (float)
Width	Numérico (int)
Min	Numérico (int)
Max	Numérico (int)
Nmax	Numérico (int)
Nzeros	Numérico (int)
Mode	Numérico (int)
Mean	Numérico (int)
Median	Numérico (int)
Variance	Numérico (int)
Tendency	Numérico (int)
NSP	Categórico (string)

DL:Número de desaceleraciones leves por segundo. Tiene media: 0.00, valor máximo: 0.01 y valor mínimo: 0.00

DS: Número de desaceleraciones severas por segundo. Hay un 99.67% con valor 0.0 y un 0.33% con un valor 0.001

DP: Número de desaceleraciones prolongadas por segundo. Hay un 91.58% con valor 0.0, 3.40% con un valor 0.002, 1.13% con un valor 0.003, 3.31% con un valor 0.001, 0.43% con un valor 0.004 y 0.14% con un valor 0.005

ASTV: Porcentaje de tiempo con variabilidad anormal a corto plazo. Tiene media: 46.98, valor máximo: 87.00 y valor mínimo: 12.00

MSTV: Valor medio de la variabilidad a corto plazo. Tiene media: 1.34, valor máximo: 7.00 y valor mínimo: 0.20

ALTV: Porcentaje de tiempo con variabilidad anormal a largo plazo. Tiene media: 9.79, valor máximo: 91.00 y valor mínimo: 0.00

MLTV: Valor medio de la variabilidad a largo plazo. Tiene media: 8.17, valor máximo: 50.70 y valor mínimo: 0.00

Width: Ancho del histograma de FCF. Tiene media: 70.51, valor máximo: 180.00 y valor mínimo: 3.00

Min: Mínimo del histograma de FCF. Tiene media: 93.57, valor máximo: 159.00 y valor mínimo: 50.00

Max: Máximo del histograma de FCF. Tiene media: 164.09, valor máximo: 238.00 y valor mínimo: 122.00

Nmax: Número de picos del histograma. Tiene media: 4.08, valor máximo: 18.00 y valor mínimo: 0.00

Nzeros: Número de ceros del histograma. Hay un 76.26% con valor 0, 17.30% con un valor 1, 0.99% con un valor 3, 5.11% con un valor 2, 0.09% con un valor 4, 0.05% con un valor 10, 0.09% con un valor 5, 0.05% con un valor 8, y 0.05% con un valor 7.

Mode: Moda del histograma. Tiene media: 137.45, valor máximo: 187.00 y valor mínimo: 60.00

Mean: Promedio del histograma. Tiene media: 134.60, valor máximo: 182.00 y valor mínimo: 73.00

Median: Media del histograma. Tiene media: 138.08, valor máximo: 186.00 y valor mínimo: 77.00

Variance: Varianza del histograma. Tiene media: 18.89, valor máximo: 269.00 y valor mínimo: 0.00

Tendency: Tendencia del histograma. Hay un 39.67% con valor 1, 52.53% con un valor 0 y 8.27% con un valor -1

CLASS: Código de clasificación del estado fetal (N=normal; S=sospechoso; P=patológico). Hay un 13.81% con valor Sospechoso, 77.92% con un valor Normal, 8.27% con un valor Patológico. Siendo esta la **variables objetivo**.

Chapter 2

Métricas de Rendimiento Utilizadas

2.1 Introducción

El objetivo de este trabajo es evaluar el desempeño calificador de cada modelo de Aprendizaje Automático. Para alcanzarlo, se utilizaran **métricas de rendimiento** que permiten cuantificar la capacidad del algoritmo de clasificar, ergo son herramienta que utilizamos para saber que tan bien predice un modelo a una clase o que tan bien puede clasificar entre varias clases.

La importancia de las métricas se ubica en que el objetivo central de estos algoritmos no es simplemente obtener un buen rendimiento en los datos utilizados para construir el modelo, sino en su **capacidad de generalización**, su habilidad para funcionar correctamente con entradas nuevas y previamente no observadas (no utilizadas en el entrenamiento). Esto se debe a que es perfectamente normal y sumamente esperable que el modelo funcione correctamente con el conjunto de datos que se utiliza para el entrenamiento del modelo, la idea fundamental es poder tener el mismo rendimiento o incluso uno mejor que con el conjunto de entrenamiento.

Para la obtención de las métricas y entrenamiento de algoritmo se utilizara la estrategia de **Validación Cruzada k-fold**, donde el conjunto de datos se divide en k grupos (o pliegues, en una traducción más fiel) del mismo tamaño, donde en cada iteración un grupo k es utilizado para entrenar y el resto para evaluar, repitiéndose el proceso k veces. Es importante señalar que un grupo k_i es utilizado solo una vez para entrenar, el resto de veces será utilizado como parte del conjunto de prueba. El valor final estimado de la métrica, denotado por $\widehat{M}$, es el promedio de los valores obtenidos de cada grupo, es decir,

$$\hat{M} = \frac{1}{k} \sum_{i=1}^{k} M_i, \tag{2.1}$$

donde M_i es el valor de la métrica de evaluación obtenido en el i-ésimo grupo utilizado como conjunto de prueba, para $i = 1, 2, \ldots, k$.

Dentro de este trabajo no sólo se evaluaran distintos modelos, sino que se utilizaran distintos *datasets* para lograrlos. A continuación se señalaran las métricas que se utilizaran en cada caso y se podran apreciar algunas diferencias, leves, pero diferencias en si.

2.2 Métricas para caso Binario

2.2.1 Matriz de Confusión

Una matriz de confusión, que se puede observar en la Tabla 2.1, es una forma simple de saber de que forma esta clasificando el algoritmo, donde una clase es considerada **positiva** P y la otra **negativa** N. La matriz de confusión clasifica las predicciones en:

- **Verdaderos Positivos (TP):** Casos positivos clasificados correctamente.
- **Verdaderos Negativos (TN):** Casos negativos clasificados correctamente.
- **Falsos Positivos (FP):** Casos negativos clasificados incorrectamente como positivos.
- **Falsos Negativos (FN):** Casos positivos clasificados incorrectamente como positivos.

		Predicción	
		Positivo	**Negativo**
Verdad	**Positivo**	**Verdadero Positivo (TP)**	**Falso Negativo (FN)**
	Negativo	**Falso Positivo (FP)**	**Verdadero Negativo (TN)**

Tabla 2.1: Matriz de Confusion

2.2.2 *Accuracy*

El *Accuracy* es la proporción de instancias clasificadas correctamente, es una medida "ingenua" que puede ser engañosa si existe un gran desbalance entre clases, ergo se puede obtener un *Accuracy* alto si predice una clase muy bien, que tiene una distribución mucho mayor que la otra, mientras que la de menor distribución casi no la predice. En términos de la Matriz de Confusión la formula seria la siguiente:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{\text{Total}}, \tag{2.2}$$

y en términos del conjunto de predicciones y valores verdaderos, se tiene que n_{samples}: representa la cantidad total de ejemplos en la muestra, mientras que $\hat{y}_i$ es el valor predicho del i-ésimo ejemplo, e y_i es el valor verdadero correspondiente:

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i), \tag{2.3}$$

por lo tanto, se puede simplificar como la siguiente formula:

$$\text{Accuracy} = \frac{\text{Número de predicciones correctas}}{\text{Número total de muestras}}. \tag{2.4}$$

2.2.3 *Precision*

El *Precision* mide la probabilidad de que la predicción positiva del clasificador sea correcta, en otras palabras, mide que tan bien predice las clases positivas el modelo. En términos de la Matriz de Confusión, se puede expresar lo anterior de la siguiente manera;

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{2.5}$$

2.2.4 *Recall*

El *Recall* o también conocido como Sensibilidad o Tasa de Verdaderos Negativos (TPR). Mide la probabilidad de que el clasificador detecte un caso positivo cuando en verdad lo es. En términos de la Matriz de Confusión se puede entender a la *Recall* de la siguiente manera:

$$\text{Recall} = TPR = \frac{TP}{TP + FN} = \frac{TP}{P}. \tag{2.6}$$

2.2.5 *F-measure*

El *F-measure* es la media armónica ponderada de *precision* y *recall*. La versión más común es el **F1-score**, donde el parámetro de ponderación β es igual a 1. Un clasificador perfecto tiene un valor $F1 = 1$. Fórmula General (F_β):

$$F_\beta = \frac{(1 + \beta^2)\text{precision} \times \text{recall}}{\beta^2\text{precision} + \text{recall}}, \tag{2.7}$$

donde la fórmula del F1-score ($\beta = 1$) en términos de Precision y Recall se puede notar de la siguiente manera:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \tag{2.8}$$

o en términos de la Matriz de Confusión, de forma más simplificada:

$$F1 = \frac{2TP}{2TP + FP + FN}. \tag{2.9}$$

2.2.6 *Área Bajo la Curva ROC (ROC AUC)*

La métrica *ROC AUC* es un valor que resume la capacidad de un clasificador para distinguir entre clases, una métrica muy útil para comparar el desempeño entre modelos distintos o entre un mismo modelo con hipermetamorfosis distintos.

La Curva ROC es un gráfico que ilustra el rendimiento de un clasificador binario a media que se varia su umbral de discriminación. Se crea graficando la **Tasa de Verdaderos Positivos (TPR)** versus la **Tasa de Falsos Positivos (FPR)** en varios umbrales. El **AUC** mide justamente el área debajo de la Curva ROC.

Ejes utlizados para el gráfico:

- **Eje Y:** TPR
- **Eje X:** FPR

Interpretación de valores: Un clasificador **ideal** se ubica en el punto $(0, 1)$, donde $TPR = 1$ y $FPR = 0$, lo que resulta en un $AUC = 1$. Un clasificador **aleatorio** se sitúa sobre la línea $TPR = FPR$, lo que resulta en un $AUC = 0.5$. Un clasificador se considera **razonable** si $0.5 < AUC \leq 1$

2.3 Métricas para caso Multiclase

En este caso se utiliza el método *"weighted"*, el cual computa o tiene en cuenta el desequilibrio de clases calculando el promedio de métricas binarias, en las que la puntuación o peso de cada clase se pondera según su presencia en la muestra de datos reales.

