IDENTIFICACIÓN DE NIVELES DE RIESGOS EN SALUD PARA LA GESTIÓN DE AFILIADOS MEDIANTE APRENDIZAJE AUTOMÁTICO

Adriana Janeth Ávila Reina, Código 8970490 y David Orlando Aguilar Ramírez, Código 8992487

Proyecto Aplicado para optar al título de
Magister en Ciencia de Datos

Director
Hernán Darío Vargas Cardona

FACULTAD DE INGENIERÍA Y CIENCIAS
MAESTRÍA EN CIENCIA DE DATOS
SANTIAGO DE CALI, MAYO 09 DE 2025

TABLA DE CONTENIDO

Lista de Tablas

Lista de Ilustraciones

INTRODUCCIÓN

El presente trabajo es una propuesta para la identificación de niveles de riesgo en salud para la gestión de afiliados mediante técnicas de aprendizaje automático, a partir del trabajo a desarrollar por los autores, trabajadores del área de salud e innovación dentro de la organización y estudiantes de la Maestría en Ciencia de Datos de la Pontificia Universidad Javeriana de Cali. Utilizando una base de datos estructurada de la empresa Keralty, se aplicaron modelos de aprendizaje supervisado para clasificar los niveles de riesgo en salud de los pacientes. Además, se evaluó el desempeño de estos modelos utilizando métricas como exactitud, sensibilidad y especificidad, con el objetivo de seleccionar el modelo más efectivo.

Este estudio busca ofrecer una propuesta de herramienta que facilite una gestión más eficiente y precisa del riesgo en salud dentro de la organización. El contexto del problema radica en la necesidad de mejorar la gestión del riesgo en salud en sistemas de aseguramiento y prestación de servicios de salud. En Colombia, la empresa Keralty enfrenta el desafío de optimizar la estratificación del riesgo de sus pacientes para identificar aquellos en mayor riesgo de deterioro de salud, permitiendo la implementación de intervenciones preventivas que puedan reducir complicaciones y costos asociados. Actualmente, Keralty utiliza un modelo basado en reglas y criterios predefinidos por expertos, lo cual ha permitido una primera aproximación a la gestión del riesgo. Sin embargo, este modelo no aprovecha las capacidades avanzadas del aprendizaje automático para manejar y analizar grandes volúmenes de datos.

La problemática actual es vigente y relevante debido al crecimiento continuo de la población afiliada y la necesidad de personalizar y optimizar la atención en salud. Los modelos tradicionales basados en reglas tienen limitaciones en su capacidad para adaptarse a nuevos datos y para identificar patrones complejos en los datos de salud de los pacientes. El propósito de este proyecto es desarrollar e implementar modelos de aprendizaje supervisado que puedan proporcionar una clasificación más exacta de los niveles de riesgo en salud. Estos modelos fueron entrenados y validados utilizando una base de datos

estructurada de Keralty, y su desempeño fue evaluado mediante métricas que permiten seleccionar modelos para ser propuestos como una herramienta de gestión de riesgos en salud en la empresa.

Por lo anterior, el presente proyecto aplicado propone una innovación metodológica en la gestión de riesgos en salud, alineada con las necesidades actuales de Keralty y con las tendencias globales en el uso de grandes volúmenes de datos y aprendizaje automático en salud. Con este proyecto, se busca proponer una visión complementaria a la estratificación existente, no obstante, este proyecto es una propuesta teórica-práctica que no llegará a la fase de despliegue, quedando su implementación final sujeta a la evaluación y aprobación de los tomadores de decisión de la organización.

1. DEFINICIÓN DEL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

Actualmente la organización de salud en Colombia Keralty, un grupo de empresas privadas de salud con presencia en diferentes países [1], utiliza un sistema de estratificación para la gestión de riesgos en salud que clasifica a los pacientes basándose en aspectos como la cronicidad de las enfermedades, la edad, el control de la enfermedad, el costo asociado, consultas con especialistas, uso de medicamentos, condicionantes en salud, egresos hospitalarios y urgencias. Este modelo responde a una necesidad misional de la organización de hacer gestión integral y multidimensional de salud sobre sus usuarios, además de dar respuesta a normativas locales de creación de herramientas para la gestión del riesgo en salud[1]. Este modelo se encuentra en una etapa inicial, fundamentado en reglas y criterios predefinido de expertos, con quienes se ha desarrollado una primera versión.

El principal objetivo de estos modelos de estratificación es realizar identificación de pacientes que mediante acciones preventivas podría limitarse el deterioro de su estado de salud por causas evitables, además, de reducir la generación de costos adicionales a la atención que podrían no haberse incurrido si el paciente no deteriora su estado de salud.

Aunque el modelo de estratificación actual proporciona una base para la identificación de pacientes, existe la necesidad de optimizar esta clasificación mediante la implementación de técnicas de aprendizaje automático que permita manejar y procesar eficientemente los grandes volúmenes de datos con los que cuenta la organización, los cuales se actualizan permanentemente. Para ello se pretende recurrir a los algoritmos de aprendizaje supervisado por cuanto son una técnica que utiliza un conjunto de datos de entrenamiento

[1] Para el caso colombiano, por ejemplo, en lo relacionado al Plan Decenal de Salud Pública 2022 – 2031, respecto a responsabilidades de las entidades promotoras de salud, según la resolución 1035 de 2022, estas deben “7. Desarrollar o fortalecer los procesos, herramientas e instrumentos para asegurar el seguimiento de las cohortes de usuarios para la gestión del riesgo” [4]

con entradas y salidas correctas para enseñar a los modelos a generar los resultados deseados, minimizando el error [2].

