Comparación de Técnicas de Aprendizaje Automático Supervisado Aplicadas a Datos Cardiólogos

Autor: Nicolás Seivane
Tutora: Andrea Rey

Universidad Nacional de Hurlingham (UNAHUR)

Índice general

Índice de Figuras

Índice de Tablas

Glosario

Notación	Descripción
n	Número total de registros o ejemplos en el dataset.
D	Cantidad de atributos de cada registro.
L	Número total de clases posibles en el problema de clasificación.
$\hat{\theta}_{jvl}$	Probabilidad estimada de que el atributo X_j tome el valor x_{jv} dado que la clase es C_l.
$\hat{\pi}_k$	Probabilidad a priori estimada de que un registro pertenezca a la clase C_k.
$\boldsymbol{x}_i$	Vector de características del registro i, $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$.
X_j	j-ésimo atributo de un registro.
C_l	l-ésima clase del conjunto de clases $\{C_1, C_2, \ldots, C_L\}$.
$\boldsymbol{w}$	Vector normal al hiperplano en SVM.
β	Término de sesgo o intercepto en SVM o regresión logística.
ξ_i	Variable de holgura (*slack*) para el registro i-ésimo en SVM de margen suave.
C	Parámetro de regularización en SVM que controla el equilibrio entre margen y errores.
$K(\boldsymbol{a}, \boldsymbol{b})$	Función kernel que calcula la similitud entre vectores $\boldsymbol{a}$ y $\boldsymbol{b}$.
γ	Parámetro de ancho en el kernel RBF (Radial Basis Function).
r	Coeficiente de desplazamiento en kernels polinómicos y sigmoides.
d	Grado del polinomio en el kernel polinómico.
$\tilde{J}(\boldsymbol{\beta})$	Función de costo regularizada en regresión logística.
α	Parámetro de regularización L1 (Lasso) o L2 (Ridge) en regresión.
$\boldsymbol{H}$	Matriz Hessiana de la función de costo en métodos Newton-CG.
$\boldsymbol{B}$	Aproximación de la matriz Hessiana en BFGS.
$\hat{y}$	Predicción final en un modelo de clasificación.
$\widehat{y_a}(\boldsymbol{x})$	Predicción del árbol a en Random Forest para la observación $\boldsymbol{x}$.
A	Número total de árboles en un Random Forest.
M_l	Valor de la métrica (*accuracy*, F1-score, etc.) para la clase l.
$\lvert C_l \rvert$	Cardinalidad o número de registros pertenecientes a la clase l.
$\widehat{M}_{\text{weighted}}$	Métrica ponderada por tamaño de clase en problemas multiclase.
K	Número de pliegues en validación cruzada K-fold.
G_i	i-ésimo pliegue o grupo en validación cruzada.
$H(D)$	Entropía de Shannon del conjunto de datos D.
$\text{Gini}(t)$	Índice de Gini de un nodo t en un árbol de decisión.
$f(x)$	Función de densidad de probabilidad gaussiana para atributo continuo.

μ_{jl}	Media del atributo X_j en la clase C_l.
σ^2_{jl}	Varianza del atributo X_j en la clase C_l.

Resumen

En este trabajo se realiza un análisis comparativo de distintas técnicas de **Aprendizaje Automático Supervisado** aplicadas a datos cardiológicos, con el objetivo de evaluar su desempeño en la clasificación de pacientes según riesgo o presencia de enfermedad cardíaca.

En primer lugar, se lleva a cabo una **preparación del *dataset***, que incluye limpieza, transformación de variables, selección de atributos y división en conjuntos de entrenamiento y prueba. Se definen las notaciones fundamentales utilizadas a lo largo del trabajo, tales como el número de muestras n, la cantidad de atributos D, el número de clases L, y los parámetros estimados para algunos modelos.

Luego, se describen los principales algoritmos estudiados:

- **Naïve Bayes**, se resalta su fundamento probabilístico y el supuesto de independencia condicional.
- **Regresión Logística**, incluyendo la función logística, el hiperplano de decisión y el proceso de optimización.
- **Máquinas de Vectores de Soporte (SVM)**, considerando márgenes duros y suaves, los hiperparámetros clave (C, γ), el rol del vector normal $\boldsymbol{w}$ y el uso de *kernels*.
- **Árboles de Decisión** y **Random Forest**, destacando la construcción jerárquica de reglas, los criterios de impureza y la reducción de varianza aportada por los métodos de ensamble.

Para cada método se explican los **hiperparámetros relevantes**, su interpretación y cómo afectan el ajuste del modelo. Se especifican los valores considerados o el procedimiento utilizado para seleccionarlos.

Posteriormente, se detalla la **metodología de evaluación**, incluyendo las métricas utilizadas derivadas de la matriz de confusión: exactitud, precisión, recall, especificidad, sensibilidad y F1-Score.

A continuación, se presentan y discuten los **resultados obtenidos** para cada modelo. Se comparan sus desempeños utilizando las métricas seleccionadas. El método con mejor desempeño en ambos casos es el **Random Forest**, donde obtiene valores altos en las métricas utilizadas para evaluar, siendo claro el poder clasificador que se obtiene al tener varios arboles de decision distitnos que clasifican un registro, obteniendo la clasificacion más votada. En contra, se nota como a mayor cantidad de datos y claes para clasificar, el método tarda más cantidad de tiempo en procesar una decisión.

Finalmente, se exponen las **conclusiones generales** del estudio, señalando qué métodos mostraron mejor rendimiento en el *dataset* de cardiología, qué limitaciones presenta el estudio y cómo podrían extenderse o mejorarse los análisis en trabajos futuros mediante técnicas adicionales, mayor optimización o validación más exhaustiva.

Capítulo 1

Introducción

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

Por otro lado, 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.

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 medidas de calidad de un clasificador y, en consecuencia, cual técnica resulta más adecuada para el problema del pre-diagnóstico de enfermedades cardíacas.

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 ser utilizadas por los profesionales médicos como herramientas complementarias para realizar diagnósticos. 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 se utilizan cada vez más en la investigación cardiovascular. El trabajo de Isaksen et al. [7] 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] revisa las metodologías de aprendizaje automático para el diagnóstico de cardiopatías utilizando métodos no invasivos, un área crucial dada la alta mortalidad

anual (17.9 millones de personas) asociada con los problemas cardíacos.

En los estudios cardiológicos, las variables utilizadas pueden agruparse en tres grandes categorías principales: parámetros clínicos estructurados, señales cardíacas y datos provenientes de imágenes médicas. Los parámetros clínicos incluyen variables demográficas, antecedentes, síntomas y mediciones de laboratorio, que suelen encontrarse en bases de datos estructuradas como el conjunto de Cleveland, utilizado comúnmente en estudios de clasificación de enfermedades cardíacas. Las señales cardíacas, como los electrocardiogramas (ECG) y fonocardiogramas (PCG), aportan información temporal y frecuencial altamente relevante para la detección de arritmias, soplos o patrones eléctricos anómalos. Finalmente, los datos de imagen —que abarcan ecocardiografías, resonancias magnéticas cardíacas (CMR), tomografías computarizadas (CCT) o estudios SPECT— permiten evaluar morfología, función cardíaca y perfusión.

En el ámbito de los métodos de aprendizaje automático aplicados a cardiología, varios trabajos han empleado los mismos algoritmos utilizados en este estudio. Por ejemplo, modelos basados en *Support Vector Machines* (SVM) han demostrado un rendimiento competitivo en la clasificación de ECG y en la detección de enfermedades a partir de parámetros clínicos, mostrando en muchos casos un desempeño superior a métodos más simples. Los clasificadores k-Nearest Neighbors (k-NN) han sido utilizados para tareas como la detección de arritmias mediante características temporales del ECG o para la predicción de riesgo a partir de datos clínicos estructurados. Otros estudios han empleado Naïve Bayes o árboles de decisión para la clasificación de cardiopatías utilizando tanto datos tabulares como señales procesadas. Estas aplicaciones muestran que los modelos implementados en este trabajo están bien representados dentro de la literatura del área y cuentan con antecedentes sólidos en tareas diagnósticas dentro del dominio cardiológico.

1.3 Conjuntos de Datos

Para la realización de este trabajo se exploraron diversas plataformas en busca de conjuntos de datos reales que resulten 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 otro *dataset* de **clasificación multiclase**, con más de dos categorías posibles.

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

Todos estos *datasets* son procesados anteriormente a las pruebas realizadas, en donde se eliminaron tanto registros duplicados como con datos faltantes, tampoco se tienen en cuenta aquellos con datos atípicos *a priori*, como un caso donde un paciente tiene colesterol 0.

Se realiza una pequeña descripción de cada atributo utilizado en cada *dataset*. Si el atributo es categórico, se informa la distribución de los valores que posee dicho atributo y en caso de resultar ser un atributo numérico, se informa la media de los valores de dicho atributo, junto con el valor máximo y mínimo que posee. Se realiza lo anterior para contextualizar los atributos y ver los rangos de valores con que se trabajará.

1.3.1 Dataset Binario: *Insuficiencia Cardíaca Predicción*

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.

En la Tabla 1.2 se muestra qué tipo de datos son los atributos del *dataset* que son utilizados en este trabajo, considerando que estos mismos ya fueron previamente procesados para poder ser utilizados en las técnicas de aprendizaje automático. Luego, la Tabla 1.1 muestra la cantidad de registros y atributos que son utilizados, junto al tipo de dato que son.

Tabla 1.1: Cantidad de registros utilizados.

Cantidad de registros	**918**
Cantidad de atributos	**11**
Atributos Categóricos	**5**
Atributos Numéricos	**6**

Tabla 1.2: Tipo de atributo del conjunto binario.

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

Descripción de los atributos:

Age: Este atributo refiera a la edad de los pacientes. Los pacientes tienen una media de edad de 53 años, con una edad máxima de 77 y de edad mínima de 28, con proporciones de edad bastante bien distribuidas, siendo la menor de 0.11 % para algunas 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 de sexo del 78.98 % masculinos y el 21.02 % femeninos.