La métrica ponderada por la presencia de la clase, $\mathrm{M}_{\text{weighted}}$, se calcula como el promedio de la métrica por clase M_l, donde cada contribución es ponderada por el tamaño de la clase $|y_l|$, siendo L es el conjunto de etiquetas o clases. Donde $\hat{M}_{\text{weighted}}$ es el valor estimado de la métrica promedio ponderada.

$$\hat{M}_{\text{weighted}} = \frac{1}{\sum l \in L|y_l|} \sum_{l \in L} |y_l| \cdot \mathrm{M}_l. \tag{2.10}$$

2.3.1 Matriz de Confusión (Multiclase)

La matriz de confusión multiclase, que se puede ver en la Tabla 2.2, es una matriz cuadrada de tamaño $L \times L$, donde L es el número de clases. Cada celda C_{ij} representa la cantidad de muestras verdaderamente pertenecientes a la clase i que fueron clasificadas como clase j.

Para cada clase l se definen los valores que antes habíamos utilizados para la matriz de confusión del caso binario:

- $TP_l = C_{ll}$
- $FP_l = \sum_{i \neq l} C_{il}$
- $FN_l = \sum_{j \neq l} C_{lj}$
- $TN_l = N - TP_l - FP_l - FN_l$

		Predicción			
		Clase C_1	**Clase** C_2	$\cdots$	**Clase** C_l
Verdad	**Clase** C_1	TN_l	$\cdots$	TN_l	FP_l
	Clase C_2	TN_l	TN_l	$\cdots$	FP_l
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	Clase C_l	FN_l	$\cdots$	FN_l	TP_l

Tabla 2.2: Matriz Confusion Multiclase

2.3.2 *Precision*

La *Precision* por clase l mide la proporción de muestras clasificadas como positivas que realmente pertenecen a la clase l.
En términos más simple, utilizando la Matriz de Confusión:

$$\text{Precision}_l = \frac{TP_l}{TP_l + FP_l}. \tag{2.11}$$

2.3.3 *Recall*

El *Recall* por clase l mide la proporción de muestras verdaderamente positivas de la clase l que fueron correctamente identificadas.
En términos más simple, utilizando la Matriz de Confusión:

$$\text{Recall}_l = \frac{TP_l}{TP_l + FN_l}. \tag{2.12}$$

2.3.4 *F-measure*

El *F-measure* es la media armónica ponderada de *precision* y *recall*. La versión más común es el **F1-score**, donde el parámetro de ponderación β es igual a 1. Un clasificador perfecto tiene un valor $F1 = 1$.
El valor global ponderado se obtiene aplicando la fórmula de M_{weighted} sobre los $F_{1,l}$:

$$F_{1,\text{weighted}} = \sum_{l \in L} w_l \, F_{1,l}, \quad \text{con } w_l = \frac{n_l}{\sum_{i \in L} n_i}. \tag{2.13}$$

2.3.5 *Área Bajo la Curva ROC (ROC AUC)*

Para extender la métrica ROC AUC a clasificación multiclase se emplea el enfoque **One-vs-Rest (OVR)**. Para cada clase l, se considera la clase l como positiva y el resto como negativas, luego se calcula el AUC correspondiente (AUC_l) sobre la curva ROC de esa clasificación binaria. Finalmente, se obtiene un promedio ponderado por el soporte de cada clase:

$$\text{AUC}_{\text{OVR, weighted}} = \sum_{l \in L} w_l \, \text{AUC}_l. \tag{2.14}$$

2.4 Importancia de la característica

Sea un modelo predictivo $Modelo$ entrenado sobre un conjunto de datos tabular X (ya sea de entrenamiento o validación), y sea M la métrica de referencia del modelo sobre los datos originales. El procedimiento se detalla a continuación:

1. Calcular el puntaje de referencia del modelo M sobre X:

$$\text{M} = \text{calcular métrica}(Modelo, X). \tag{2.15}$$

2. Para cada atributo j del conjunto de datos:

   (a) Repetir el siguiente proceso K veces (para reducir la varianza de la estimación):

      i. Generar una versión alterada del conjunto de datos, $X^{(k,j)}$, en la que se permuta aleatoriamente, se intercambian de orden los datos de la columna correspondiente a la característica j, manteniendo las demás columnas sin cambios, para ver si el modelo empeora o no en su rendimiento.

      ii. Calcular el puntaje del modelo sobre los datos permutados:

$$M_{k,j} = \text{calcular métrica}(Modelo, X^{(k,j)}). \tag{2.16}$$

   (b) Calcular la importancia de la característica j como la disminución promedio en el puntaje del modelo respecto del puntaje de referencia:

$$I_j = M - \frac{1}{K} \sum_{k=1}^{K} M_{k,j}. \tag{2.17}$$

De esta manera, I_j mide la pérdida de desempeño al romper la relación entre la característica j y la variable objetivo. Valores más altos de I_j indican características más relevantes para el modelo.

Chapter 3

Descripción de los Métodos Utilizados

3.1 Regresión Logística

El modelo de **Regresión Logística** [4] (LR, por su equivalente en inglés *Logistic Regression*) es una técnica del análisis de datos utilizada para establecer relaciones entre las variables predictoras y la clase a la cual pertenece cada registro. Posteriormente, el modelo permite predecir la probabilidad de que un nuevo registro pertenezca a una clase determinada. Este tipo de modelo de regresión es justamente utilizado para los problemas no linealmente separables, como la mayoría de problemas.

A diferencia de la regresión lineal múltiple, la regresión logística predice una probabilidad (valor entre 0 y 1). Ambos modelos son lineales en sus parámetros, pero difieren en la naturaleza de la variable dependiente. El objetivo es estimar los coeficientes de regresión que maximizan la verosimilitud de los datos observados, o encontrar los coeficientes que mejor funcionan para los datos de entrenamiento.

El modelo busca modelar la probabilidad condicional de que una observación pertenezca a la clase objetivo y_i, siendo

$$P(y_i = 1 \mid \boldsymbol{x}_i), \tag{3.1}$$

donde $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$ es el vector de características de la observación i, el cual también llamamos registro.

3.1.1 Función de Probabilidad

La función logística define la probabilidad de pertenencia de x_i a la clase 1 como:

$$p(\boldsymbol{x}_i) = P(y_i = 1 \mid \boldsymbol{x}_i) = \frac{\exp(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})}{1 + \exp(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})}, \tag{3.2}$$

donde β_0 es el intercepto y β_j los coeficientes asociados a cada predictor.

3.1.2 Función Logit

La función inversa de la función logística, denominada *logit*, relaciona el logaritmo de las *odds* con un modelo lineal, siendo *odds* lo que se utiliza para analizar si la probabilidad de ocurrencia de un evento -caso/no caso- difiere o no en distintos grupos,

$$\text{logit}(p(\boldsymbol{x}_i)) = \ln\left(\frac{p(\boldsymbol{x}_i)}{1-p(\boldsymbol{x}_i)}\right) = \beta_0 + \sum_{j=1}^{k}\beta_j x_{ij}. \tag{3.3}$$

Esta transformación asegura que las probabilidades estén acotadas entre 0 y 1, mientras que la combinación lineal de predictores puede tomar cualquier valor real, siendo este ultimo punto un factor que realiza el calculo de coeficientes muy difícil.

3.1.3 Estimación por Máxima Verosimilitud

Los coeficientes de regresión se estiman mediante el método de **Máxima Verosimilitud** (MLE), donde la función de log-verosimilitud a maximizar

$$\ell(\beta_0, \beta_1, \ldots, \beta_k) = \sum_{i=1}^{n}\left[y_i \ln(p(\boldsymbol{x}_i)) + (1-y_i)\ln(1-p(\boldsymbol{x}_i))\right], \tag{3.4}$$

en donde la solución analítica no existe, por lo que se utilizan métodos numéricos iterativos para obtener los parámetros óptimos.

3.1.4 Hiperparámetros

- **Parámetro de Regularización (C):** Controla la complejidad del modelo. Valores pequeños de C implican mayor regularización (menor sobreajuste), mientras que valores grandes permiten mayor flexibilidad del modelo.
- **Penalización (*penalty*):** Es un término regulador (Ω) que se suma a la función de coste original (J) para formar una función ajustada $\tilde{J}$. Controla la capacidad del modelo y reduce el error de generalización.
  - Regularización L1 (Lasso).
  $$\tilde{J}_{L1}(\boldsymbol{\beta}) = J(\boldsymbol{\beta}) + \alpha\sum_{j=1}^{k}|\beta_j|. \tag{3.5}$$
  - Regularización L2 (Ridge).
  $$\tilde{J}_{L2}(\boldsymbol{\beta}) = J(\boldsymbol{\beta}) + \frac{1}{2}\alpha\sum_{j=1}^{k}\beta_j^2. \tag{3.6}$$
  - Regularización Elastic Net
  $$\tilde{J}_{EN}(\boldsymbol{\beta}) = J(\boldsymbol{\beta}) + \alpha\left[\rho\sum_{j=1}^{k}|\beta_j| + \frac{1-\rho}{2}\sum_{j=1}^{k}\beta_j^2\right]. \tag{3.7}$$
- **Algoritmo de Optimización (*solver*):** El *solver* es el algoritmo numérico encargado de minimizar la función de coste regularizada $\tilde{J}(\boldsymbol{\beta})$.
  - Método Newton-CG (basado en segunda derivada).
  $$\boldsymbol{\beta}^{(t+1)} \leftarrow \boldsymbol{\beta}^{(t)} - \boldsymbol{H}^{-1}\nabla_{\boldsymbol{\beta}}\tilde{J}(\boldsymbol{\beta}^{(t)}). \tag{3.8}$$
  - Método BFGS (quasi-Newton).
  $$\boldsymbol{\beta}^{(t+1)} \leftarrow \boldsymbol{\beta}^{(t)} - \boldsymbol{B}^{-1}\nabla_{\boldsymbol{\beta}}\tilde{J}(\boldsymbol{\beta}^{(t)}). \tag{3.9}$$
- **Estrategia Multiclase (*multi_class*):** La regresión logística está diseñada originalmente para clasificación binaria. Para extenderla a múltiples clases se emplean estrategias como:
  - *one-vs-rest* (OvR): Entrena un clasificador por clase.
  - *multinomial*: Optimiza una única función de verosimilitud multinomial conjunta.