Estos clasificadores supervisados pueden detectar patrones y relaciones en los datos que no son evidentes para los expertos, proporcionando una comprensión más profunda y detallada de los factores de riesgo, además, estos modelos pueden actualizarse y entrenarse continuamente con nuevos datos, permitiendo que se adapten rápidamente a cambios en los perfiles de los pacientes y en las condiciones de salud. Esto permitiría un mejor uso de los datos demográficos, clínicos y de utilización de servicios de salud ya existentes para refinar la estratificación actual.

Con el desarrollo de este ejercicio académico, se pretende dar una propuesta de insumo a la organización (donde los autores de este proyecto de grado trabajan), para una gestión más eficiente de los riesgos en salud, permitiendo a los tomadores de decisión en la organización implementar estrategias más efectivas y centradas en los pacientes.

1.2 FORMULACIÓN DEL PROBLEMA

Dado lo anterior, se plantea la siguiente pregunta de investigación: ¿Cómo se puede automatizar una estratificación existente para la gestión de riesgos en salud de los afiliados de una aseguradora en Colombia usando modelos de aprendizaje automático supervisado?

1.2.1 SISTEMATIZACIÓN

En adición se plantean las siguientes preguntas específicas:

- ¿Cuáles datos y características disponibles se requieren para la clasificación del riesgo de pacientes?
- ¿Qué técnicas de modelado supervisado son adecuados para agrupar a los afiliados en diferentes niveles de riesgo?
- ¿Cuáles métricas permiten evaluar el mejor modelo que reagrupa los pacientes dentro de una jerarquía de gestión de riesgo?

2. OBJETIVOS DEL PROYECTO

2.1 OBJETIVO GENERAL

Identificar niveles de riesgo en salud de pacientes mediante la aplicación de técnicas de aprendizaje automático supervisado que pueda ser implementado como herramienta de gestión en la empresa Keralty.

2.2 OBJETIVOS ESPECÍFICOS

- Acondicionar una base de datos estructurada de la empresa Keralty mediante la gestión del permiso de uso, etiquetado, y anonimizado.
- Implementar modelos de aprendizaje supervisado que permitan la clasificación de niveles de riesgo en salud de pacientes a partir de la información disponible.
- Evaluar el desempeño de diferentes clasificadores supervisados utilizando métricas para clasificación como exactitud, sensibilidad y especificidad para seleccionar el mejor modelo.

3. MARCO TEÓRICO Y ANTECEDENTES

La medición del riesgo en salud poblacional constituye el fundamento para una gestión eficiente de las instituciones responsables de la organización y provisión de servicios sanitarios, dado que el perfil de riesgo determina las necesidades específicas de atención. En los modelos de gestión en salud, actores clave coordinan las redes de prestadores y proveedores para equilibrar la oferta de servicios con la demanda de los asegurados. Su rol estratégico comprende: (1) el diagnóstico de necesidades poblacionales, (2) la estructuración de una oferta adecuada -incluyendo la dotación de personal calificado- y (3) la garantía de acceso oportuno. En el sistema colombiano, esta función recae en las Entidades Promotoras de Salud (EPS) [3].

La medición del riesgo en salud de los afiliados requiere que las aseguradoras implementen sistemas integrales de gestión de información. Estos sistemas deben incluir procesos de: recopilación, organización y análisis de datos clínicos y demográficos, permitiendo identificar tanto el estado de salud como los factores de riesgo potenciales de la población asegurada. Esta capacidad analítica constituye la base para una gestión proactiva del riesgo sanitario [4].

En este contexto, el holding de servicios de salud Keralty ha implementado un sistema de estratificación de riesgos que clasifica a los pacientes mediante múltiples variables clínicas y socio-sanitarias: edad, presencia de enfermedades crónicas, grado de control, frecuencia de consultas especializadas, uso de medicamentos, determinantes sociales, tasas de hospitalización y utilización de servicios de urgencias.

Este modelo se alinea con la misión institucional de Keralty de ofrecer una gestión sanitaria integral y multidimensional, al tiempo que cumple con los requisitos normativos locales en materia de gestión del riesgo en salud [1].

Aunque el modelo de estratificación de riesgo actual permite la identificación, caracterización y estratificación de los pacientes según perfiles de riesgo establecidos [5],

existe la necesidad de optimizar esta clasificación. Para ello, es esencial implementar técnicas de aprendizaje automático que permitan manejar y procesar eficientemente los grandes volúmenes de datos disponibles en la organización.

Este proyecto pretende construir y aplicar algoritmos de aprendizaje supervisado para detectar patrones y relaciones en los datos que no son evidentes con el modelo de clasificación actual. Esto permitirá evolucionar el modelo de estratificación de riesgo, proporcionando una comprensión más profunda y detallada de los factores de riesgo. Además, con la capacidad de actualizarse y entrenarse continuamente con nuevos datos, el modelo podrá adaptarse rápidamente a los cambios en los perfiles de los pacientes y en las condiciones de salud, lo cual no es posible con la herramienta actualmente utilizada.

3.1 MARCO TEÓRICO

3.1.1 MEDICIÓN Y ESTRATIFICACIÓN DE RIESGO EN SALUD

La medición y estratificación de riesgo en salud son procesos fundamentales en la gestión sanitaria, orientados a identificar poblaciones y optimizar recursos. Se sustentan en el principio de que no todos los individuos requieren el mismo nivel de atención, y que una asignación eficiente de servicios mejora resultados en salud [6].