ChestPainType: Tipo del dolor en el pecho, con exactitud dolor torácico causado por isquemia cardíaca. Tiene una distribución 18.85 % de angina atípica, luego un 22.11 % de dolor no anginoso, un 54.03 % asintomático, y un 5.01 % de dolor de pecho anginoso típico.

RestingBP: Se está 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 sérico, la medida total de colesterol en sangre; tiene un valor medio en los pacientes de 199.02, con un valor máximo de 603.00 y un valor mínimo de 0.00. Se encuentra en miligramos por decilitro.

FastingBS: Es la Glucosa en sangre en ayuno; hay un 76.66 % de registros con valores de glucosa en sangre menores a 120 mg/dl, codificado en 0, y un 23.34 % con valores mayores a 120 mg/dl, codificado en 1.

RestingECG: Son los resultados de electrocardiogramas en reposo; hay 60.09 % codificado en Normal, un 19.41 % codificado en ST (tiene una anormalidad en el estudio) y, por último, un 20.50 % codificado en LVH (probablemente una hipertrofia en el ventrículo izquierdo).

MaxHR: Este atributo es el máximo ritmo cardíaco registrado, tiene una media en los pacientes de 136.79, con un valor máximo de 202.00 y un valor mínimo de 60.00.

ExerciseAngina: Es la angina producido por ejercicio, dolor en el pecho; donde hay un 59.54 % que no tenían dolor, codificado en N, y hay un 40.46 % que sí tenían dolor, 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 una media de 0.90, valor máximo de 6.20 y valor mínimo de -0.10.

ST_Slope: La pendiente del segmento ST durante el ejercicio máximo; hay un 43.08 % en Up, un 50.05 % en Flat y luego un 6.87 % en Down [Up: pendiente ascendente, Flat: pendiente plana, Down: pendiente descendente].

HeartDisease: Variable de salida si posee una enfermedad cardíaca; donde hay un 44.71 % que no tiene enfermedad cardíaca, codificado en 0, y hay un 55.29 % que sí tienen enfermedad cardíaca, codificado en 1. Siendo ésta la **variable objetivo**.

1.3.2 Dataset Multiclase: *Cardiotocografía Predicción*

En el *dataset* [2] utilizado se procesaron automáticamente 2126 cardiotocogramas fetales 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 con respecto al estado fetal, el cual puede ser

normal, sospechoso o patológico.

En la Tabla 1.4 se muestran los tipos de las variables utilizadas en este trabajo, junto con la media y valores tanto máximos como mínimos. Luego, la Tabla 1.3 muestra la cantidad de registros y atributos que son utilizados, junto al tipo de dato que son.

Tabla 1.3: Cantidad de registros utilizados.

Cantidad de registros	**2115**
Cantidad de atributos	**21**
Atributos Categóricos	**0**
Atributos Numéricos	**21**

Tabla 1.4: Tipo de atributo del conjunto multiclase.

Atributo	**Tipo de dato**	**Media**	**Máximo**	**Mínimo**
LB	Numérico (int)	133.301	160.000	106.000
AC	Numérico	0.003	0.019	0.000
FM	Numérico (float)	0.009	0.481	0.000
UC	Numérico (float)	0.004	0.015	0.000
DL	Numérico (float)	0.001	0.015	0.000
DS	Numérico (float)	-	-	-
DP	Numérico (float)	-	-	-
ASTV	Numérico (int)	46.977	87.000	12.000
MSTV	Numérico (float)	1.335	7.000	0.200
ALTV	Numérico (int)	9.759	91.000	0.000
MLTV	Numérico (float)	8.170	50.700	0.000
Width	Numérico (int)	70.511	180.000	3.000
Min	Numérico (int)	93.574	159.000	50.000
Max	Numérico (int)	164.085	238.000	122.000
Nmax	Numérico (int)	4.075	18.000	0.000
Nzeros	Numérico (int)	-	-	-
Mode	Numérico (int)	137.448	187.000	60.000
Mean	Numérico (int)	134.596	182.000	73.000
Median	Numérico (int)	138.084	186.000	77.000
Variance	Numérico (int)	18.891	269.000	0.000
Tendency	Numérico (int)	-	-	-
NSP	Categórico (string)	-	-	-

Descripción de los atributos:

LB Frecuencia cardíaca fetal basal (latidos por minuto).

AC Número de aceleraciones por segundo.

FM Número de movimientos fetales por segundo.

UC Número de contracciones uterinas por segundo.

DL Número de desaceleraciones leves por segundo.

DS Número de desaceleraciones severas por segundo.

DP Número de desaceleraciones prolongadas por segundo. Hay un 91.58 % con valor de 0, un 3.40 % con valor 0.002, un 1.13 % con valor 0.003, un 3.31 % con valor 0.001, un 0.43 % con valor 0.004 y un 0.14 % con valor 0.005.

ASTV Porcentaje de tiempo con variabilidad anormal a corto plazo.

MSTV Valor medio de la variabilidad a corto plazo.

ALTV Porcentaje de tiempo con variabilidad anormal a largo plazo.

MLTV Valor medio de la variabilidad a largo plazo.

Width Ancho del histograma de CTG.

Min Mínimo del histograma de CTG.

Max Máximo del histograma de CTG.

Nmax Número de picos del histograma de CTG.

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

Mode Moda del histograma de CTG.

Mean Promedio del histograma de CTG.

Median Mediana del histograma de CTG.

Variance Varianza del histograma de CTG.

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

CLASS Código de clasificación del estado fetal (N = normal; S = sospechoso; P = patológico). Hay un 13.81 % sospechosos, 77.92 % normales y 8.27 % patológicos. Siendo ésta la **variable objetivo**.

Capítulo 2

Descripción de los Métodos Utilizados

En este capítulo se presentan los métodos de aprendizaje automático supervisado que se emplearon en este trabajo. Se describen sus fundamentos teóricos, las fórmulas y notaciones utilizadas, así como los hiperparámetros y criterios de optimización más relevantes. El objetivo es describir y explicar cómo funcionan cada uno de los algoritmos, sus supuestos y la forma en que se aplican a los datos cardiólogos analizados.

Los métodos requieren de **hiperparámetros**, que son un parámetro cuyo valor se fija antes del proceso de entrenamiento de un modelo y que controla aspectos del aprendizaje, como la complejidad del modelo, la velocidad de convergencia o la regularización. A diferencia de los parámetros internos del modelo (por ejemplo, los pesos en una regresión logística o en una red neuronal), los hiperparámetros no se aprenden automáticamente a partir de los datos, sino que deben ser seleccionados mediante experimentación, validación cruzada u optimización de hiperparámetros, siendo un análisis en sí mismos.
Los **hiperparámetros** utilizados en este trabajo corresponden a aquellos definidos por la librería *scikit-learn* del lenguaje de programación *Python*, la cual ofrece implementaciones estandarizadas y bien documentadas de modelos clásicos de aprendizaje automático. En particular, la búsqueda en grilla se realiza con los valores posibles que determina la librería, junto con la combinación de valores que son posibles de combinar, como el grado del polinomio cuando utilizamos el kernel de polinomia en **SVM**.

En particular, se emplearon los valores por defecto sugeridos por la librería y, cuando fue necesario, se ajustaron manualmente algunos hiperparámetros relevantes —como la profundidad máxima del árbol, el criterio de partición o la cantidad de vecinos en modelos basados en distancia— con el fin de evaluar su influencia en el desempeño del modelo. Este procedimiento permite garantizar consistencia en las comparaciones, reproducibilidad de los experimentos y una interpretación clara acerca del efecto de cada hiperparámetro sobre los resultados.

2.1 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. 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 los 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 estimar la probabilidad condicional de que una observación pertenezca a la clase positiva (o clase objetivo). Sea C_l el conjunto de clases posibles, con $l \in \{1, 2, \ldots, L\}$. Para cada registro i (con $i = 1, 2, \ldots, n$), su etiqueta verdadera se denota por l_i, donde definimos:

$$l_i = \begin{cases} 1, & \text{si la observación pertenece a la clase positiva } C_1, \\ 0, & \text{si la observación pertenece a la clase negativa } C_0. \end{cases}$$

$$P(l_i = 1 \mid \boldsymbol{x}_i), \tag{2.1}$$

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, en el cual D es la cantidad de atributos que posee la observación.

Por lo cual, la función logística define la **función de probabilidad** de pertenencia de $\boldsymbol{x}_i$ a la clase 1 como:

$$p(\boldsymbol{x}_i) = P(l_i = 1 \mid \boldsymbol{x}_i) = \frac{\exp(\beta_0 + \sum_{j=1}^{D} \beta_j x_{ij})}{1 + \exp(\beta_0 + \sum_{j=1}^{D} \beta_j x_{ij})}, \tag{2.2}$$

donde β_0 es el intercepto y β_j son los coeficientes asociados a cada predictor, para $j = 1, 2, \ldots, D$.

La función inversa de la función logística, denominada ***función 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,

$$\text{logit}(p(\boldsymbol{x}_i)) = \ln\left(\frac{p(\boldsymbol{x}_i)}{1 - p(\boldsymbol{x}_i)}\right) = \beta_0 + \sum_{j=1}^{D} \beta_j x_{ij}. \tag{2.3}$$

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 último punto un factor que realiza el cálculo de coeficientes muy difícil. Para solucionar este obstáculo, el cálculo de los coeficientes de regresión se estiman mediante el método de **Máxima Verosimilitud** (MLE), donde la función de log-verosimilitud a maximizar es

$$\ell(\beta_0, \beta_1, \ldots, \beta_k) = \sum_{i=1}^{n} \left[l_i \ln(p(\boldsymbol{x}_i)) + (1 - l_i) \ln(1 - p(\boldsymbol{x}_i)) \right], \tag{2.4}$$

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

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 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):

$$\tilde{J}_{L1}(\boldsymbol{\beta}) = J(\boldsymbol{\beta}) + \alpha \sum_{j=1}^{D} |\beta_j|, \tag{2.5}$$

  donde

  - β_j son los coeficientes del modelo, que representan la influencia de cada variable x_j en la predicción, para $j = 1, 2, \ldots, k$;
  - Donde $\alpha \geq 0$ es el parámetro de regularización que controla la intensidad de la penalización: valores más altos implican mayor castigo a los coeficientes.