3.2 Árboles de Decisión

El aprendizaje mediante **Árboles de Decisión** (RF, por su equivalente en inglés *Random Forest*) es un método no paramétrico que utiliza divisiones jerárquicas sobre los atributos de los datos, construyendo reglas de decisión del tipo *if-else* para predecir el valor de una variable objetivo.

El objetivo principal es encontrar las divisiones (particiones) que maximicen la pureza de los nodos hijos, es decir, que minimicen la impureza del nodo resultante. Es un método mucho mas sencillo de realizar, donde se crea un camino de decisión, por lo cual según los valores de los atributos de un registro podemos predecir que a que clase pertenecerían, donde el computo pesado se encuentra en la creación del propio camino.

3.2.1 Conceptos Fundamentales

La probabilidad de que un ejemplo en el nodo t pertenezca a la clase C_k se define como

$$p(k|t) = \frac{N_k(t)}{N(t)}, \tag{3.10}$$

donde $N(t)$ es la cantidad total de ejemplos en el nodo t, y $N_k(t)$ la cantidad de ejemplos de la clase C_k. Es importante este fenómeno porque es un costo computacional muy barato el calcular esta probabilidad y se asemejan a las probabilidades equiprobables, donde la probabilidad de una clase en un nodo esta dada por la cantidad de clases que tienen en el mismo.

Impureza del Nodo

La impureza de un nodo t se mide mediante una función ϕ que depende de las probabilidades de clase en dicho nodo, donde la función de impureza i es un hiperparametro en sí mismo.

$$i(t) = \phi\big(p(1|t), p(2|t), \ldots, p(K|t)\big). \tag{3.11}$$

La impureza es máxima cuando las clases están perfectamente mezcladas y mínima (cero) cuando el nodo contiene solo una clase, en donde la impureza máxima seria que la probabilidad de cada clase sea aleatoria, ya que están perfectamente mezcladas.

Entropía de Shannon de un conjunto de datos D

$$H(D) = -\sum_{k=1}^{K} \frac{N_k(D)}{N(D)} \log_2 \left(\frac{N_k(D)}{N(D)} \right). \tag{3.12}$$

Índice de Gini de un nodo t

$$\text{Gini}(t) = 1 - \sum_{k=1}^{K} \big[p(k|t)\big]^2 = 1 - \sum_{k=1}^{K} \left(\frac{N_k(t)}{N(t)} \right)^2. \tag{3.13}$$

Disminución de Impureza

La reducción de impureza generada al dividir el nodo t en dos nodos hijos t_1 y t_2 mediante una partición s se calcula como

$$\Delta i(s,t) = i(t) - q_1 i(t_1) - q_2 i(t_2), \tag{3.14}$$

donde $q_j = \frac{N(t_j)}{N(t)}$ para $j = 1, 2$.

3.2.2 Bosques Aleatorios (Random Forest)

El algoritmo de **Random Forest** [1] (RF) combina múltiples árboles de decisión independientes construidos sobre subconjuntos aleatorios de los datos (muestreo con reemplazo o *bootstrap*). Cada árbol se entrena sobre un subconjunto de atributos aleatorios en cada división, lo que introduce diversidad y reduce la varianza.
La predicción final para clasificación se obtiene mediante el voto mayoritario de los árboles

$$\hat{y} = \underset{c \in \mathcal{C}}{\operatorname{argmax}} \sum_{m=1}^{M} \mathbb{I}\big(h_m(\boldsymbol{x}) = c\big), \tag{3.15}$$

donde $h_m(\boldsymbol{x})$ es la predicción del árbol m.

3.2.3 Hiperparámetros

- **Criterio de Partición:** Función de impureza utilizada (e.g., Índice de Gini o Entropía de Shannon).
- **Algoritmo de Construcción:** *ID3* emplea la ganancia de información (entropía), mientras que *CART* utiliza el índice de Gini y genera árboles binarios.
- **Número de Atributos Muestreados (RF):** En Random Forest, típicamente se seleccionan $\sqrt{a}$ o $\ln(a)$ atributos por partición, donde a es la cantidad total de atributos.
- **Número de Árboles (RF):** Cantidad de árboles a construir en el bosque.

3.3 Clasificador Naïve Bayes

El **Clasificador Naïve Bayes** [6] (CNB) es un método supervisado probabilístico basado en el *Teorema de Bayes*, que asume independencia condicional entre los atributos dado la clase, lo cual sabemos que no es posible que estas mismas variables sean independientes entre sí. La regla de clasificación, o la probabilidad de que se de una clase dado un registro,

$$P(Y = C_k \mid X_1 = x_1, \ldots, X_d = x_d) = \frac{P(X_1 = x_1, \ldots, X_d = x_d \mid Y = C_k)\, P(Y = C_k)}{P(X_1 = x_1, \ldots, X_d = x_d)}. \tag{3.16}$$

Probabilidad Condicional: Bajo el supuesto de independencia condicional, la probabilidad condicional puede expresarse como

$$P(X_1 = x_1, \ldots, X_d = x_d \mid Y = C_k) = \prod_{j=1}^{d} P(X_j = x_j \mid Y = C_k), \tag{3.17}$$

en donde puede ser expresado para una clase k cuando se presenta un registro j

$$\hat{\theta}_{jmk} = P(X_j = x_{jm} \mid Y = C_k) = \frac{\#\{X_j = x_{jm} \wedge Y = C_k\}}{\#\{Y = C_k\}}. \tag{3.18}$$

Probabilidad a Priori: Otra estimación fundamental a calcular y expresar es la a priori, en la cual se asemeja a la probabilidad de que un nodo pertenezca a una clase k en *Random Forest*

$$\hat{\pi}_k = P(Y = C_k) = \frac{\#\{Y = C_k\}}{N}. \tag{3.19}$$

3.3.1 Caso Continuo (Naïve Bayes Gaussiano)

Cuando los atributos son continuos y se asume distribución normal, las verosimilitudes se estiman con la función de densidad Gaussiana:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right], \tag{3.20}$$

por lo cual la decisión final se obtiene como

$$\hat{Y} = \underset{C_k}{\operatorname{argmax}} \left[\log P(Y = C_k) + \sum_{j=1}^{d} \log f(x_j \mid \mu_{jk}, \sigma_{jk}^2).\right] \tag{3.21}$$

3.4 Máquinas de Soporte Vectorial (SVM)

Las **Máquinas de Soporte Vectorial** [3] (SVM, por sus siglas en inglés) constituyen una técnica de clasificación supervisada basada en la búsqueda de una **función de decisión** que permita predecir la clase de una observación a partir de sus atributos. Dado que este método opera sobre variables numéricas, los atributos categóricos deben codificarse previamente.

El objetivo fundamental de una SVM es encontrar un **hiperplano de separación óptimo** que divida los datos en función de sus clases, **maximizando el margen** M, es decir, la distancia mínima entre el hiperplano y los puntos más cercanos de cada clase (denominados **vectores de soporte**). Dichos vectores determinan la posición y orientación del hiperplano, por lo que son los únicos puntos relevantes en el entrenamiento del modelo.

Aunque existen infinitos hiperplanos que pueden separar los datos, el principio de las SVM consiste en seleccionar aquel que logre la **máxima separación posible entre las clases**, minimizando simultáneamente el riesgo de sobreajuste y aumentando la capacidad de generalización.

3.4.1 Margen Rígido (Hard Margin)

El caso de **margen rígido** supone que los datos son linealmente separables, es decir, existe un hiperplano que separa perfectamente las clases sin errores de clasificación. En este caso, el problema de optimización se formula como:

$$\text{Minimizar } \frac{1}{2}|\boldsymbol{w}|^2 \quad \text{sujeto a } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + \beta) \geq 1, \tag{3.22}$$

donde $\boldsymbol{w}$ es el vector normal al hiperplano, β es el término de sesgo, y la restricción garantiza que todas las observaciones queden correctamente clasificadas, y a una distancia mínima de $\frac{1}{\|\boldsymbol{w}\|}$ del hiperplano.

3.4.2 Margen Suave (Soft Margin)

En la práctica, los datos rara vez son perfectamente separables. Por ello, se introduce el concepto de **margen suave**, que permite violaciones controladas de las restricciones mediante variables de holgura $\theta_i \geq 0$. El problema se redefine como:

$$\text{Minimizar } \frac{1}{2}|\boldsymbol{w}|^2 + C\sum_{i=1}^{n} \xi_i \quad \text{sujeto a } \Big\{ y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + \beta) \geq 1 - \xi_i \xi_i \geq 0, \tag{3.23}$$

donde el parámetro $C > 0$ actúa como un **control de regularización**: valores grandes de C penalizan más fuertemente los errores, buscando una separación más estricta (a costa de menor margen), mientras que valores pequeños permiten más errores, favoreciendo márgenes amplios y mayor generalización.