El concepto de riesgo en salud se refiere a la probabilidad de que ocurra un evento relacionado con la desmejora de un estado de salud en un individuo o en una población en un período de tiempo determinado [7]. Este evento puede incluir el desarrollo de una enfermedad, la progresión de una condición preexistente o la aparición de complicaciones que afecten la calidad de vida o la mortalidad.

El análisis de riesgo en salud es una herramienta clave para la prevención y gestión de enfermedades, que permite, identificar poblaciones vulnerables que pueden beneficiarse de intervenciones preventivas [8] , priorizar a los pacientes en función de su nivel de riesgo y gestionar los recursos optimizando su asignación [9].

Para caracterizar las condiciones de salud de los pacientes, se utiliza la noción de valor en salud, planteada por Michael Porter, por la cual la medición permanente de los resultados de salud y los costos se debe centrar en alcanzar mejoras en el bienestar y calidad de vida de las personas [11].

Para ello, se requiere identificar cuáles son las condiciones que conjuntamente tiene un usuario y no solo una condición específica que atienda algún tipo de especialista; la medición de multimorbilidad ayuda a asociar su impacto en indicadores de salud de los pacientes, ya que permite identificar de forma holística las necesidades de atención y diseñar intervenciones más efectivas.

Un estudio en BMJ Open destaca que los instrumentos de medición de multimorbilidad predicen resultados como la mortalidad, la salud mental y la calidad de vida [11]. Además, una organización de salud puede modificar sus modelos de negociación con grupos de interés, como sus proveedores, al considerar el riesgo asociado a sus usuarios. Este enfoque permite asegurar que las primas y tarifas sean adecuados para cubrir los costos médicos, evitando discriminaciones contra usuarios con necesidades de salud elevadas y manteniendo un entorno competitivo en el mercado de seguros de salud [12].

Existen diversas herramientas diseñadas para la caracterización y clasificación del riesgo en poblaciones de pacientes, que se utilizan para mejorar la gestión clínica, la asignación de recursos y la toma de decisiones en salud.

Los Clinical Risk Groups (CRG), desarrollados por la empresa 3M, son un sistema de clasificación que agrupa a los pacientes según sus condiciones crónicas y episodios agudos, asignándolos a categorías de riesgo basadas en la gravedad de sus diagnósticos. No obstante, los CRG requieren información clínica detallada y codificación precisa, lo que puede limitar su precisión en poblaciones con datos clínicos incompletos [9].

Otra referencia en los modelos de riesgo es, los Adjusted Clinical Groups (ACG), desarrollados por la Universidad Johns Hopkins, son un sistema que clasifica a los pacientes según su carga de morbilidad utilizando datos de diagnóstico y servicios para predecir la

utilización de recursos y los costos futuros en salud. Este sistema es utilizado en la predicción de costos y necesidades de atención médica, ya que puede manejar grandes volúmenes de datos para la gestión de la salud poblacional y ofrece flexibilidad para su aplicación en diversos entornos clínicos y sistemas de salud. Sin embargo, los ACG requieren un alto nivel de calidad en la codificación de diagnósticos. [14]

3.1.2 MODELO KERALTY DE ESTRATIFICACIÓN DE RIESGO EN SALUD

Cómo se estableció, el objetivo de este trabajo es evolucionar a un sistema de medición de riesgo automatizado dentro de Keralty, el sistema de medición y estratificación de riesgo en salud que esta empresa ha desarrollado, se fundamenta en una matriz 6×6 que cruza dos dimensiones: el nivel de riesgo (desde "Bajo" hasta "Crítico") y el grado de gravedad, generando 36 categorías de clasificación. Cada afiliado es asignado a una celda específica mediante algoritmos que analizan datos clínicos (comorbilidades, complicaciones), patrones de uso de servicios y determinantes socio-sanitarios. Los riesgos ubicados en el cuadrante superior derecho (alto riesgo y gravedad) activan protocolos de intervención inmediata, mientras que aquellos en el cuadrante inferior izquierdo (bajo riesgo/gravedad) reciben seguimiento preventivo. [5].

La estratificación de los niveles de riesgo se fundamenta en cuatro variables clínicas clave: (1) presencia de enfermedades crónicas, (2) cantidad de comorbilidades, (3) grado de control clínico, y (4) edad del paciente. Para categorizar las condiciones crónicas, el sistema emplea un enfoque económico-clínico que clasifica las patologías en tres grupos según su impacto financiero: Enfermedades de costo superior, costo intermedio y costo inferior.

Esta clasificación permite establecer reglas de clasificación progresiva para los 6 niveles de riesgo. El nivel 1 corresponde a pacientes sin enfermedades crónicas y menores de 60 años, mientras que los niveles superiores (2-6) escalan según:

- Aumento en el número de comorbilidades.
- Mayor complejidad de las condiciones.

- Deficiente control clínico.

El sistema incorpora un factor correctivo; los pacientes con enfermedades controladas (evidenciado por indicadores clínicos documentados) son reclasificados a niveles de riesgo inferiores, reflejando el principio epidemiológico de que el control de la condición reduce la probabilidad de complicaciones.

La gravedad se estratifica mediante un índice que pondera: frecuencia de servicios utilizados (consultas, urgencias, hospitalizaciones), complejidad (UCI, reingresos), y factores psicosociales.

3.1.3 MODELOS DE APRENDIZAJE DE MÁQUINA SUPERVISADO

La presencia de varias fuentes de información y de grandes volúmenes de datos, permiten proponer nuevas alternativas que mejoren la forma en que se realiza.