- Regularización L2 (Ridge):

$$\tilde{J}_{L2}(\boldsymbol{\beta}) = J(\boldsymbol{\beta}) + \frac{1}{2}\alpha \sum_{j=1}^{D} \beta_j^2, \tag{2.6}$$

  donde

  - β_j controla cuánto influye cada característica sobre la salida;
  - $\alpha \geq 0$ determina cuánto se penaliza el tamaño de los coeficientes, evitando que crezcan demasiado.

- Regularización Elastic Net:

$$\tilde{J}_{EN}(\boldsymbol{\beta}) = J(\boldsymbol{\beta}) + \alpha \left[\rho \sum_{j=1}^{D} |\beta_j| + \frac{1-\rho}{2} \sum_{j=1}^{D} \beta_j^2 \right], \tag{2.7}$$

  donde

  - β_j son los coeficientes asociados a cada característica;
  - $\alpha \geq 0$ controla el grado total de regularización;
  - $\rho \geq 0$ balancea la combinación entre L1 y L2. Si es 0 entonces tenemos L2 y si es 1 tenemos L1.

- **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).

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{\beta}} \tilde{J}(\boldsymbol{\beta}^{(t)}), \tag{2.8}$$

    donde $\boldsymbol{\beta}$ es el vector de parámetros del modelo que se actualiza en cada iteración. El método utiliza el gradiente $\nabla_{\boldsymbol{\beta}}$ y la inversa de la matriz Hessiana $\boldsymbol{H}$ (o una aproximación de la misma mediante *Conjugate Gradient*) para determinar la dirección de actualización del vector de parámetros.

  - Método BFGS (quasi-Newton).

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \boldsymbol{B}^{-1} \nabla_{\boldsymbol{\beta}} \tilde{J}(\boldsymbol{\beta}^{(t)}). \tag{2.9}$$

    En este método, $\boldsymbol{\beta}$ también es el vector de parámetros a optimizar. La matriz $\boldsymbol{B}$ aproxima al Hessiano, permitiendo actualizaciones sin calcular derivadas segundas reales.

  - Método Saga. Un algoritmo de tipo *stochastic gradient descent* con soporte para regularización L1 y L2. Permite trabajar con grandes datasets y converger de manera estable incluso con penalización L1.

  - Método Liblinear. Optimiza la función de coste usando un método de descenso por coordenadas, eficiente para datasets grandes y modelos lineales. Soporta L1 y L2.

- **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.

2.2 Árboles de Decisión

El aprendizaje mediante **Árboles de Decisión** 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 más sencillo de realizar, donde se crea un camino de decisión, por lo cual según los valores de los atributos de un registro podemos predecir a que clase pertenecen C_l, donde el cómputo pesado se encuentra en la creación del propio camino.

En consecuencia, la probabilidad de que un ejemplo en el nodo t pertenezca a la clase C_l se define como

$$p(l|t) = \frac{N_l(t)}{N(t)}, \tag{2.10}$$

donde $N(t)$ es la cantidad total de ejemplos en el nodo t, y $N_l(t)$ la cantidad de ejemplos de la clase C_l. 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 está dada por la cantidad de ejemplos de la clase que tienen en el mismo.

De esta manera, la impureza de un nodo t se mide mediante una función ϕ que depende de las probabilidades de clase en dicho nodo. La función de impureza im es un hiperparámetro en sí mismo, y debe cumplir las siguientes condiciones, para una función de densidad de probabilidad $\mathbf{p}$:

- **No negatividad:**
$$\phi(\mathbf{p}) \geq 0.$$
- **Impureza mínima en nodos puros:**
$$\phi(\mathbf{p}) = 0 \quad \text{si existe } l \text{ tal que } p_l = 1.$$
- **Máxima impureza con distribución uniforme:**
$$\phi(\mathbf{p}) \text{ es máxima cuando } p_l = \frac{1}{L} \quad \forall l.$$

Tanto el índice de Gini como la entropía de Shannon satisfacen estas propiedades, por lo que son funciones de impureza válidas para la construcción del árbol.

$$im(t) = \phi\big(p(1|t), p(2|t), \ldots, p(K|t)\big). \tag{2.11}$$

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

Dentro de las funciones de impureza más utilizadas, se encuentran la entropía de Shannon y el índice de Gini.

La entropía de Shannon mide el grado de desorden o incertidumbre en el conjunto de datos D. Toma valores cercanos a cero cuando el nodo es puro y crece a medida que las clases se distribuyen de forma más uniforme,

$$H(D) = -\sum_{k=1}^{K} \frac{N_k(D)}{N(D)} \log_2 \left(\frac{N_k(D)}{N(D)} \right). \tag{2.12}$$

El índice de Gini cuantifica la probabilidad de clasificar incorrectamente una instancia si se asigna aleatoriamente según la distribución de clases en t. Al igual que la entropía, es mínimo cuando el nodo es puro y aumenta cuando las clases están mezcladas.

$$\text{Gini}(t) = 1 - \sum_{k=1}^{K} \big[p(k|t)\big]^2 = 1 - \sum_{k=1}^{K} \left(\frac{N_k(t)}{N(t)} \right)^2. \tag{2.13}$$

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

$$\Delta im(s,t) = im(t) - q_1 im(t_1) - q_2 im(t_2), \tag{2.14}$$

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

2.2.1 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

$$\hat{y} = \underset{l \in \mathcal{L}}{\text{argmax}} \sum_{a=1}^{A} \mathbb{I}\big(\widehat{y_a}(\boldsymbol{x}) = l\big), \tag{2.15}$$

donde $\widehat{y_a}(\boldsymbol{x})$ es la predicción del árbol a, A es la cantidad total de árboles en el bosque $\mathbb{I}$ es la **función indicadora**, definida como

$$\mathbb{I}(\widehat{y_a}(\boldsymbol{x}) = l) = \begin{cases} 1, & \text{si el árbol } a \text{ asigna la clase } l, \\ 0, & \text{en caso contrario.} \end{cases}$$

Esta función contabiliza cuántos árboles votan por cada clase l, permitiendo que el modelo escoja aquella con mayor cantidad de votos.

Hiperparámetros

- **Criterio de Partición:** Función de impureza utilizada para evaluar la calidad de una división. En `scikit-learn`, las opciones disponibles son: **`'gini'`** (índice de Gini) y **`'entropy'`** (entropía de Shannon).
- **Algoritmo de Construcción:** *ID3* emplea la ganancia de información (entropía), mientras que *CART* utiliza el índice de Gini y construye árboles binarios. En `scikit-learn`, el árbol implementado corresponde a una versión optimizada de *CART*.

- **Número de Atributos Muestreados (`max_features`):** Cantidad de atributos candidatos por división. En Random Forest, las opciones típicas son:

$$\text{``sqrt''} = \sqrt{a}, \qquad \text{``log2''} = \ln(a),$$

  o bien un número entero, un porcentaje o `None` (usar todos los atributos).

- **Número de Árboles (`n_estimators`):** Cantidad total de árboles que componen el bosque en Random Forest. Valores altos reducen la varianza pero aumentan el costo computacional.

- **Profundidad Máxima (`max_depth`):** Límite superior a la altura del árbol. Un valor `None` permite crecer hasta que las hojas sean puras o no haya más divisiones posibles.

- **Tamaño Mínimo de Muestra para Dividir (`min_samples_split`):** Número mínimo de muestras requerido para realizar una partición interna.

- **Tamaño Mínimo de Muestra en una Hoja (`min_samples_leaf`):** Cantidad mínima de muestras que debe contener una hoja para evitar sobreajuste.

- **Profundidad Mínima del Nodo Interno (`min_impurity_decrease`):** Umbral mínimo de reducción de impureza necesario para aceptar una división.

- **Bootstrap (`bootstrap`):** Indica si los árboles del bosque se entrenan con muestras generadas mediante remuestreo con reemplazo. Por defecto, `True` en Random Forest.

2.3 Clasificador Naïve Bayes

El **Clasificador Naïve Bayes** [6] (NB) es un método supervisado probabilístico basado en el *Teorema de Bayes*, que asume independencia condicional entre los atributos dado la clase. Si bien esta suposición rara vez se cumple estrictamente en la práctica —debido a que las variables suelen estar correlacionadas entre sí, el método continúa funcionando de manera efectiva en numerosos problemas reales. La regla de clasificación, o la probabilidad de pertenencia de un registro a una clase C_l, donde para cada clases $l \in \{1, 2, \ldots, L\}$, donde cada registro tiene D atributos, indexados por $d \in \{1, 2, \ldots, D\}$.

$$P(Y = C_l \mid X_1 = x_1, \ldots, X_d = x_d) = \frac{P(X_1 = x_1, \ldots, X_d = x_d \mid Y = C_l)\, P(Y = C_l)}{P(X_1 = x_1, \ldots, X_d = x_d)}, \tag{2.16}$$

En donde $P(Y = C_l)$ es la **probabilidad *a priori*** de la clase C_l (qué tan frecuente es en el conjunto de datos). Luego, $P(X_1 = x_1, \ldots, X_d = x_d \mid Y = C_l)$ es la **verosimilitud**, es decir, cuán probable es observar las características de un registro si supiéramos que pertenece a la clase C_l.