3.4.3 Formulación Dual y Kernel Trick

La formulación dual del problema permite expresar la solución en términos de los productos internos entre las observaciones:

$$\text{Maximizar } -\frac{1}{2}\sum_{i,\ell=1}^{n}\alpha_i\alpha_\ell y_i y_\ell K(\boldsymbol{x}i,\boldsymbol{x}\ell)+\sum_{i=1}^{n}\alpha_i, \tag{3.24}$$

$$\text{sujeto a } 0\le\alpha_i\le C,\quad \sum_{i=1}^{n}\alpha_i y_i=0. \tag{3.25}$$

Aquí, los α_i son los multiplicadores de Lagrange asociados a las restricciones, y $K(\boldsymbol{x}_i,\boldsymbol{x}_\ell)$ representa el producto interno entre las observaciones en un espacio transformado.

En muchos casos, las clases no son separables linealmente en el espacio original de los datos. El **Kernel Trick** permite proyectar los datos a un espacio de mayor dimensión (posiblemente infinito) donde sí exista un hiperplano separador, *sin necesidad de calcular explícitamente la transformación.*
Esto se logra sustituyendo los productos internos $\langle\boldsymbol{x}_i,\boldsymbol{x}_\ell\rangle$ por una función **kernel** $K(\boldsymbol{x}_i,\boldsymbol{x}_\ell)$, que computa directamente el producto interno en el espacio transformado. De esta forma, se mantiene la eficiencia computacional mientras se obtiene un modelo capaz de representar fronteras de decisión no lineales.

Funciones Kernel Comunes

Kernel Lineal:

$$K(\boldsymbol{a},\boldsymbol{b})=\langle\boldsymbol{a},\boldsymbol{b}\rangle. \tag{3.26}$$

Kernel Radial (RBF o Gaussiano):

$$K(\boldsymbol{a},\boldsymbol{b})=\exp\left(-\gamma\|\boldsymbol{a}-\boldsymbol{b}\|^2\right). \tag{3.27}$$

Kernel Polinómico:

$$K(\boldsymbol{a},\boldsymbol{b})=(\langle\boldsymbol{a},\boldsymbol{b}\rangle+r)^d. \tag{3.28}$$

Kernel Sigmoide:

$$K(\boldsymbol{a},\boldsymbol{b})=\tanh(\gamma\langle\boldsymbol{a},\boldsymbol{b}\rangle+r). \tag{3.29}$$

3.4.4 Hiperparámetros

- **Parámetro de Regularización (C):** Controla el equilibrio entre la maximización del margen y la penalización por errores de clasificación.
- **Tipo de Kernel:** Define la forma de la frontera de decisión (Lineal, Polinómico, Radial o Sigmoideo).
- **Parámetros del Kernel:** Por ejemplo, γ o σ^2 en el kernel RBF; grado d y coeficiente r en el kernel polinómico.

Chapter 4

Resultados

En este capítulo se presentan los resultados obtenidos mediante la aplicación de los modelos de aprendizaje supervisado sobre los distintos conjuntos de datos. Se analizan las métricas de evaluación alcanzadas, las configuraciones óptimas halladas mediante búsqueda en malla (*Grid Search*) y la importancia relativa de las características más influyentes en las predicciones.

4.1 Introducción

En esta sección se presentan los resultados obtenidos tras la aplicación de distintos modelos de aprendizaje automático sobre los conjuntos de datos binario y multiclase seleccionados. Se buscó evaluar el rendimiento de cada modelo bajo diferentes configuraciones de hiperparámetros, a través de métricas como la precisión (*Accuracy*), el F1-Score y el área bajo la curva ROC (ROC AUC), entre otras.

Durante la fase experimental se exploraron distintas estrategias de balanceo de clases, dado que varios de los conjuntos presentaban desbalances significativos entre las clases. En particular, se probó la técnica de **undersampling**, reduciendo la cantidad de ejemplos de la clase mayoritaria para equilibrar el dataset. Sin embargo, esta estrategia no arrojó resultados satisfactorios: los modelos tendieron a perder capacidad de generalización, mostrando un descenso notable en las métricas de validación, aunque se sostuvo la mejor configuración con mejor valor de métricas. Por este motivo, se optó finalmente por mantener la distribución original y aplicar técnicas de regularización y ajuste de hiperparámetros para mitigar el sesgo hacia la clase dominante.

A continuación se presentan las métricas finales alcanzadas por cada modelo y los valores óptimos de los hiperparámetros encontrados mediante *Grid Search*. Posteriormente, se analiza la importancia de las características más relevantes en la predicción de las clases.

4.2 Métricas de Evaluación

A continuación, se detallan las principales métricas obtenidas, junto con los hiperparámetros evaluados mediante **Grid Search** y las configuraciones óptimas seleccionadas.

4.2.1 Dataset Binario

Regresión Logística

La Regresión Logística obtuvo un desempeño correcto en la clasificación binaria.

La mejor configuración se alcanzó utilizando un valor de regularización $C = 1$, sin penalización (*Penalty = l1*), con el solver *saga* y estrategia *ovr* (one-vs-rest) para el tratamiento multiclase.

Tabla 4.1: Resultados finales del modelo de Regresión Logística

Configuración	Precisión (Acc)	Recall	F1 Score	ROC AUC	Tiempo (s)	Precisión
$C = 1$ Penalty = l1 Solver = saga Multiclass = ovr	0.84	0.84	0.84	0.90	0.13	0.84

Los resultados cuantitativos finales se resumen en la Tabla 4.1. El modelo logra una precisión y un F1 Score de 0.84 y 0.84 respectivamente, junto con un AUC de 0.90, lo que refleja una buena capacidad discriminatoria, pero siendo datos médicos se requeriría incluso mejores valores.

El tiempo de entrenamiento fue de apenas 0.13 segundos, lo que lo convierte en una opción eficiente para este tipo de problema.

Tabla 4.2: Grid de hiperparámetros - Regresión Logística (binario)

Hiperparámetro	Valores evaluados
C	[0, 0.1, 0.01]
Penalty	[None, l1, l2, elasticnet]
Solver	[lbfgs, saga, newton-s]
Multiclass	[ovr, multinomial]

Máquinas de Soporte Vectorial (SVM)

La Máquinas de Soporte Vectorial obtuvo un desempeño sólido en la clasificación binaria, mostrando un equilibrio adecuado entre precisión y generalización.

La mejor configuración se alcanzó utilizando un valor de regularización $C = 1$, con kernel radial (*Kernel = rbf,*), con un gamma de 0.1 (*Gamma = 0.1*)

Tabla 4.3: Resultados finales del SVM

Configuración	Precisión (Acc)	Recall	F1 Score	ROC AUC	Tiempo (s)	Precisión
$C = 1$ Kernel = rbf Gamma = 0.1	0.86	0.86	0.86	0.92	0.60	0.86

Los resultados cuantitativos finales se resumen en la Tabla 4.3. El modelo logra una precisión y un F1 Score de 0.86 y 0.86 respectivamente, junto con un AUC de 0.92, lo que refleja una mejor capacidad discriminativa. El tiempo de entrenamiento fue de apenas 0.60 segundos, siendo un gran valor.

Tabla 4.4: Grid de hiperparámetros - SVM (binario)

Hiperparámetro	**Valores evaluados**
C	[0.001, 0.01, 0.1, 1, 10, 15, 20, 25]
Kernel	[linear, poly, rbf, sigmoid]
Gamma	[scale, auto, 0.001, 0.01, 0.1, 1]
Degree	[2–10]

Naive Bayes Gaussiano

Naive Bayes Gaussiano obtuvo un buen desempeño en la clasificación multiclase, considerando su supuesto de independencia entre atributos.

La mejor configuración se alcanzó con cualquier suavizado, no hubo diferencias

Tabla 4.5: Resultados finales del Naive Bayes Gaussiano

Configuración	**Precisión (Acc)**	**Recall**	**F1 Score**	**ROC AUC**	**Tiempo (s)**	**Precisión**
var_smoothing = Cualquiera	0.84	0.84	0.84	0.91	0.10	0.84

Los resultados cuantitativos finales se resumen en la Tabla 4.5. El modelo logra una precisión y un F1 Score de 0.84 y 0.84 respectivamente, junto con un AUC de 0.91, lo que refleja una buena capacidad discriminativa, bastante similar con los resultados de los modelos a los cuales se compara.

El tiempo de entrenamiento fue de apenas 0.10 segundos, siendo su mayor fortaleza, la rapidez de su entrenamiento.

Tabla 4.6: Grid de hiperparámetros - Naive Bayes Gaussiano (binario)

Hiperparámetro	**Valores evaluados**
Suavizado	Variaciones de suavizado

Random Forest

Random Forest demostró un desempeño sobresaliente en la clasificación multiclase, mostrando alta capacidad de generalización para registros no vistos durante el entrenamiento.

La mejor configuración se obtuvo utilizando el criterio de *Entropía de Shannon*, profundidad de 7 hojas los árboles, división mínima de 7 ejemplos por nodo, hoja mínima de 1 ejemplo y seleccionando la cantidad de atributos mediante la raíz cuadrada.

Los resultados cuantitativos finales se resumen en la Tabla 4.7.

Tabla 4.7: Resultados finales del Random Forest

Configuración	**Precisión (Acc)**	**Recall**	**F1 Score**	**ROC AUC**	**Tiempo (s)**	**Precisión**
Criterion = entropy Max Depth = 7 Min Samples Split = 5 Min samples Leaf = 1 Max Features = sqrt	0.87	0.87	0.87	0.93	1.38	0.87

El modelo logra una precisión y un F1 Score de 0.87 y 0.87 respectivamente, junto con un AUC de 0.93, lo que refleja una buena excelente capacidad discriminatoria.