El aprendizaje supervisado (*supervised learning*) es un enfoque de la inteligencia artificial y ciencia de datos donde un modelo algorítmico aprende patrones o relaciones entre variables de entrada (*features*) a variables de salida (*labels*) de un conjunto de datos previamente estructurados [15]. Se realiza a través de la minimización de la función de pérdida entre las predicciones del modelo y el dato real y sus aplicaciones son útiles en la predicción, generación de nuevo conocimiento o automatización de procesos.

El desarrollo de modelos de aprendizaje supervisado se fundamenta en la teoría del aprendizaje estadístico, donde el equilibrio entre sesgo (error por simplificaciones del modelo) y varianza (sensibilidad a variabilidad en los datos) es crítico para evitar el sobreajuste (*overfitting*), es decir, cuando el modelo es incapaz de generalizar [15]. Para medir el desempeño del modelo, se emplean métricas como[2]:

[2] TP: Verdaderos Positivos – casos correctamente clasificados como positivos (clase 1).
TN: Verdaderos Negativos – casos correctamente clasificados como negativos (clase 0).
FP: Falsos Positivos – casos negativos clasificados incorrectamente como positivos.
FN: Falsos Negativos – casos positivos clasificados incorrectamente como negativos.

- Exactitud, proporción de predicciones correctas:

$$Exactitud = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

- Precisión, capacidad de no clasificar falsos positivos:

$$Precisión = \frac{TP}{(TP + FP)}$$

- *Recall*, sensibilidad para detectar casos positivos o especificidad para los negativos:

$$Recall = \frac{TP}{(TP + FN)}$$

- *F1-score*, media armónica de precisión y *recall*:

$$F1 = 2\,\frac{(Precisión * Recall)}{(Precisión + Recall)}$$

- AUC - ROC, área bajo la curva que mide la discriminación entre clases:

$$AUC - ROC = área\ bajo\ la\ curva\ ROC\ (TPR\ vs\ FPR)$$

TPR (True Positive Rate): también llamado Recall o Sensibilidad; mide la capacidad para detectar correctamente los positivos (clase 1)
FPR (False Positive Rate): proporción de negativos incorrectamente clasificados como positivos
TNR (True Negative Rate): también llamado Especificidad; mide la capacidad para detectar correctamente los negativos (clase 0)
Nota: Sensibilidad se refiere a la detección de la clase positiva (1), mientras que Especificidad se refiere a la detección de la clase negativa (0). Son complementarias y relevantes según cuál sea la clase de mayor interés en el análisis.

Estas métricas se validan mediante técnicas *k-fold cross-validation*, donde los datos se dividen en k subconjuntos, entrenando en $k-1$ y testeando en 1, iterativamente, para reducir el sesgo de evaluación.

Los algoritmos aplicados para este trabajo son los siguientes:

Decision Trees:

Los árboles de decisión (*Decision Trees*, DT) son algoritmos de aprendizaje supervisado que modelan decisiones mediante una estructura jerárquica tipo árbol, donde cada nodo representa una condición sobre una característica, cada rama un posible resultado, y cada hoja una decisión final. Son ampliamente utilizados en clasificación y regresión por su simplicidad e interpretabilidad [16].

Su rendimiento depende de la configuración de sus hiperparámetros, los cuales controlan su estructura, capacidad predictiva y riesgo de sobreajuste.

La profundidad del árbol (*max_depth*), limita el número máximo de niveles (divisiones) en cada árbol. Una profundidad baja implica modelos simples, interpretables, pero pueden sub-ajustar, en tanto que una profundidad alta, captura patrones complejos, pero con riesgo de sobreajuste. La mayor profundidad exige más recursos de cómputo y calidad en los datos.

Bosque Aleatorio:

Los bosques aleatorios (*Random Forest*, RF), propuestos por Breiman [17], son un método de aprendizaje supervisado basado en el ensamblado (*ensemble*) de múltiples árboles de decisión. Combina las ideas de *bagging* (*bootstrap aggregating*) y aleatorización para mejorar la precisión y robustez frente a problemas de sobreajuste.

El *bagging* entrena múltiples árboles en subconjuntos aleatorios de los datos y promedia las predicciones de todos los árboles. La aleatoriedad adicional está dada por que, en cada división de un árbol, solo se considera un subconjunto aleatorio de características. Este

algoritmo tiene mejor desempeño que árboles individuales en problemas complejos, aunque puede requerir mayor tiempo de entrenamiento.

Los algoritmos de RF y DT comparten algunos hiperparámetros, pero RF introduce parámetros adicionales propios de su naturaleza de ensamblado, como el número de árboles que determina cuántos árboles componen el modelo; un número pequeño de árboles entregará un modelo rápido, pero potencialmente subajustado y un número alto de árboles mejora la precisión, pero aumenta el costo computacional.

Regresión Logística:

La regresión logística (RL) es un algoritmo de aprendizaje automático utilizado para predecir la probabilidad de un evento binario, es decir, uno de dos posibles resultados. Utiliza una función logística para modelar la relación entre una característica dependiente binaria y una o más características independientes [18].

A diferencia de la regresión lineal, LR modela la probabilidad de que una observación pertenezca a una clase mediante una función sigmoide[3], lo que la hace ideal para predecir resultados categóricos.

Dentro de las ventajas destacadas del LR está la capacidad de explicación del coeficiente y el impacto de cada variable, el entrenamiento es rápido incluso en *datasets* grandes, el resultado del modelo devuelve probabilidades lo que es útil para tomar decisiones, sin embargo, su aplicación requiere balanceo en clases porque datos desbalanceados sesgan las probabilidades.