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

$$P(X_1 = x_1, \ldots, X_d = x_d \mid Y = C_l) = \prod_{j=1}^{d} P(X_j = x_j \mid Y = C_l), \tag{2.17}$$

Por lo tanto, basta estimar individualmente, para cada atributo X_j con $j \in \{1, 2, \ldots, D\}$ y para cada clase C_l con $l \in \{1, 2, \ldots, L\}$, la probabilidad:

$$\hat{\theta}_{jvl} = P(X_j = x_{jv} \mid Y = C_l) = \frac{|\{\, X_j = x_{jv} \wedge Y = C_l \,\}|}{|Y_l|}, \tag{2.18}$$

donde $v \in \{1, 2, \ldots, V_j\}$ indexa los valores posibles del atributo X_j.
Aquí, cada parámetro cumple una función específica:

- $\hat{\theta}_{jvl}$: probabilidad estimada de que el atributo X_j tome el valor x_{jv} dentro de la clase C_l. Representa qué tan típico es ese valor dentro de una clase.
- $|\{ X_j = x_{jv} \wedge Y = C_l \}|$: cantidad de registros en los que el atributo X_j toma el valor x_{jv} y simultáneamente la clase es C_l.
- $|C_l|$: número total de registros pertenecientes a la clase C_l.

Podemos pensar a θ_{jmk} como una especie de *"frecuencia interna de la clase"*, una medida que indica qué características son más comunes dentro de cada grupo.

Probabilidad *a priori*: Otra estimación fundamental a calcular y expresar es la *a priori*.

$$\hat{\pi}_l = P(Y = C_l) = \frac{\#\{Y = C_l\}}{n'}. \tag{2.19}$$

Los parámetros involucrados son:

- $\hat{\pi}_l$: probabilidad estimada de que un registro pertenezca a la clase C_k. Corresponde a la frecuencia relativa de esa clase en la muestra.
- $|C_l|$: cantidad de registros cuya clase es C_l
- n: tamaño muestral total.

La intuición es sencilla: antes de mirar ningún atributo, ¿qué tan probable es cada clase?

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:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right], \tag{2.20}$$

por lo cual la decisión final se obtiene como

$$\hat{Y} = \underset{C_k}{\operatorname{argmax}} \left[\log P(Y = C_l) + \sum_{j=1}^{d} \log f(x_j \mid \mu_{jl}, \sigma_{jl}^2)\right]. \tag{2.21}$$

donde los parámetros de la distribución normal para cada atributo X_j y clase C_l se estiman como

$$\hat{\mu}_{jl} = \frac{1}{|C_l|} \sum_{i:Y_i = C_l} x_{ij}, \tag{2.22}$$

$$\hat{\sigma}_{jl}^2 = \frac{1}{|C_l|} \sum_{i:Y_i = C_l} (x_{ij} - \hat{\mu}_{jl})^2. \tag{2.23}$$

2.4 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** φ, 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.

El caso de **Margen Rígido (Hard Margin)** separa perfectamente las clases sin errores de clasificación. En este caso, el problema de optimización se formula como:

$$\text{Minimizar } \frac{1}{2}|\boldsymbol{w}|^2 \quad \text{sujeto a } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + \beta) \geq 1, \tag{2.24}$$

donde $\boldsymbol{w}$ es el vector normal al hiperplano, β es el término de sesgo, $\boldsymbol{x}_i$ es el i-ésimo registro del conjunto de datos, $y_i \in \{-1, +1\}$ es la etiqueta asociada a dicho registro, y la restricción garantiza que todas las observaciones queden correctamente clasificadas y a una distancia mínima de $\frac{1}{\|\boldsymbol{w}\|}$ del hiperplano.

Pero hay un gran obstáculo, debido a que en la práctica los datos rara vez son perfectamente separables. Por ello, se introduce el concepto de **Margen Suave (Soft Margin)**, que permite violaciones controladas de las restricciones mediante variables de holgura $\xi_i \geq 0$. El problema se redefine como:

$$\text{Minimizar } \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i \quad \text{sujeto a } \begin{cases} y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + \beta) \geq 1 - \xi_i, \\ \xi_i \geq 0, \end{cases} \tag{2.25}$$

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.

La **Formulación Dual** del problema, en lo que respecta a los dos margenes permite expresar la solución en términos de los productos internos entre las observaciones:

$$\text{Maximizar } -\frac{1}{2}\sum_{i,\ell=1}^{n} \alpha_i \alpha_\ell y_i y_\ell K(\boldsymbol{x}_i, \boldsymbol{x}_\ell) + \sum_{i=1}^{n} \alpha_i, \tag{2.26}$$

$$\text{sujeto a } 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0. \tag{2.27}$$

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

En las siguientes definiciones, los vectores $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^D$ representan puntos de datos en el espacio original de características. Cada kernel introduce parámetros que controlan la flexibilidad del modelo:

- $\gamma > 0$: controla la influencia local de los puntos (utilizado en el kernel RBF y el sigmoide).
- $r \in \mathbb{R}$: término independiente o sesgo (usado en los kernels polinómico y sigmoide).
- $d' \in \mathbb{N}$: grado del polinomio en el kernel polinómico.

Kernel Lineal:

$$K(\boldsymbol{a}, \boldsymbol{b}) = \langle \boldsymbol{a}, \boldsymbol{b} \rangle. \tag{2.28}$$

Kernel Radial (RBF o Gaussiano):

$$K(\boldsymbol{a}, \boldsymbol{b}) = \exp\left(-\gamma \|\boldsymbol{a} - \boldsymbol{b}\|^2\right). \tag{2.29}$$

Kernel Polinómico:

$$K(\boldsymbol{a}, \boldsymbol{b}) = (\langle \boldsymbol{a}, \boldsymbol{b} \rangle + r)^d. \tag{2.30}$$

Kernel Sigmoide:

$$K(\boldsymbol{a}, \boldsymbol{b}) = \tanh(\gamma \langle \boldsymbol{a}, \boldsymbol{b} \rangle + r). \tag{2.31}$$

Hiperparámetros

- **Parámetro de Regularización (C):** Controla el equilibrio entre maximizar el margen y permitir errores de clasificación. Valores altos priorizan clasificar correctamente los datos de entrenamiento; valores bajos generan un margen más amplio y toleran mayor error, donde $C > 0$.
- **Tipo de Kernel (`kernel`):** Determina la forma de la frontera de decisión. Scikit-learn permite: `linear`, `poly`, `rbf`, `sigmoid` (y `precomputed`).
- **Parámetros del Kernel:** Dependiendo del tipo elegido, scikit-learn utiliza:
  - γ: controla la influencia de los puntos en kernels `rbf`, `poly` y `sigmoid`, $\gamma > 0$. Es equivalente a $\frac{1}{2\sigma^2}$ en formulaciones gaussianas, aunque scikit-learn nunca usa σ explícitamente.
  - d (`degree`): grado del polinomio cuando el kernel es `poly`, $d \geq 1$.
  - r (`coef0`): coeficiente independiente usado por los kernels `poly` y `sigmoid`.

Capítulo 3

Métricas de Rendimiento Utilizadas

El objetivo de este trabajo es evaluar el desempeño del calificador de cada modelo de Aprendizaje Automático. Para alcanzarlo, se utilizan **métricas de rendimiento** que permiten cuantificar la capacidad del algoritmo de clasificar, ergo son herramienta que utilizamos para saber qué tan bien predice un modelo a una clase o qué 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 obtener las métricas y entrenar el algoritmo se utiliza la estrategia de **Validación Cruzada K-fold**. En este método, el conjunto de datos se divide en K grupos $\{G_1, G_2, \ldots, G_K\}$ (o *pliegues*) de igual tamaño. En cada iteración, un grupo G_i $(1 \leq i \leq K)$ se utiliza como conjunto de prueba, mientras que los $K-1$ restantes se emplean para el entrenamiento. Cada grupo actúa como conjunto de prueba exactamente una vez.

El valor final estimado de la métrica M, denotado por $\widehat{M}$, es el promedio de los valores obtenidos en cada iteración, es decir,

$$\widehat{M} = \frac{1}{K} \sum_{i=1}^{K} M_i, \tag{3.1}$$

donde M_i representa el valor de la métrica de evaluación cuando G_i se utiliza como conjunto de prueba, para $i = 1, 2, \ldots, K$.

Dentro de este trabajo no sólo se evalúan distintos modelos, sino que se utilizan distintos *datasets* para lograrlos. A continuación se señalan las métricas que se utilizan en cada caso y se pueden apreciar algunas diferencias, leves, pero diferencias en sí.

3.1 Métricas para el Caso Binario

Matriz de Confusión

Una matriz de confusión, que se puede observar en la Tabla 3.1, es una forma simple de saber de qué forma está 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.

Tabla 3.1: Matriz de confusión.

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

A partir de la matriz de confusión se calculan varias métricas, algunas de las cuales se presentan a continuación.

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 se predice una clase muy bien, que tiene una distribución mucho mayor que la otra, mientras que la de menor aparición casi no la predice. En términos de la matriz de confusión,

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{\text{Total}}. \tag{3.2}$$

En términos del conjunto de predicciones y valores verdaderos, se tiene que n: representa la cantidad total de ejemplos en el *dataset*, mientras que $\hat{y}_l$ es el valor predicho del l-ésimo ejemplo, e y_l es el valor verdadero correspondiente, en cual caso si $\hat{y}_l$ es igual al valor obtenido. La expresión $1(\hat{y}_l = y_l)$ corresponde a la función indicadora, que toma el valor 1 si la predicción del ejemplo l es correcta y 0 en caso contrario. Al sumar estos valores sobre todas las muestras, se obtiene el número total de aciertos del modelo.

$$\texttt{Accuracy}\,(y, \hat{y}) = \frac{1}{n} \sum_{l=0}^{n-1} \mathbb{I}(\hat{y}_l = y_l), \tag{3.3}$$

donde la **función indicadora** se define como:

$$\mathbb{I}(\hat{y}_l = y_l) = \begin{cases} 1, & \text{si } \hat{y}_l = y_l, \\ 0, & \text{en caso contrario.} \end{cases}$$

Por lo tanto, se puede simplificar como la siguiente fórmula:

$$Accuracy = \frac{\text{Número de predicciones correctas}}{\text{Número total de muestras}}. \tag{3.4}$$

Precision

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

$$Precision = \frac{TP}{TP + FP}. \tag{3.5}$$

Recall

El *Recall* o también conocido como Sensibilidad o Tasa de Verdaderos Positivos (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 al *Recall* de la siguiente manera:

$$\text{Recall} = \text{TPR} = \frac{TP}{TP + FN} = \frac{TP}{P}. \tag{3.6}$$

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, donde $\beta > 0$ controla la importancia relativa entre la precisión y el recall. Un clasificador perfecto tiene un valor $F1 = 1$. La fórmula general es:

$$F_\beta = \frac{(1 + \beta^2)\, precision \times recall}{\beta^2 precision + recall}, \tag{3.7}$$

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

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \tag{3.8}$$

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

$$F1 = \frac{2\,TP}{2\,TP + FP + FN}. \tag{3.9}$$

Área Bajo la Curva ROC (AUC)

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

La Curva ROC es un gráfico que ilustra el rendimiento de un clasificador binario a media que se varía 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.