El tiempo de entrenamiento fue de apenas 1.38 segundos, siendo considerablemente el modelo en cual más se tarda en obtener los resultados.

Tabla 4.8: Grid de hiperparámetros - Random Forest (binario)

Hiperparámetro	**Valores evaluados**
Criterion	[gini, entropy]
Max Depth	[None, 3, 5, 7, 9]
Min Samples Split	[2, 5, 10]
Min Samples Leaf	[1, 2, 4]
Max Features	[None, sqrt, log2]

En la Figura 4.1 se observa que Random Forest obtiene la mayor puntuación en todas las métricas, seguido por la Regresión Logística y SVM. Esto indica que, para nuestro dataset, el modelo de Random Forest presenta mejor capacidad de generalización, mientras que los demás modelos muestran un desempeño bastante competitivo.

4.3 Importancia de las Características

La importancia de las características se calculó mediante el método de *Permutation Feature Importance*. Este método evalúa cuánto se degrada el desempeño del modelo cuando se altera aleatoriamente una característica, manteniendo fijas las demás. Cuanto mayor sea la disminución en la métrica de desempeño, mayor será la importancia atribuida a dicha característica.

Los resultados obtenidos se presentan en las Tablas 4.9, 4.10, 4.12 y 4.11 correspondientes a los modelos RF, RL, NB y SVM.

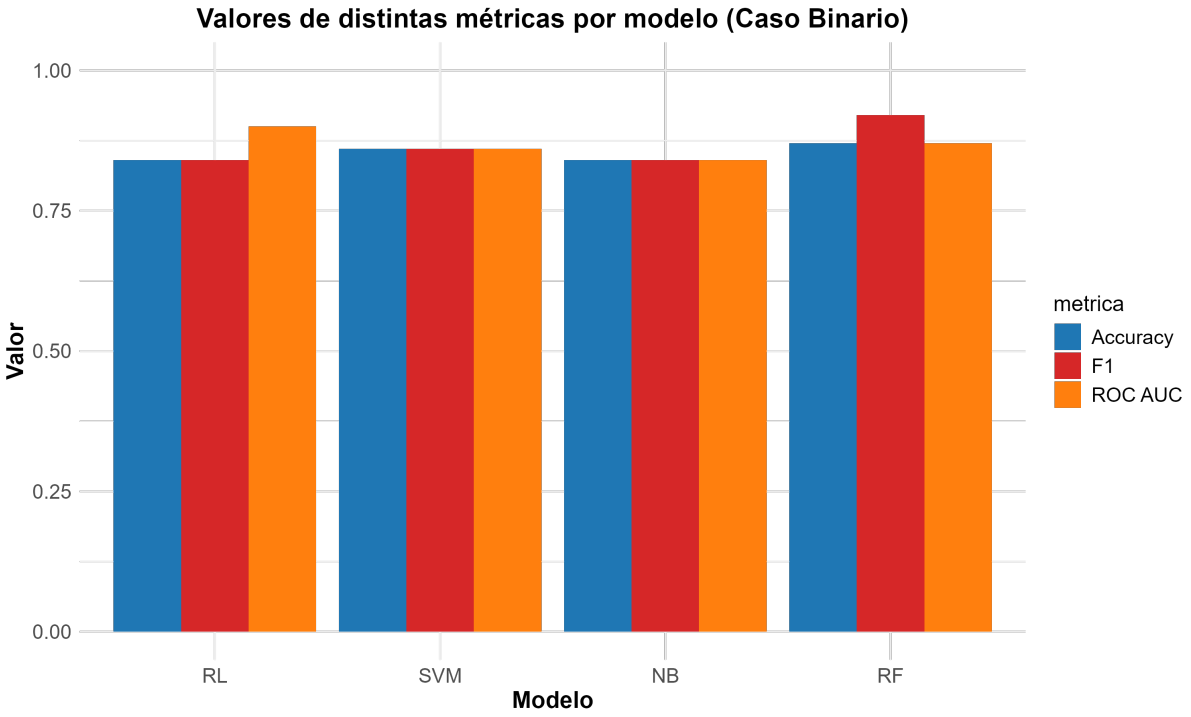

Figura 4.1: Comparación de desempeño de los mejores modelos (Binario)

Tabla 4.12: Importancia de las características según permutación (NB)

Característica	**Importancia (Permutación)**
ST_Slope	0.027015
ExerciseAngina	0.023747
Oldpeak	0.018736
ChestPainType	0.018519
Cholesterol	0.014815
Sex	0.014270
FastingBS	0.004575
RestingBP	0.001852
MaxHR	-0.000218
RestingECG	-0.001198
Age	-0.003595

Tabla 4.9: Importancia de las características según permutación (RF)

Característica	**Importancia (Permutación)**
ST_Slope	0.254265
ChestPainType	0.127319
Oldpeak	0.113156
ExerciseAngina	0.105952
Cholesterol	0.099872
MaxHR	0.088635
Age	0.065807
RestingBP	0.055053
Sex	0.040916
FastingBS	0.030069
RestingECG	0.018956

Tabla 4.10: Importancia de las características según permutación (RL)

Característica	Importancia (Permutación)
ST_Slope	0.072440
ExerciseAngina	0.026797
ChestPainType	0.020806
Sex	0.011329
FastingBS	0.010240
Cholesterol	0.009150
Oldpeak	0.006100
Age	0.003922
MaxHR	0.003268
RestingBP	0.000545
RestingECG	0.000218

Tabla 4.11: Importancia de las características según permutación (SVM)

Característica	Importancia (Permutación)
ST_Slope	0.106209
Cholesterol	0.031808
Oldpeak	0.024510
ChestPainType	0.023312
Sex	0.011438
MaxHR	0.008715
ExerciseAngina	0.008388
Age	0.007952
RestingBP	0.005773
RestingECG	0.004902
FastingBS	0.004575

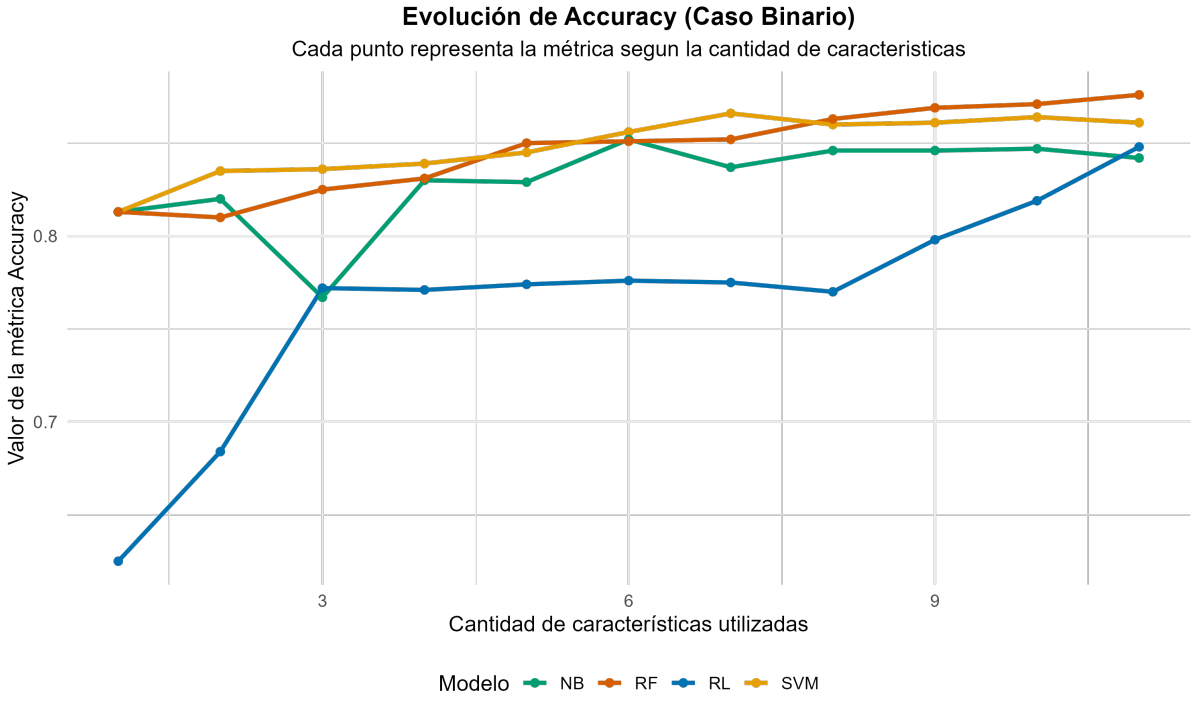

Figura 4.2: Evolución de Accuracy (Binario)