K-Vecinos más Cercanos:

El algoritmo K-Vecinos más cercanos (KNN, *K-Nearest Neighbors*) es un algoritmo de aprendizaje automático que clasifica un nuevo dato basándose en las etiquetas de sus 'k' vecinos más cercanos. Funciona midiendo la distancia entre el nuevo dato y todos los demás

[3] La **función sigmoide** (o *función logística*) es una curva en forma de "S" que transforma cualquier valor real en un rango entre **0 y 1**.

en el conjunto de datos, y luego asigna la etiqueta más común entre los vecinos cercanos [19].

KNN clasifica o predice en función de la cercanía entre puntos de datos, calculada mediante métricas de distancia:

- Distancia Euclidiana (L_2): en datos continuos y espacios de baja dimensión.
- Distancia Manhattan (L_1): Más robusta a *outliers*.
- Distancia de Minkowski: generaliza Euclidiana y Manhattan.
- Distancia de Hamming para datos categóricos.

El hiperparámetro clave en el algoritmo es el valor de *k, un k* pequeño implica fronteras de decisión complejas; y un *k* grande suaviza las fronteras, pero puede perder patrones locales (subajuste).

Máquina de Vectores de Soporte:

Las máquinas de vectores de soporte (*Support Vector Machines,* SVM) son algoritmos de aprendizaje supervisado que clasifica datos encontrando el hiperplano óptimo, es decir, la línea o superficie que separa dos clases maximizando la distancia (margen) entre los puntos de datos de cada clase. SVM es efectivo en espacios de alta dimensión y es útil para problemas de clasificación binaria y multiclase [20].

Los elementos claves del SVM son el hiperplano optimo, los vectores de soporte y la optimización. El hiperplano óptimo es el que maximiza la distancia o margen entre las clases que se están clasificando, cuanto mayor sea el margen, mejor será la capacidad de generalización del modelo. Los vectores de soporte son los puntos de datos más cercano de cada clase. La optimización de esta técnica implica maximiza el margen y minimiza los errores de clasificación.

Los parámetros de aplicación del modelo son el C de regularización, en donde un C alto tiene menos tolerancia a errores o riesgo de sobreajuste y el parámetro y de Kernel que

define la influencia de un solo punto de entrenamiento, un y alto significa mayor complejidad en el modelo.

El SVM tiene mayor efectividad en datos con muchas características y adaptabilidad a problemas no lineales usando Kernel, aunque tiene alto costo computacional en *datasets* grandes e interpretabilidad más compleja que otros algoritmos como los árboles de decisión.

XGBoost:

eXtreme Gradient Boosting (XGBoost) es un algoritmo de aprendizaje automático que mejora la precisión del modelo mediante la técnica de *boosting*, la cual combina múltiples árboles de decisión secuencialmente, donde cada nuevo árbol corrige los errores del anterior [21].

El *boosting* es el aprendizaje secuencial que entrena árboles de decisión de forma iterativa, donde cada nuevo árbol corrige los errores del anterior y minimiza el gradiente.

Los hiperparámetros de aplicación en *XGBoost* controlan la complejidad o profundidad de los árboles, la regularización y el sobreajuste. El algoritmo ha demostrado eficiencia computacional y alto rendimiento predictivo, aunque mayores costes computacionales que otras técnicas como *Random Forest* en *datasets* grandes.

Consecuentemente, luego de aplicar los algoritmos sobre los datos existentes, se requiere definir criterios de selección sobre el modelo más adecuado, según las necesidades y alcances de la investigación, para ello se usan las métricas de evaluación como la exactitud, sensibilidad y especificidad que miden la eficacia de los modelos de clasificación en aprendizaje automático [22]. A partir de las cuales se toma la decisión del mejor modelo.

3.2 ANTECEDENTES

Se realizó una revisión de literatura para identificar el uso de modelos de aprendizaje automático supervisado en trabajos o publicaciones realizadas que sirvan de insumo para este estudio, así como estudios que utilizan características o métricas para definir la relación entre costo de atención y características de los pacientes.

En el estudio "*A machine learning-based risk stratification tool for in-hospital mortality of intensive care unit patients with heart failure*" [23] se utilizó el algoritmo *XGBoost* para desarrollar modelos predictivos de mortalidad en pacientes con insuficiencia cardíaca, en más de 200 hospitales de Estados Unidos. El modelo *XGBoost* mostró un mejor poder predictivo en comparación con otros modelos, lo que permite una estratificación más precisa del riesgo y mejora la gestión clínica en unidades de cuidados intensivos.

El artículo "*Data Mining in Healthcare using Machine Learning Techniques*" [24] aborda la aplicación de algoritmos de clasificación supervisada en la atención médica, específicamente para predecir y diagnosticar enfermedades como la cardiopatía. El estudio utiliza algoritmos como *Naive* Bayes, regresión logística y *Random Forest* para analizar conjuntos de datos que incluyen información demográfica, historial médico y diagnósticos de pacientes. Las métricas utilizadas para evaluar el desempeño de estos algoritmos incluyen la exactitud, con la regresión logística como método seleccionado mostrando una exactitud del 99%, superando a *Naive* Bayes (90.39%) y *Random Forest* (98.10%). El objetivo principal del estudio es demostrar cómo estas técnicas de minería de datos pueden mejorar la precisión en la predicción de riesgos de enfermedades, asistiendo en el diagnóstico temprano y la prevención, y permitiendo a los proveedores de salud tomar decisiones más informadas sobre el tratamiento y manejo de pacientes.