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 $0{,}5 < AUC \leq 1$.

3.2 Métricas para caso Multiclase

En este caso se utiliza el método *"weighted"*, el cual tiene en cuenta el desequilibrio de clases calculando el promedio de las métricas binarias por clase, ponderadas según su frecuencia en el conjunto de datos reales.

Sea $\{1, 2, \ldots, L\}$ el conjunto de etiquetas o clases. Para cada clase l, se define $|C_l|$ como la cantidad de muestras reales pertenecientes a dicha clase (las barras verticales representan la **cardinalidad** del conjunto de ejemplos con etiqueta l, es decir la cantidad de muestras que tiene esa clase). Asimismo, M_l es el valor de la métrica binaria correspondiente a la clase l.

La métrica ponderada, $\widehat{M}_{\text{weighted}}$, se define como:

$$\widehat{M}_{\text{weighted}} = \frac{1}{\sum_{l \in L} |C_l|} \sum_{l \in L} |C_l| \cdot M_l. \tag{3.10}$$

De esta forma, las clases con mayor cantidad de muestras tienen una contribución proporcionalmente mayor en el valor final de la métrica evaluada.

Matriz de Confusión Multiclase

La matriz de confusión multiclase, que se puede ver en la Tabla 3.2, 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 $i, j \in \{1, 2, \ldots, L\}$.

Para cada clase l se definen los valores que antes habíamos utilizados para la matriz de confusión del caso binario, en donde Z es el total de instancias o la sumatoria de todas las celdas:

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

Tabla 3.2: Matriz de confusión multiclase.

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

Precision

La *Precision* por clase l mide la proporción de muestras clasificadas como positivas que realmente pertenecen a la clase l, donde $l \in \{1, 2, \ldots, L\}$. En términos más simple, utilizando la Matriz de Confusión:

$$Precision_l = \frac{TP_l}{TP_l + FP_l}. \tag{3.11}$$

Recall

El *Recall* por clase l, donde $l \in \{1, 2, \ldots, L\}$, mide la proporción de muestras verdaderamente positivas de la clase l que fueron correctamente identificadas. En términos más simple, utilizando la Matriz de Confusión:

$$Recall_l = \frac{TP_l}{TP_l + FN_l}. \tag{3.12}$$

F1-measure

El valor global ponderado se obtiene aplicando la fórmula de M_{weighted} sobre los F_l, para $\beta = 1$:

$$F_{\text{weighted}} = \sum_{l \in L} w_l \, F_l, \quad \text{con } w_l = \frac{|C_l|}{\sum_{l=1}^{L} |C_l|}. \tag{3.13}$$

Área Bajo la Curva ROC (AUC)

Para extender la métrica *AUC* a clasificación multiclase se emplea el enfoque **One-vs-Rest (OVR)**. Para cada clase l (donde $l \in \{1, 2, \ldots, L\}$), se considera la clase l como positiva y el resto como negativas; luego, se calcula el valor correspondiente AUC_l a partir de la curva ROC del problema binario generado para esa clase. Finalmente, el *AUC* multiclase se obtiene como un promedio ponderado por el soporte de cada clase.

$$AUC_{\text{OVR, weighted}} = \sum_{l \in L} w_l \, AUC_l \quad \text{con } w_l = \frac{|C_l|}{\sum_{l=1}^{L} |C_l|}. \tag{3.14}$$

Importancia de la característica

Sea un modelo predictivo $\mathcal{M}$ entrenado sobre un conjunto de datos X, y sea M la métrica de referencia del modelo sobre los datos originales. El procedimiento se detalla a continuación:

1. Para cada atributo j del conjunto de datos:

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

      1) 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.
      2) Calcular la métrica M del modelo sobre los datos permutados:

$$M_{k,j} = \text{valor de la métrica } M \text{ del modelo } \mathcal{M} \text{ con } X^{(k,j)}. \tag{3.15}$$

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

$$I_j = M - \frac{1}{K}\sum_{k=1}^{K} M_{k,j}. \tag{3.16}$$

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.

Capítulo 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. Se busca 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 (*AUC*), entre otras.

Durante la fase experimental se exploraron distintas estrategias de balanceo de clases, dado que varios de los conjuntos presentan 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 tienden a perder capacidad de generalización, mostrando un descenso notable en las métricas de validación, aunque se sostiene la mejor configuración con mejor valor de métricas. Por este motivo, se opta finalmente por mantener la distribución original y aplicar técnicas de regularización y ajuste de hiperparámetros para mitigar el sesgo hacia la clase dominante.

4.1 Dataset Binario

En esta sección se presentan los resultados obtenidos al aplicar los distintos modelos de aprendizaje automático al **dataset binario**. En este caso, la tarea de clasificación consiste en distinguir entre dos clases posibles, lo que típicamente permite a los algoritmos aprender fronteras de decisión más simples en comparación con escenarios multiclase.

Regresión Logística

RL obtuvo un correcto desempeño en la clasificación binaria, siempre tomando un **buen resultado** cuando cualquier métrica por lo menos supere el valor de 0.80.

Los resultados cuantitativos se resumen en la Tabla 4.2, en la cual se muestra la mejor configuración obtenida de los hiperparámetros de la Tabla 4.1. El modelo logra un *Accuracy* y un F1-Score de 0.8445, junto con un *AUC* de 0.9050, lo que refleja una buena capacidad discriminatoria, pero siendo datos médicos se requieren incluso mejores valores.

Tabla 4.1: Grilla 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]

Tabla 4.2: Resultados finales del modelo de Regresión Logística.

Configuración	***Accuracy***	**Recall**	**F1-Score**	*AUC*	**Tiempo (s)**	**Precisión**
$C = 1$ Penalty = l1 Solver = saga Multiclass = ovr	0.8485	0.8485	0.8481	0.9050	0.13	0.8495

El tiempo de entrenamiento fue de apenas 0.13 segundos, lo que lo convierte en una opción eficiente para este tipo de problema, teniendo en cuenta el tamaño pequeño del *dataset*.

Máquinas de Soporte Vectorial

SVM obtuvo un desempeño sólido en la clasificación binaria, mostrando un equilibrio adecuado entre precisión y generalización. Demostrando la solidez habitual con que es descripto este método.

Los resultados cuantitativos finales se resumen en la Tabla 4.4. El modelo logra un *Accuracy* y un F1-Score de 0.8616, junto con un *AUC* de 0.9232, lo que refleja la mejor capacidad de clasificación hasta ahora.

El tiempo de entrenamiento fue de apenas 0.60 segundos, siendo un tiempo acotado, pero costoso si vemos que el tamaño del *dataset* es bastante pequeño, refleja un costo de evaluación más grande que con RL.

Tabla 4.3: Grilla 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

NB obtuvo un desempeño sólido en la tarea de clasificación binaria, especialmente considerando su supuesto fuerte de independencia condicional entre atributos.Si bien este supuesto rara vez se cumple estrictamente en datos reales, el modelo suele mantener un rendimiento competitivo. En este trabajo, evaluamos su desempeño utilizando la métricas distitas métricas, y consideramos que un valor superior a 0.80 constituye un nivel de desempeño adecuado para el tipo de problema abordado.

Tabla 4.4: Resultados finales del SVM.

Configuración	***Accuracy***	**Recall**	**F1-Score**	*AUC*	**Tiempo (s)**	**Precisión**
$C = 1$ Kernel = rbf Gamma = 0.1	0.8616	0.8616	0.8609	0.9232	0.60	0.8628

La mejor configuración se alcanzó con cualquier suavizado, no hubo diferencias. Los resultados cuantitativos finales se resumen en la Tabla 4.6. El modelo logra un *Accuracy* y un F1-Score de 0.8420, junto con un *AUC* de 0.9138, lo que refleja una muy buena capacidad clasificadora siendo un método tan eficaz, además de ser bastante similar con los resultados de los modelos a los cuales se compara.

Tabla 4.5: Grilla de hiperparámetros - Naive Bayes Gaussiano (binario).

Hiperparámetro	**Valores evaluados**
Suavizado	$\{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}\}$

Tabla 4.6: Resultados finales del Naive Bayes Gaussiano.

Configuración	***Accuracy***	**Recall**	**F1-Score**	*AUC*	**Tiempo (s)**	**Precisión**
var_smoothing = Cualquiera	0.8420	0.8420	0.8420	0.9138	0.10	0.8433

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

Random Forest

RF demostró un desempeño bastante sólido en la clasificación binaria, mostrando alta capacidad de generalización para registros no vistos durante el entrenamiento.

Los resultados cuantitativos finales se resumen en la Tabla 4.8, donde el modelo logra un *Accuracy* y un F1-Score de 0.8780, junto con un *AUC* de 0.9315, lo que refleja una excelente capacidad de clasificación, pero evidentemente la tarea de creación de los árboles y del voto mayoritario lo vuelven un método bastante costoso aunque efectivo.

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

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 RF presenta mejor capacidad de generalización, mientras que los demás modelos muestran un desempeño bastante competitivo. Esto no quiere decir que sea el mejor modelo, sino que es el que mejor clasifica este *dataset*, pero si se quieren resultados sólidos con menor tiempo de cálculo, entonces el NB es una buena solución. Podemos afirmar que todos los modelos tuvieron desempeños correctos.