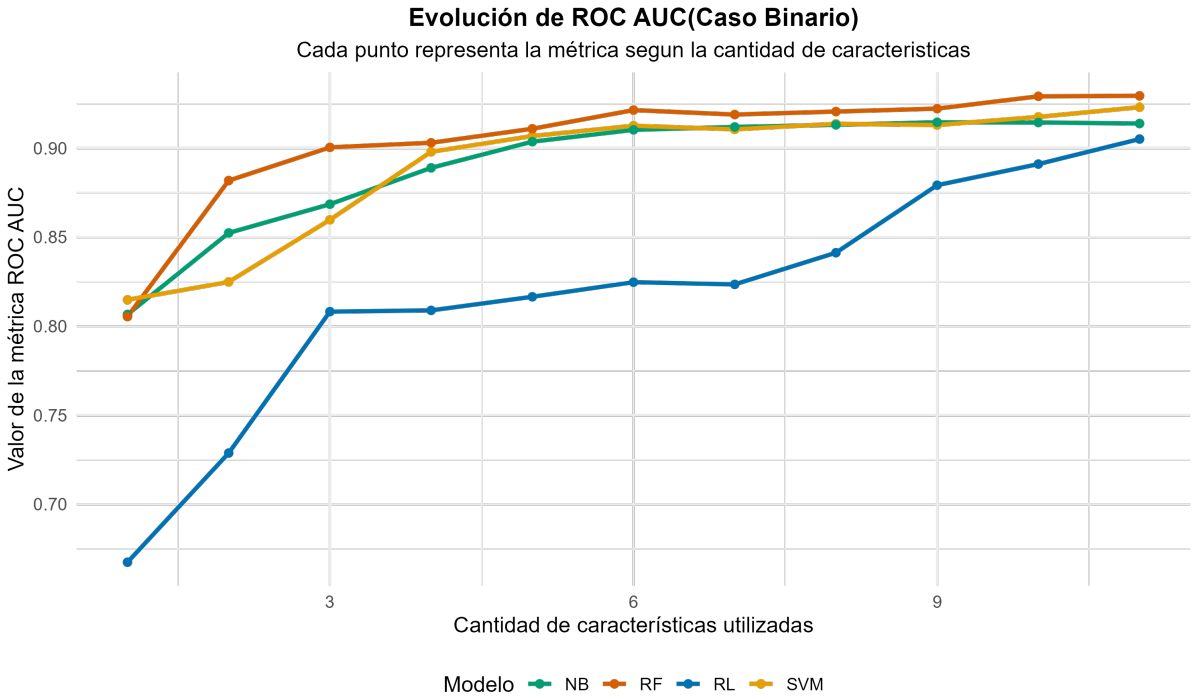

Figura 4.3: Evolución de ROC AUC (Binario)

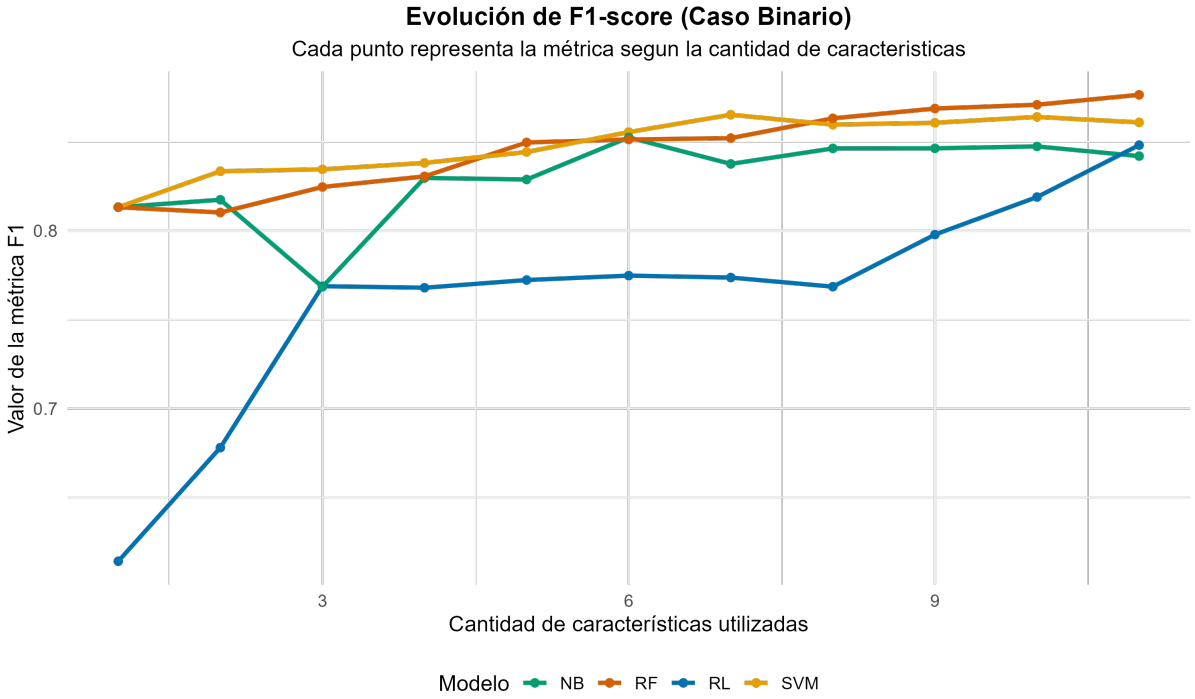

Figura 4.4: Evolución de F1-Score (Binario)

La Figuras 4.2, 4.3 y 4.4 muestran cómo las métricas del modelo mejoran al incorporar las características más relevantes según su importancia. Se observa que inicialmente, con pocas características, el desempeño es limitado, siendo prudente incrementar la cantidad de características. En algunos casos es muy sorprendentemente, ya que se obtiene un buen resultado, y a medida que se agregan las variables de mayor relevancia, las métricas tienden a incrementarse hasta estabilizarse. Este comportamiento permite

identificar el conjunto de características que maximiza el rendimiento sin necesidad de incluir todas las variables disponibles, optimizando tanto la complejidad del modelo como el tiempo de entrenamiento.

Siendo los modelos **SVM** y **RF** los que mejor comportamiento tienen a lo largo del incremento de características, donde podemos ver en su gran mayoría un incremento del valor de las métricas a medida que se aumentan las caracterizaras. Ademas, como se puede observar, tienen valores aceptables para pocas características.

4.3.1 Dataset Multiclase

4.3.2 Resultados del modelo de Regresión Logística

La Regresión Logística obtuvo un desempeño sólido en la clasificación multiclase, mostrando un equilibrio adecuado entre precisión y generalización.

La mejor configuración se alcanzó utilizando un valor de regularización $C = 1$, sin penalización (*Penalty = None*), con el solver *newton-cg* y estrategia *ovr* (one-vs-rest) para el tratamiento multiclase.

Tabla 4.13: Resultados finales del modelo de Regresión Logística

Configuración	**Precisión (Acc)**	**Recall**	**F1 Score**	**ROC AUC**	**Tiempo (s)**	**Precisión**
$C = 1$ Penalty = None Solver = newton-cg Multiclass = ovr	0.89	0.89	0.89	0.96	0.30	0.89

Los resultados cuantitativos finales se resumen en la Tabla 4.13. El modelo logra una precisión y un F1 Score de 0.897 y 0.896 respectivamente, junto con un AUC de 0.964, lo que refleja una buena capacidad discriminativa. El tiempo de entrenamiento fue de apenas 0.30 segundos, lo que lo convierte en una opción eficiente para este tipo de problema.

Tabla 4.14: Grid de hiperparámetros - Regresión Logística (multiclase)

Hiperparámetro	**Valores evaluados**
C	[0.01, 0.1, 1, 10]
Penalty	[None, l2, elasticnet]
Solver	[lbfgs, saga, newton-s]
Multiclass	[ovr]

Máquinas de Soporte Vectorial (SVM)

La Máquinas de Soporte Vectorial obtuvo un desempeño sólido en la clasificación multiclase, mostrando un equilibrio adecuado entre precisión y generalización.

La mejor configuración se alcanzó utilizando un valor de regularización $C = 0.1$, con kernel polinomico de segundo grado (*Kernel = Poly*, *Degree = 2*), con un gamma de 1 (*Gamma = 1*)

Tabla 4.15: Resultados finales del SVM

Configuración	Precisión (Acc)	Recall	F1 Score	ROC AUC	Tiempo (s)	Precisión
$C = 0.1$ Kernel = poly Gamma = 1 Degree = 2	0.90	0.89	0.89	0.96	1.45	0.89

Los resultados cuantitativos finales se resumen en la Tabla 4.15. El modelo logra una precisión y un F1 Score de 0.90 y 0.89 respectivamente, junto con un AUC de 0.96, lo que refleja una buena capacidad discriminativa. El tiempo de entrenamiento fue de apenas 1.42 segundos..

Tabla 4.16: Grid de hiperparámetros - SVM (multiclase)

Hiperparámetro	Valores evaluados
C	[0.001, 0.01, 0.1, 1, 10, 15, 20, 25]
Kernel	[linear, poly, rbf, sigmoid]
Gamma	[scale, auto, 0.001, 0.01, 0.1, 1]
Degree	[2–10]

Naive Bayes Gaussiano

Naive Bayes Gaussiano obtuvo un buen desempeño en la clasificación multiclase, considerando su supuesto de independencia entre atributos.

La mejor configuración se alcanzó con cualquier suavizado, no hubo diferencias

Tabla 4.17: Resultados finales del Naive Bayes Gaussiano

Configuración	Precisión (Acc)	Recall	F1 Score	ROC AUC	Tiempo (s)	Precisión
var_smoothing = Cualquiera	0.82	0.82	0.83	0.91	0.11	0.87

Los resultados cuantitativos finales se resumen en la Tabla 4.17. El modelo logra una precisión y un F1 Score de 0.82 y 0.83 respectivamente, junto con un AUC de 0.91, lo que refleja una buena capacidad discriminativa, pero lo suficientemente menor a los modelos con los cuales se compara. El tiempo de entrenamiento fue de apenas 0.11 segundos, siendo su mayor fortaleza, la rapidez de su entrenamiento.

Tabla 4.18: Grid de hiperparámetros - Naive Bayes Gaussiano (multiclase)

Hiperparámetro	Valores evaluados
Suavizado	Variaciones de suavizado

Random Forest

Random Forest demostró un desempeño sobresaliente en la clasificación multiclase, mostrando alta capacidad de generalización para registros no vistos durante el entrenamiento.

La mejor configuración se obtuvo utilizando el criterio de *Gini*, profundidad ilimitada de los árboles, división mínima de 2 ejemplos por nodo, hoja mínima de 1 ejemplo y seleccionando la cantidad de atributos mediante la raíz cuadrada.