El estudio "*Predicting healthcare expenditure by multimorbidity groups*" [25] utiliza modelos en dos partes para predecir el gasto en salud de un distrito de salud en España utilizando grupos de riesgo clínico. La primera parte del modelo utiliza una regresión logística (logit)

para modelar la probabilidad de cualquier gasto, mientras que la segunda parte emplea una regresión lineal logarítmica (OLS) para modelar los costos. Estos modelos permiten ajustar los pesos relativos de cada grupo para mejorar la asignación presupuestaria basada en la multimorbilidad.

Por otra parte, el estudio de Liao et al. para Kaiser Permanente Northern California "*Development and validation of prediction models for gestational diabetes treatment modality using supervised machine learning: a population-based cohort study*" [26] desarrolló modelos predictivos a partir de datos clínicos en diferentes etapas del embarazo para predecir la necesidad de tratamiento farmacológico en mujeres con diabetes gestacional, como algoritmos como árboles de decisión (CART), regresión LASSO, *Random Forest* y *XGBoost*.

Además, en un estudio sobre "*Applied machine learning for the risk-stratification and clinical decision support of hospitalised patients with dengue in Vietnam*" [27] se desarrollaron modelos de predicción utilizando redes neuronales para estratificar el riesgo de pacientes con dengue. Utilizando datos clínicos, los modelos lograron demostrar la identificación temprana de complicaciones graves y apoyando la toma de decisiones clínicas en entornos de bajos recursos.

Como se evidencia en la literatura revisada, la aplicación de algoritmos de aprendizaje supervisado en el ámbito de la salud es frecuente y de gran utilidad, particularmente en la predicción de desenlaces clínicos asociados al curso de enfermedades. Estos modelos suelen enfocarse en riesgos específicos vinculados a condiciones médicas particulares. Sin embargo, no se identificaron estudios previos que aborden la estratificación de riesgo en salud integral propuesta en esta investigación, lo que resalta la novedad de nuestro enfoque.

4. PREPARACIÓN DE LA BASE DE DATOS ESTRUCTURADA MEDIANTE LA GESTIÓN DEL PERMISO DE USO, ETIQUETADO Y ANONIMIZADO

En este capítulo se describe el procedimiento metodológico para lograr el primer objetivo específico. La preparación de la base de datos constituye la fase inicial para el desarrollo de este proyecto. Esta sección documenta el pipeline de transformación de datos, iniciando con la descripción de los tipos de información: poblacionales, administrativas y clínicas; seguido por su agrupación categórica, etiquetado estandarizado y consolidación en una estructura unificada, optimizada para su uso en aplicación de los modelos automatizados.

La producción de datos en el sector salud inicia con la recolección de información del individuo. Esta base poblacional -que en el contexto asegurador contiene los registros de afiliación- almacena variables clave como: ID único del paciente, edad, estado de afiliación, antigüedad en el sistema y datos demográficos básicos, los cuales funcionan como llave de enlace con otras bases del sistema. Para el presente estudio, se seleccionaron específicamente tres campos: ID de persona (identificador único), nombre completo, y fecha de nacimiento, para calcular la edad, siendo esta última variable fundamental en el posterior modelo de estratificación de riesgo.

La segunda fuente de datos proviene de las transacciones financieras del sistema, las facturas de servicios médicos reclamados al seguro. Esta base contiene información detallada de cada prestación, incluyendo:

- Identificación del servicio: Código único de procedimiento (CUPS en Colombia).
- Contexto clínico: Diagnóstico asociado (codificado en CIE-10).
- Características operativas: Fecha de servicio, Ámbito (hospitalario/ambulatorio/urgencias/medicamentos), Prestador (institución o profesional codificado).
- Costo total de la reclamación.

Para la estratificación de riesgo, se priorizan tres elementos clave: el tipo de servicio, que aproxima la complejidad clínica; la distancia entre la fecha del servicio y el momento de estimación de riesgo de la persona; y los costos acumulados, los cuales se transforman en variables cuantitativas y categóricas para el modelo.

La tercera fuente dispone de la información sobre los antecedentes médicos del paciente, la cual incluye los diagnósticos históricos registrados. Esta información es utilizada para la conformación de cohortes de seguimiento de enfermedades. A partir de esta fuente, es posible identificar si el paciente presenta enfermedades crónicas y cuáles son, según los códigos CIE-10 correspondientes.

La integración de las distintas bases de datos se realiza mediante la creación de una llave compuesta por el tipo y número de identificación, complementada con la validación del nombre de la persona. El proceso de limpieza, verificación de consistencia y calidad de los datos es llevado a cabo por el área de tecnología de la empresa, lo que permite que los procesos analíticos se inicien con información de alta calidad.

La categorización de diagnósticos para identificar condiciones crónicas constituye un proceso metodológicamente complejo que requiere análisis clínico experto. Con un equipo de médicos y epidemiólogos evalúan los cerca de 18.000 códigos CIE-10 disponibles, estableciendo relaciones semánticas y jerárquicas entre ellos; se definieron 21 agrupaciones prioritarias de enfermedades crónicas. Estas categorías se implementan en reglas de negocio que permiten calcular tres métricas clave: número de condiciones presentes, severidad por frecuencia de servicios asociados y control de la condición cuando aplique, las cuales alimentan directamente el algoritmo de riesgo individual. Este proceso garantiza que la transformación de datos crudos (CIE-10) en características para el modelo.