Tabla 4.7: Grilla 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]

Tabla 4.8: Resultados finales del Random Forest.

Configuración	***Accuracy***	**Recall**	**F1-Score**	*AUC*	**Tiempo (s)**	**Precisión**
Criterion = entropy Max Depth = 7 Min Samples Split = 5 Min samples Leaf = 1 Max Features = sqrt	0.8780	0.8780	0.8775	0.9315	1.38	0.8731

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, 4.11 y 4.12, correspondientes a los modelos RL, NB, SVM y RF, respectivamente.

Tabla 4.9: 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

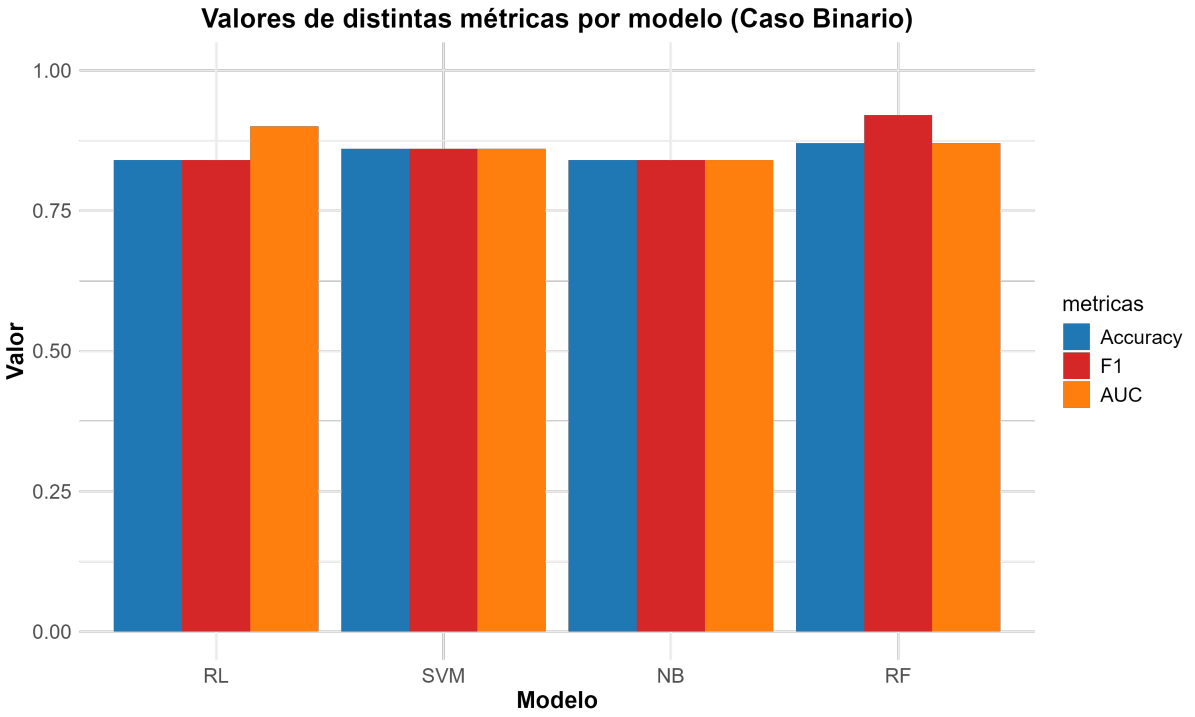

Figura 4.1: Comparación de desempeño de los mejores modelos (Binario).

Tabla 4.10: 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

Tabla 4.11: 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 4.12: 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

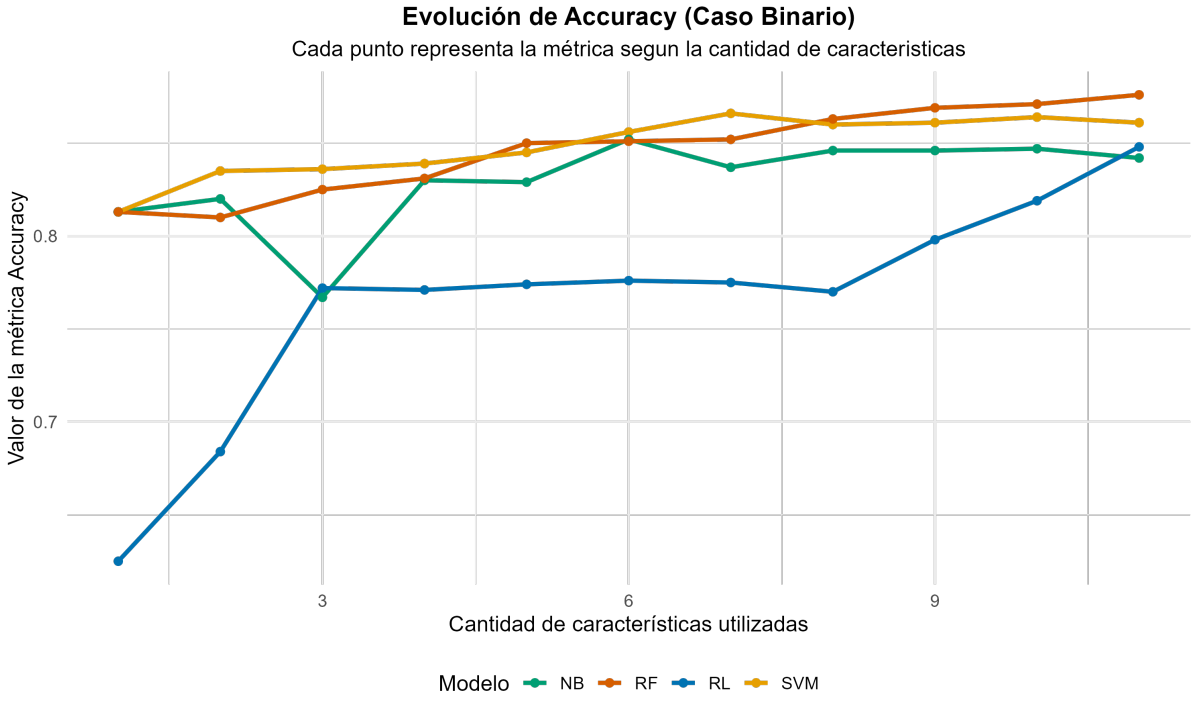

Figura 4.2: Evolución de *Accuracy* (Binario).

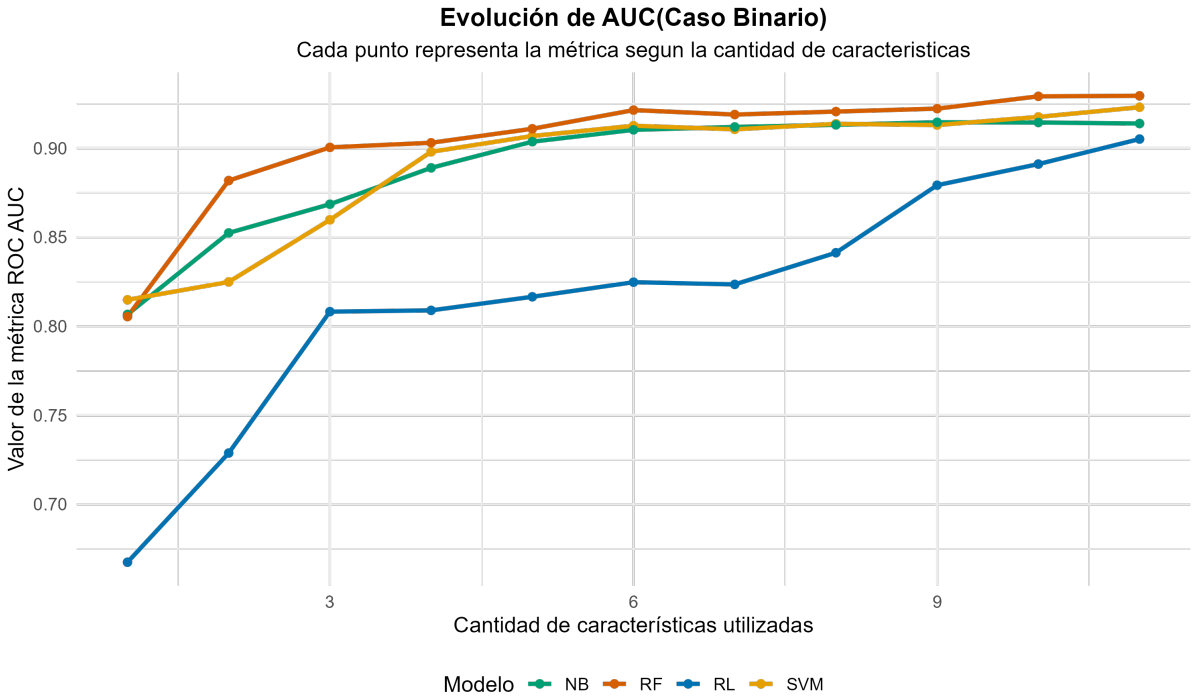

Figura 4.3: Evolución de *AUC* (Binario).