Tabla 4.19: Resultados finales del Random Forest

Configuración	**Precisión (Acc)**	**Recall**	**F1 Score**	**ROC AUC**	**Tiempo (s)**	**Precisión**
Criterion = gini Max Depth = None Min Samples Split = 2 Min samples Leaf = 1 Max Features = sqrt	0.94	0.94	0.94	0.98	2.24	0.94

Los resultados cuantitativos finales se resumen en la Tabla 4.19. El modelo logra una precisión y un F1 Score de 0.94 y 0.94 respectivamente, junto con un AUC de 0.98, lo que refleja una buena excelente capacidad discriminativa. El tiempo de entrenamiento fue de apenas 2.24 segundos..

Tabla 4.20: Grid de hiperparámetros - Random Forest (multiclase)

Hiperparámetro	**Valores evaluados**
Criterion	[gini, entropy]
Max Depth	[None, 3, 5, 7, 9]
Min Samples Split	[2, 5, 10]
Min Samples Leaf	[1, 2, 4]
Max Features	[None, sqrt, log2]

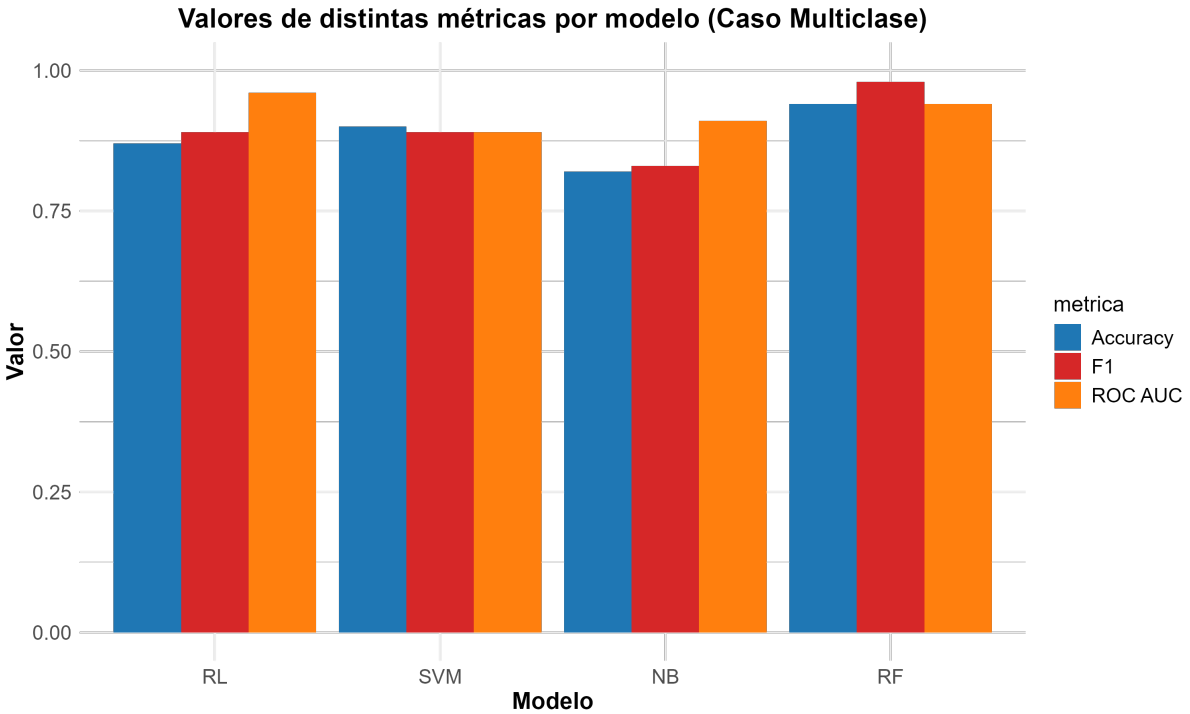

Figura 4.5: Comparación de desempeño de los mejores modelos (Multiclase)

En la Figura 4.5 se observa que Random Forest obtiene la mayor puntuación en todas las métricas, seguido

por la Regresión Logística y SVM. Esto indica que, para nuestro dataset, el modelo de Random Forest presenta mejor capacidad de generalización, mientras que los demás modelos muestran un desempeño bastante competitivo.

4.4 Importancia de las Características

La importancia de las características se calculó mediante el método de *Permutation Feature Importance*. Este método evalúa cuánto se degrada el desempeño del modelo cuando se altera aleatoriamente una característica, manteniendo fijas las demás. Cuanto mayor sea la disminución en la métrica de desempeño, mayor será la importancia atribuida a dicha característica.

Los resultados obtenidos se presentan en las Tablas 4.21, 4.22, 4.24 y 4.23 correspondientes a los modelos RF, RL, NB y SVM.

Tabla 4.21: Importancia de las características según permutación (RF)

Característica	**Importancia (Permutación)**
ASTV	0.139807
ALTV	0.109941
MSTV	0.104823
Mean	0.091579
AC	0.063645
Mode	0.061986
Median	0.060633
DP	0.047945
LB	0.045324
MLTV	0.045132
Variance	0.040531
UC	0.039166
Width	0.030551
Min	0.030109
Max	0.027147
FM	0.020801
Nmax	0.018407
DL	0.011128
Tendency	0.007652
Nzeros	0.003405
DS	0.000287

Las Figuras 4.6, 4.8 y 4.7 muestran cómo las métricas del modelo mejoran al incorporar las características más relevantes según su importancia. Se observa que inicialmente, con pocas características, el desempeño es menor al obtenido anteriormente, pero mucho mejor del esperable. En algunos casos, de forma muy sorprendentemente, se obtiene un buen resultado ya a los pocos atributos utilizados, y a medida que se agregan las variables de mayor relevancia, las métrica tienden a incrementarse hasta estabilizarse.

Este comportamiento permite identificar el conjunto de características que maximiza el rendimiento sin necesidad de incluir todas las variables disponibles, optimizando tanto la complejidad del modelo como el tiempo de entrenamiento.

Tabla 4.22: Importancia de las características según permutación (RL)

Característica	Importancia (Permutación)
Mean	0.098487
AC	0.084113
ASTV	0.057069
Median	0.031631
DP	0.029740
LB	0.023404
Variance	0.022270
UC	0.022080
ALTV	0.019243
Max	0.018109
Nmax	0.014374
Mode	0.011348
Min	0.005910
MSTV	0.004208
FM	0.003830
Tendency	0.003546
MLTV	0.002979
Nzeros	0.002837
DL	0.001655
Width	0.000189
DS	0.000000

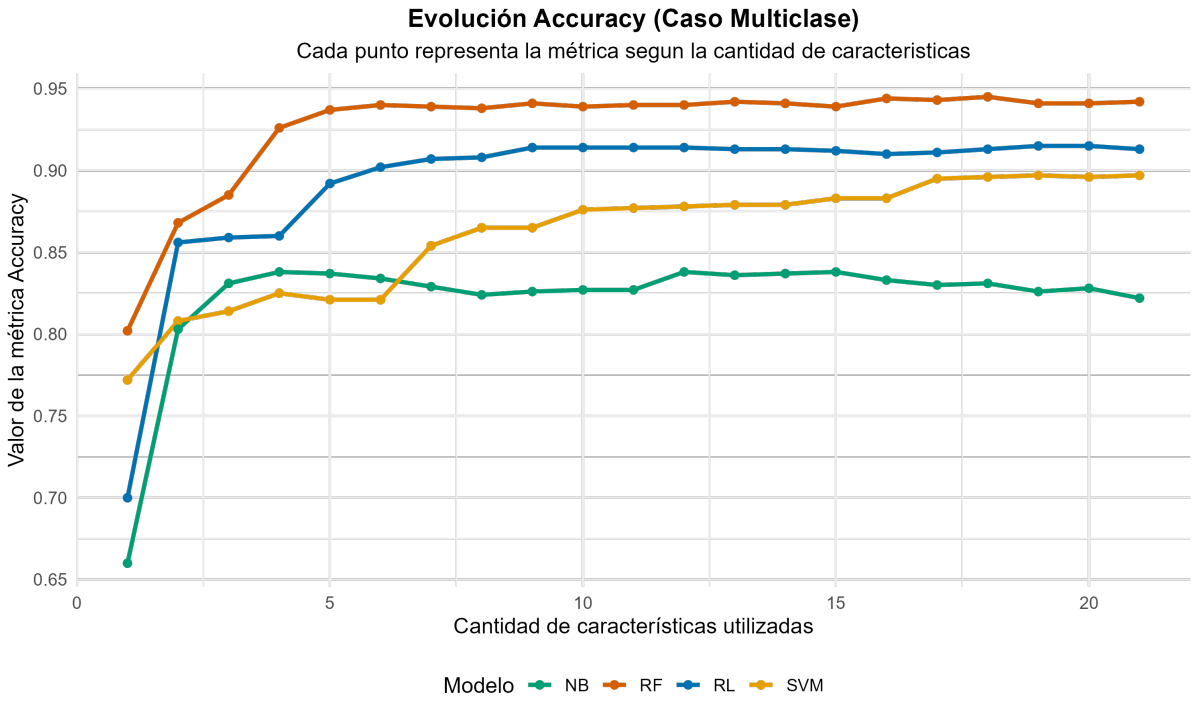

Figura 4.6: Evolución de Accuracy (Multiclase)

Tabla 4.23: Importancia de las características según permutación (SVM)

Característica	Importancia (Permutación)
ASTV	0.050355
ALTV	0.037069
UC	0.030638
AC	0.026903
DP	0.018345
Mean	0.015887
Mode	0.014988
Median	0.014043
Nmax	0.011915
MSTV	0.009125
DL	0.005059
Variance	0.004775
Nzeros	0.004586
Min	0.004444
Max	0.004350
MLTV	0.003357
Tendency	0.003026
FM	0.002459
Width	0.001418
DS	0.000000
LB	-0.000804