En el marco de la preparación de datos para el análisis, se trabajó con una base de datos anonimizada. La base fue procesada y cargada en Google Colab, facilitando el análisis en un entorno colaborativo y seguro. Esta base, correspondiente al tercer trimestre de 2022, fue utilizada debido a la conveniencia de contar con registros actualizados de facturación y

utilización de servicios de salud, garantizando la relevancia de los datos para los objetivos planteados. La base incluye una columna de resultado a clasificar y 42 características, y contiene información de 5.791.957 personas únicas, donde cada fila representa a un individuo. La base de datos utilizada tiene un tamaño de 1.370.837 KB, lo que refleja la magnitud de la información recopilada para el análisis.

El objetivo principal del análisis fue la predicción de una variable relacionada con el riesgo y la severidad, combinando estas dos dimensiones para evaluar el estado de salud de los pacientes. Se realizó un proceso de etiquetado de las características, de acuerdo con los criterios internos del modelo de riesgo de Keralty, para definir problemas de clasificación de 36 clases (6 de riesgo subdividido en 6 niveles cada uno de severidad).

En cuanto al tratamiento de los datos, no fue necesario eliminar *outliers*, ya que todos los registros médicos han sido previamente validados, asegurando la calidad de la información y un valor atípico representa un resultado en sí. En relación con los datos faltantes, se optó por asignar la categoría "No Aplica" en aquellas variables relacionadas con el control de enfermedades, asegurando así la consistencia de los datos sin eliminar registros valiosos cuando un usuario no tuviese alguna condición crónica. Este enfoque permitió trabajar con un conjunto de datos limpio y representativo, adecuado para el análisis posterior.

A continuación, se describen las características utilizadas y las categorías que se toman en el proceso de etiquetado:

- Los valores de Riesgo/Severidad reflejan una calificación numérica del nivel de severidad y riesgo de salud en una persona. Los valores varían en 36 categorías ordinales (para riesgo: 1 Sin severidad, 2 Baja, 3 Media, 4 Media alta, 5 Alta, 6 Muy alta; y para severidad: 1 Sin severidad, 2 Baja, 3 Media, 4 Media alta, 5 Alta, 6 Muy alta).
- La característica Artropatías Inflamatorias indica la presencia o ausencia de este tipo de enfermedades inflamatorias articulares. Los valores posibles son 0, que representa la ausencia de la enfermedad, y 1, que indica la presencia de artropatías inflamatorias.

- La característica Asma refleja si el individuo tiene o no diagnóstico de asma. Los valores son 0 para aquellos que no presentan la condición, y 1 para quienes sí la padecen.

- La característica c_Artropatías Inflamatorias se refiere a la categoría de control para las personas con artropatías inflamatorias. Los valores son: "No Aplica", para quienes no tienen la enfermedad; "CONTROLADO", que indica que la enfermedad está bajo control; "NO CONTROLADO", para los casos en los que no se ha logrado controlar la enfermedad; y "NO ESPECIFICO", que se refiere a casos sin información clara sobre el estado de control.

- La característica c_Asma describe el control del asma en las personas que tienen la condición. Los valores son: "No Aplica" para quienes no tienen asma, "CONTROLADO" para aquellos con la condición controlada, y "NO CONTROLADO" para quienes no tienen la enfermedad bajo control.

- La característica c_Diabetes representa el estado de control en las personas que tienen diabetes. Los valores son: "No Aplica" para quienes no padecen diabetes, "CONTROLADO" para quienes la tienen controlada, "NO CONTROLADO" para quienes no han logrado controlar la condición, y "NO ESPECIFICO" en los casos donde no hay claridad sobre el control.

- La característica c_Epoc refiere al control de la Enfermedad Pulmonar Obstructiva Crónica (EPOC). Los valores son: "No Aplica" para quienes no tienen esta enfermedad, "CONTROLADO" para aquellos con la enfermedad controlada, y "NO CONTROLADO" para los que no han alcanzado control.

- La característica c_ERC indica el control de la Enfermedad Renal Crónica. Los valores son: "No Aplica" para quienes no presentan esta enfermedad, "CONTROLADO" para los casos bajo control, "NO CONTROLADO" para aquellos sin control de la enfermedad, y "NO ESPECIFICO" para quienes no tienen claridad sobre el control.

- La característica c_FallaCardiaca describe el control de la falla cardíaca en la población. Los valores son: "No Aplica" para quienes no tienen esta condición, "CONTROLADO" para aquellos con la falla cardíaca controlada.

- La característica c_HTA indica el control de la hipertensión arterial. Los valores son: "No Aplica" para aquellos sin hipertensión, "CONTROLADO" para quienes la tienen bajo control, "NO CONTROLADO" para quienes no han controlado la condición, y "NO ESPECIFICO" para los casos donde no se especifica el control.

- La característica c_SaludMental refleja el control de las condiciones de salud mental. Los valores son: "No Aplica" para quienes no tienen estas condiciones, "CONTROLADO" para quienes tienen la condición bajo control, "NO CONTROLADO" para los casos no controlados, y "NO ESPECIFICO" cuando no se ha especificado el control.

- La característica c_VIH describe el control de la infección por VIH. Los valores son: "No Aplica" para quienes no están infectados, "CONTROLADO" para quienes tienen la infección controlada, y "NO CONTROLADO" para aquellos sin control de la infección.

- La característica Cáncer indica la presencia o ausencia de cáncer. Los valores posibles son 0, que representa la ausencia de la enfermedad, y 1, que indica la presencia de cáncer.

- La característica Consultas Especialistas muestra la cantidad de consultas con especialistas que ha recibido una persona. Los valores varían desde 0, que indica ninguna consulta, hasta valores que representan múltiples consultas.