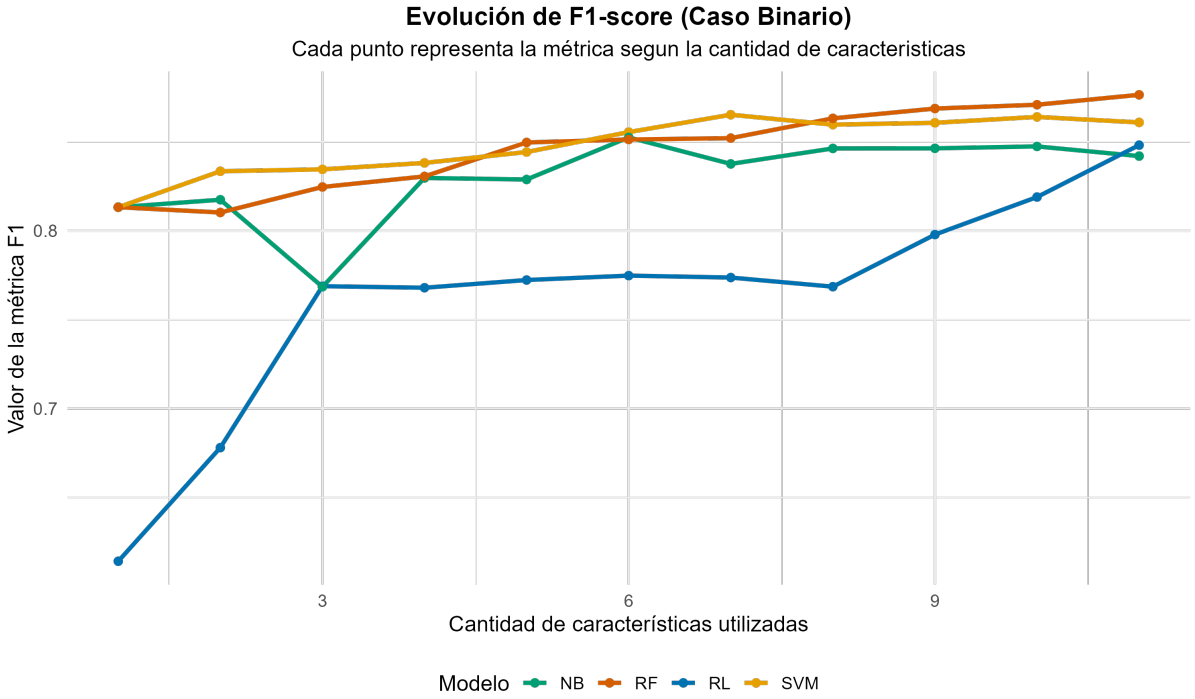

Figura 4.4: Evolución de F1-Score (Binario).

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 permi-

te 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. Además, como se puede observar, tienen valores aceptables para pocas características.

4.2 Dataset Multiclase

En esta sección se presentan los resultados obtenidos al aplicar los distintos modelos de aprendizaje automático al **dataset multiclase**. A diferencia del problema binario, este conjunto de datos requiere que los algoritmos distingan entre varias categorías posibles, lo cual introduce desafíos adicionales como la separación más compleja entre clases, mayor variabilidad interna y posibles desbalances entre categorías.

Regresión Logística

RL obtuvo un muy buen desempeño en la clasificación multiclase, mostrando un equilibrio adecuado entre precisión y generalización.

Los resultados cuantitativos finales se resumen en la Tabla 4.14, junto con la configuración de hiperparámetros que logro esos resultados. El modelo logra un *Accuracy* y un F1-Score de 0.8978 respectivamente, junto con un *AUC* de 0.9644, lo que refleja una buena capacidad discriminatoria, logrando un buen resultado al incremento de datos y de dificultad al existir más de dos clases.

El tiempo de entrenamiento fue de apenas 0.30 segundos, lo que lo convierte en una opción eficiente para este tipo de problema, teniendo en cuanta que se mantiene en rangos acotados, incluso con más datos que en el caso binario.

Tabla 4.13: Grilla 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]

Tabla 4.14: Resultados finales del modelo de Regresión Logística.

Configuración	***Accuracy***	**Recall**	**F1-Score**	*AUC*	**Tiempo (s)**	**Precisión**
$C = 1$ Penalty = None Solver = newton-cg Multiclass = ovr	0.8978	0.8978	0.8953	0.9644	0.30	0.8945

Máquinas de Soporte Vecotorial

SVM obtuvo un desempeño bastante sólido en la clasificación multiclase, mostrando unos grandes valores en sus métricas.

Los resultados cuantitativos finales se resumen en la Tabla 4.16, junto con la mejor configuración que logro las mejoras métricas. El modelo logra un *Accuracy* y un F1-Score de 0.9082, junto con un *AUC* de 0.9556, lo que refleja una muy buena capacidad de clasificación, logrando ser un método útil para *datasets* acotados y grandes, como a casos binarios y multiclase. El tiempo de entrenamiento fue de apenas 1.42 segundos, mostrando una clara demora a cuantos más datos deben de procesar, si comparamos como sube el tiempo de ejecución con el caso binario.

Tabla 4.15: Grilla 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]

Tabla 4.16: Resultados finales del SVM.

Configuración	***Accuracy***	**Recall**	**F1-Score**	*AUC*	**Tiempo (s)**	**Precisión**
$C = 0{,}1$ Kernel = poly Gamma = 1 Degree = 2	0.9082	0.9082	0.9064	0.9556	1.45	0.9057

Naive Bayes

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

Tabla 4.17: Grilla de hiperparámetros - Naive Bayes Gaussiano (multiclase).

Hiperparámetro	**Valores evaluados**
Suavizado	$\{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}\}$

Tabla 4.18: Resultados finales del Naive Bayes Gaussiano.

Configuración	***Accuracy***	**Recall**	**F1-Score**	*AUC*	**Tiempo (s)**	**Precisión**
var_smoothing = Cualquiera	0.8208	0.8208	0.8366	0.9112	0.11	0.8747

Los resultados cuantitativos finales se resumen en la Tabla 4.18. El modelo logra un *Accuracy* y un F1-Score de 0.8208, junto con un *AUC* de 0.91, lo que refleja una buena capacidad clasificadora, pero

lo suficientemente menor a los modelos con los cuales se compara, para decidir que no es útil en cuanto a tener buenos valores de métricas a la hora de resolver este problema. Es evidente que se mantuvo en valores similares al caso binario, sin tener caídas aún cuando se incrementa el tamaño del *dataset* o se dificulta el mismo con un caso multiclase, pero tampoco muestra mejora como el resto de modelos.

El tiempo de entrenamiento fue de apenas 0.11 segundos, siendo su mayor fortaleza, incluso con más datos el modelo se mantiene en valores parecidos de tiempo de ejecución.

Random Forest

RF demostró un desempeño claramente superador en la clasificación multiclase, mostrando alta capacidad de generalización incluso para un *dataset* más grande y con caso multiclase.

Los resultados cuantitativos finales se resumen en la Tabla 4.20, junto con la mejor configuración con la cual se obtuvieron estos resultados. El modelo logra un *Accuracy* y un F1-Score de 0.9432, junto con un *AUC* de 0.9868, lo que refleja una buena excelente capacidad clasificadora. Se entienden que los árboles de decisión no se ven afectados por casos multiclase y que la presencia de más cantidad de registros los enriquecen para realizar mejor clasificaciones. El tiempo de entrenamiento fue de 2.24 segundos, mostrando como estos modelos incrementan su tiempo de ejecución a mayores datos requieren procesar.

Tabla 4.19: Grilla 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]

Tabla 4.20: Resultados finales del Random Forest.

Configuración	***Accuracy***	**Recall**	**F1-Score**	*AUC*	**Tiempo (s)**	**Precisión**
Criterion = gini Max Depth = None Min Samples Split = 2 Min samples Leaf = 1 Max Features = sqrt	0.9432	0.9432	0.9412	0.9868	2.24	0.9417

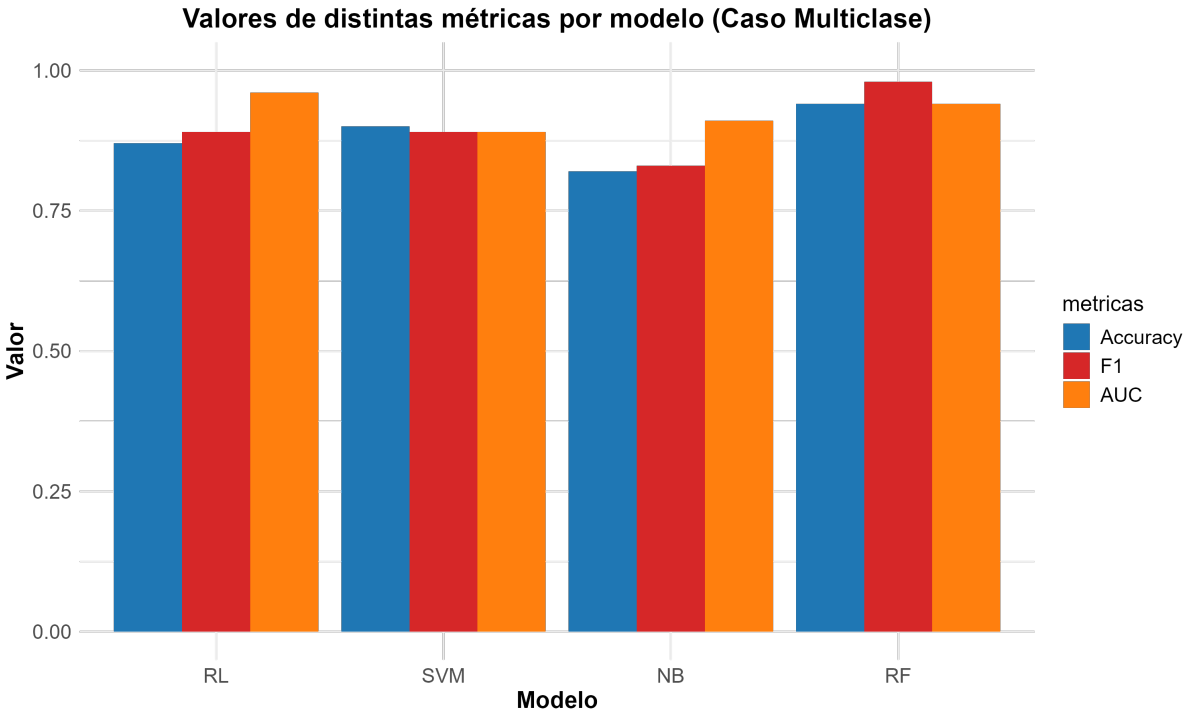

Figura 4.5: Comparación de desempeño de los mejores modelos (Multiclase).