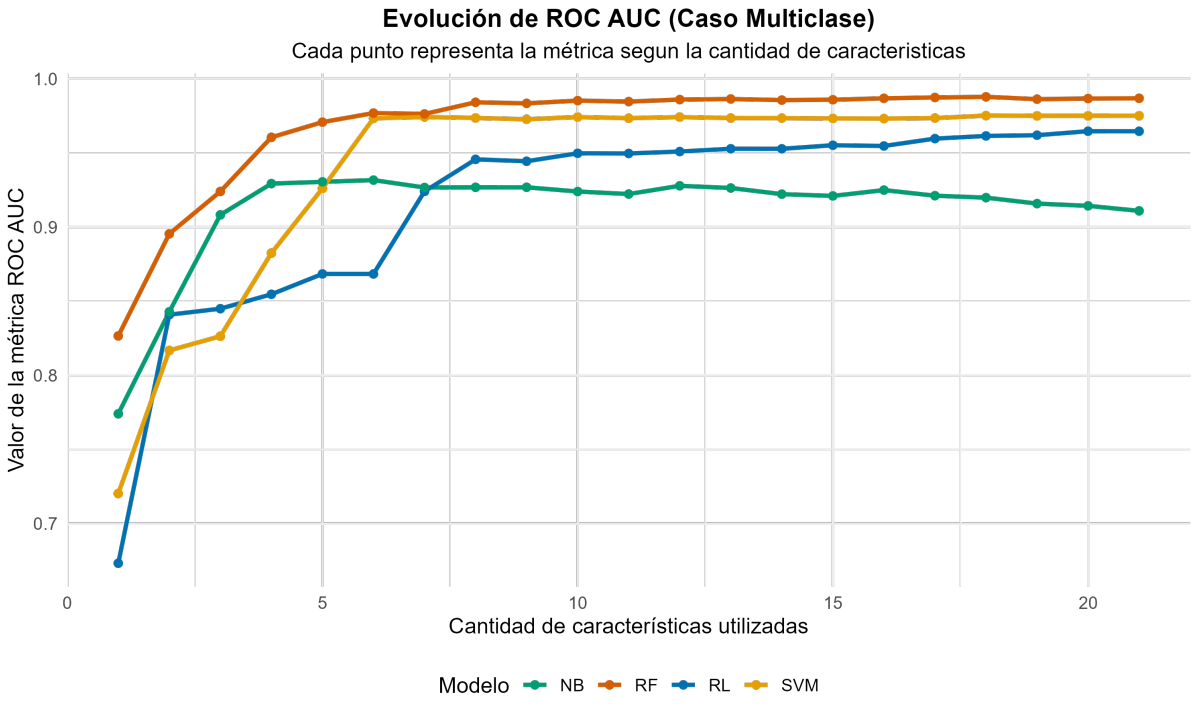

Figura 4.7: Evolución de ROC AUC (Multiclase)

Tabla 4.24: Importancia de las características según permutación (Naive Bayes Gaussiano)

Característica	**Importancia (Permutación)**
AC	0.057163
DP	0.018676
ALTV	0.015461
ASTV	0.005106
DS	0.002695
UC	0.002364
FM	0.001371
Variance	0.001087
Nzeros	0.001040
Nmax	-0.000993
Tendency	-0.001040
Mode	-0.001324
Max	-0.001371
Min	-0.001986
LB	-0.001986
MLTV	-0.002222
Width	-0.002459
Median	-0.003310
MSTV	-0.004965
Mean	-0.005768
DL	-0.006809

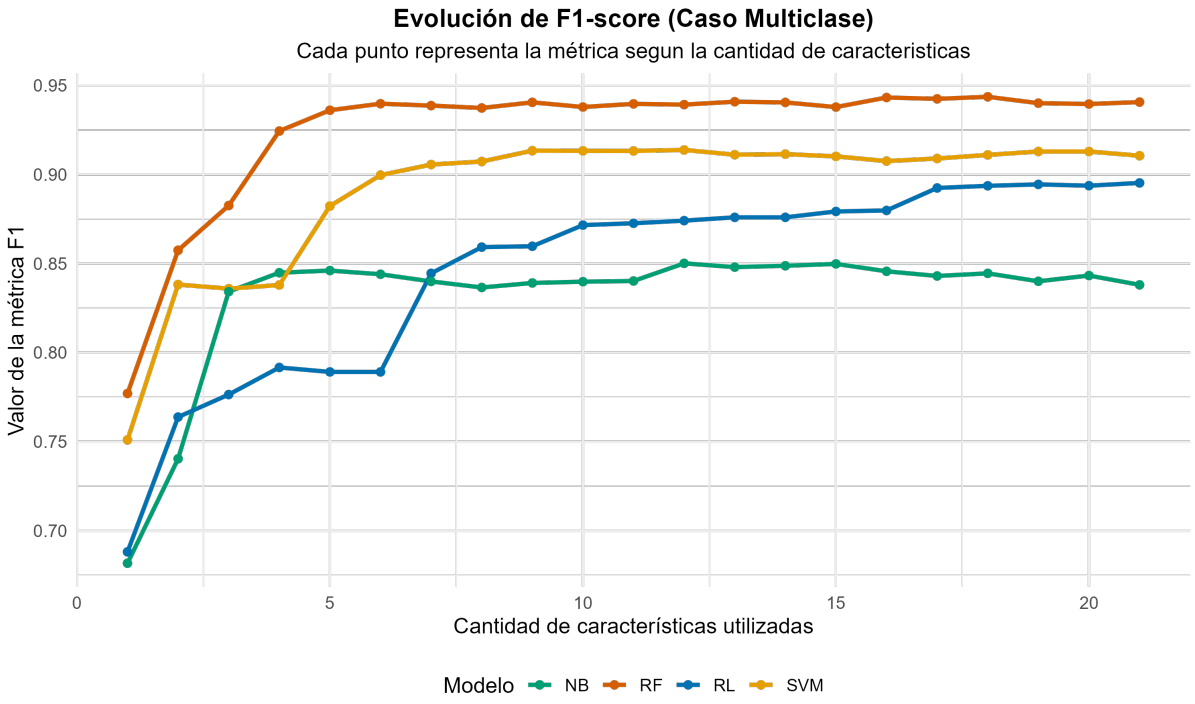

Figura 4.8: Evolución de F1-score (Multiclase)

Chapter 5

Conclusiones

5.1 Análisis General e Inferencias

Del análisis de los resultados obtenidos se puede observar que el modelo **Random Forest** alcanzó reiteradamente los mejores valores en todas las métricas, tanto en el problema binario como en el multiclase. Esto se debe a su capacidad de combinar múltiples árboles de decisión, lo que permite capturar relaciones no lineales y reducir el sobreajuste, sobretodo a la hora de hacer un voto mayoritario de estos mismos arboles que permiten una representación mejor.

El modelo de **SVM** mostró también un rendimiento muy bueno, especialmente con el kernel **RBF** en el caso binario y el **polinómico** en el caso multiclase, destacándose su capacidad para definir fronteras de decisión complejas en espacios transformados.

Regresión Logística presentó resultados positivos y de buena generalización, aunque con menor capacidad para capturar patrones que los otros modelos, no por ser malos resultados, sino que el resto tuvo mejores valores de métricas. Por su parte, el modelo **Naive Bayes** ofreció un rendimiento aceptable, siendo el más liviano computacionalmente, aunque con limitaciones inherentes a su supuesto de independencia de las variables. Esto no quita que aunque posea este supuesto, es el más liviano y rápido de los modelos obteniendo resultados sumamente buenos.

En cuanto a la **importancia de las características**, se identificaron atributos dominantes en cada conjunto de datos. En el binario, variables como *ST_Slope*, *ChestPainType* y *Oldpeak* fueron recurrentemente relevantes; mientras que en el multiclase destacaron *ASTV*, *ALTV* y *MSTV*.

5.2 Mejoras Potenciales y Consideraciones

Para optimizar aún más las métricas, podrían explorarse las siguientes estrategias:

- **Ajuste más fino de hiperparámetros:** empleando *Randomized Search* o *Bayesian Optimization* para reducir tiempos de búsqueda, y luego utlizar un *Grid Search* en los hiperparámetros encontrados.
- **Manipulación de características:** Reducción de dimensionalidad (PCA) o creación de variables sintéticas, para observar mejor las importancias de cada característica.
- **Validación cruzada más robusta:** utilizando más particiones para estimar mejor la generalización.

En conjunto, los modelos demostraron un desempeño satisfactorio, con un claro potencial de mejora mediante el refinamiento de hiperparámetros y una mejor comprensión de la estructura de los datos.

Bibliography

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] J Campos, D. y Bernardes. Cardiotocography. https://doi.org/10.24432/C51S4N., 2000.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[4] Jan Salomon Cramer. The origins of logistic regression. Technical report, Tinbergen Institute discussion paper, 2002.

[5] fedesoriano. Heart failure prediction dataset. https://www.kaggle.com/fedesoriano/heart-failure-predictionhttps://www.kaggle.com/fedesoriano/heart-failure-prediction, September 2021.

[6] David J Hand and Keming Yu. Idiot's bayes—not so stupid after all? *International statistical review*, 69(3):385–398, 2001.

[7] Jonas L Isaksen, Malene Nørregaard, Martin Manninger, Dobromir Dobrev, Thomas Jespersen, Ben Hermans, Jordi Heijman, Gernot Plank, Daniel Scherr, Thomas Pock, et al. Evaluating artificial intelligence-enabled medical tests in cardiology: Best practice. *IJC Heart & Vasculature*, 60:101783, 2025.

[8] Narender Kumar and Dharmender Kumar. Machine learning based heart disease diagnosis using non-invasive methods: A review. In *Journal of Physics: Conference Series*, volume 1950, page 012081. IOP Publishing, 2021.