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 *datasets*, el modelo de Random Forest presenta mejor capacidad de generalización, mientras que los demás modelos muestran un desempeño bastante competitivo.

Importancia de las Características

Los resultados obtenidos se presentan en las Tablas 4.24, 4.21, 4.23 y 4.22 correspondientes a los modelos RF, RL, NB y SVM.
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. Se ven como **RF** muestra resultados muy prometedores incluso con pocas métricas, siendo claramente superior en los valores de métricas incluso con pocos atributos.

Tabla 4.21: 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

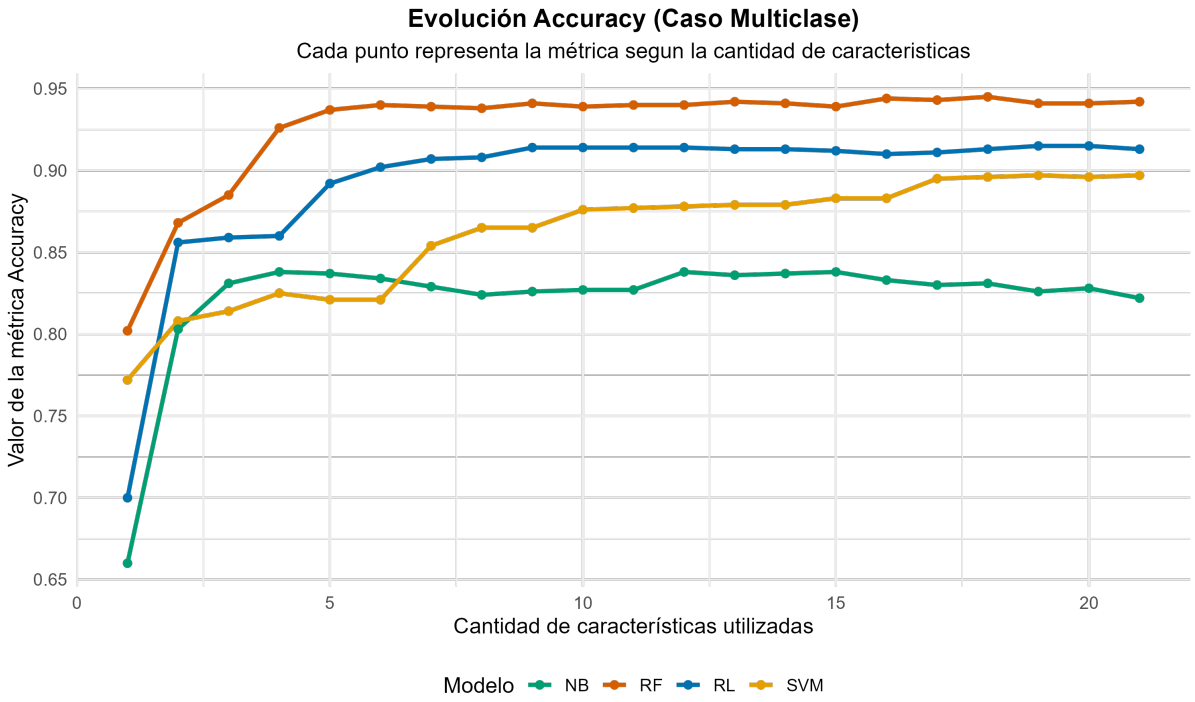

Figura 4.6: Evolución de *Accuracy* (Multiclase).

Tabla 4.22: 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

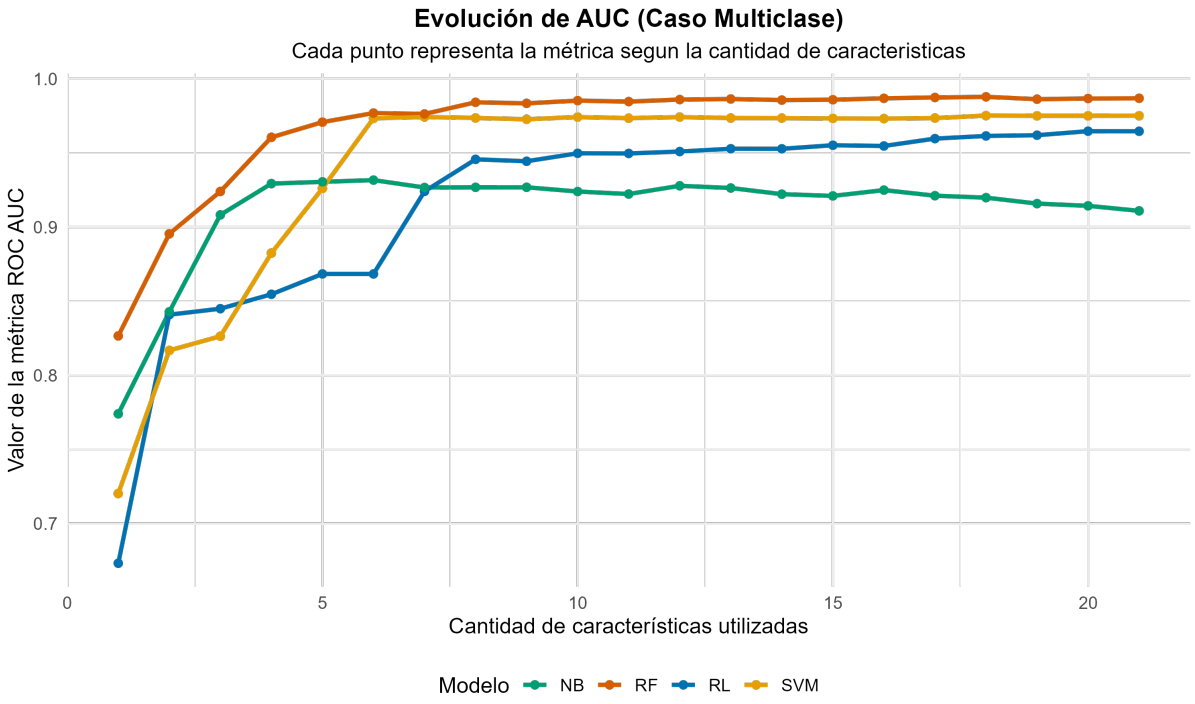

Figura 4.7: Evolución de *AUC* (Multiclase).

Tabla 4.23: 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

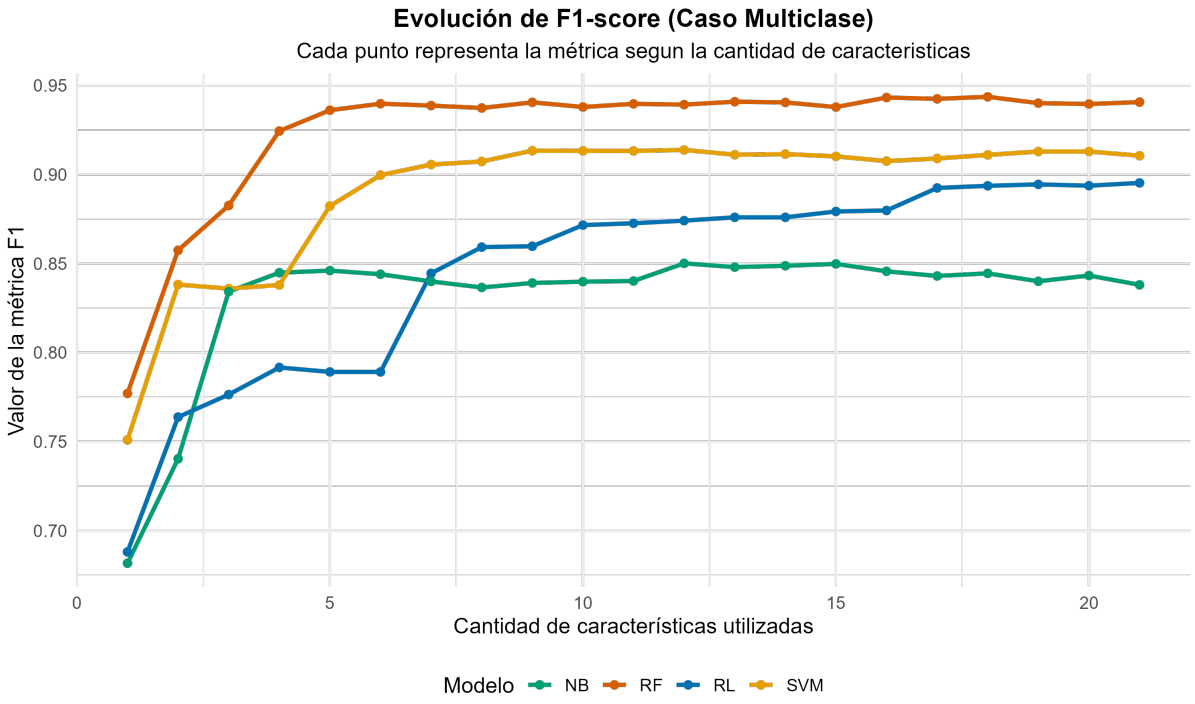

Figura 4.8: Evolución de F1-score (Multiclase).

Tabla 4.24: 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

Capítulo 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 árboles que permiten una representación mejor.

El modelo de **Máquinas de Soporte Vectorial** 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 caso binario, variables como *ST_Slope*: este segmento del ECG se analiza porque cambia cuando hay isquemia, es decir, cuando el corazón recibe menos sangre de la necesaria; *ChestPainType*: el tipo de dolor en el pecho, asociado a isquemia cardíaca; y *Oldpeak*: componente utilizado para estimar el riesgo de isquemia o infarto de miocardio, fueron recurrentemente relevantes.

En el caso multiclase destacaron $ASTV$: porcentaje de tiempo con variabilidad anormal a corto plazo; $ALTV$: porcentaje de tiempo con variabilidad anormal a largo plazo; y $MSTV$: valor medio de la variabilidad a largo plazo.

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.

Bibliografía

[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.