- La característica Diabetes refleja si la persona tiene o no diagnóstico de diabetes. Los valores son 0 para quienes no la tienen, y 1 para quienes sí.

- La característica Dislipidemia indica la presencia o ausencia de dislipidemia, que son alteraciones en los niveles de lípidos en la sangre. Los valores son 0 para ausencia de dislipidemia, y 1 para presencia de la condición.

- La característica Edad muestra la edad de las personas, con valores que representan los diferentes grupos etarios en la muestra.

- La característica Epoc señala la presencia o ausencia de Enfermedad Pulmonar Obstructiva Crónica. Los valores son 0 para la ausencia de la enfermedad, y 1 para la presencia de EPOC.

- La característica ERC representa la presencia o ausencia de Enfermedad Renal Crónica. Los valores posibles son 0, que indica la ausencia de la enfermedad, y 1, que señala la presencia de ERC.

- La característica FallaCardiaca refleja si una persona tiene o no diagnóstico de falla cardíaca. Los valores son 0 para ausencia de la condición y 1 para su presencia.

- La característica Gastrointestinal indica la presencia o ausencia de condiciones gastrointestinales. Los valores son 0 para quienes no tienen estas condiciones y 1 para quienes sí las padecen.

- La característica Hemofilia refleja la presencia o ausencia de hemofilia en la población. Los valores son 0 para quienes no tienen la enfermedad, y 1 para quienes sí.

- La característica Hospitalizaciones representa el número de veces que una persona ha sido hospitalizada. Los valores van desde 0, para quienes no han sido hospitalizados, hasta valores que indican múltiples hospitalizaciones.

- La característica HTA indica si la persona tiene o no hipertensión arterial. Los valores son 0 para quienes no tienen hipertensión, y 1 para quienes sí.

- La característica HTP refleja la presencia o ausencia de hipertensión pulmonar. Los valores son 0 para ausencia y 1 para presencia de esta condición.

- La característica Medicamentos Crónicos muestra el número de medicamentos crónicos que una persona consume. Los valores varían desde 0, para quienes no consumen medicamentos crónicos, hasta valores que indican el consumo de varios medicamentos.

- La característica n_Condicionantes indica el número de condicionantes de salud presentes en una persona. Los valores varían desde 0 hasta números que reflejan múltiples condicionantes.

- La característica Neurológicas señala la presencia o ausencia de enfermedades neurológicas. Los valores son 0 para quienes no tienen estas condiciones y 1 para quienes sí las padecen.

- La característica Num_Crónicas representa el número de enfermedades crónicas que presenta una persona. Los valores van desde 0, para quienes no tienen enfermedades crónicas, hasta números que indican la presencia de múltiples condiciones crónicas.

- La característica Obesidad indica si una persona tiene o no obesidad. Los valores son 0 para quienes no presentan obesidad y 1 para quienes sí.

- La característica Osteomuscular refleja la presencia o ausencia de condiciones osteomusculares. Los valores son 0 para ausencia y 1 para presencia de estas condiciones.

- La característica Otras Cardiovasculares muestra si la persona tiene o no otras condiciones cardiovasculares. Los valores son 0 para ausencia y 1 para presencia de estas enfermedades.

- La característica Otras Huerfanas indica la presencia o ausencia de enfermedades huérfanas. Los valores son 0 para quienes no tienen estas enfermedades y 1 para quienes sí.

- La característica Paliativos refleja si la persona recibe cuidados paliativos. Los valores son 0 para quienes no reciben este tipo de cuidados y 1 para quienes sí.

- La característica Reingreso Crónico muestra si una persona ha tenido reingreso por alguna condición crónica. Los valores son 0 para quienes no han reingresado y 1 para quienes sí.

- La característica Salud Mental indica la presencia o ausencia de condiciones de salud mental. Los valores son 0 para ausencia de estas condiciones y 1 para quienes las padecen.

- La característica Sedentarismo refleja si la persona tiene una vida sedentaria. Los valores son 0 para quienes no presentan sedentarismo y 1 para quienes sí.

- La característica Tabaquismo indica si la persona es fumadora. Los valores son 0 para quienes no fuman y 1 para quienes sí.

- La característica Trasplantes refleja si la persona ha recibido algún trasplante. Los valores son 0 para quienes no han recibido trasplante y 1 para quienes sí.

- La característica Trastornos Tiroideos indica la presencia o ausencia de trastornos en la glándula tiroides. Los valores son 0 para quienes no tienen la condición y 1 para quienes sí.

- La característica UCI refleja si la persona ha sido ingresada en la Unidad de Cuidados Intensivos. Los valores son 0 para quienes no han ingresado a UCI y 1 para quienes sí.

- La característica Urgencias muestra el número de visitas a urgencias que una persona ha realizado. Los valores varían desde 0, para quienes no han acudido a urgencias, hasta números que reflejan múltiples visitas.

- La característica VIH indica la presencia o ausencia de infección por VIH. Los valores son 0 para quienes no tienen VIH y 1 para quienes sí.

5. IMPLEMENTACIÓN DE MODELOS DE APRENDIZAJE SUPERVISADO PARA LA CLASIFICACIÓN DE NIVELES DE RIESGO EN SALUD DE PACIENTES

En este capítulo se describe el procedimiento metodológico que responde al segundo objetivo específico. Cómo ya se mencionó, se decidió implementar y comparar múltiples algoritmos de aprendizaje supervisado. Todo el trabajo fue realizado en Google Colab Pro, una plataforma que facilita la ejecución de modelos y procesamiento de datos en la nube. Para manejar la complejidad de los modelos y el tamaño del conjunto de datos, fue necesario utilizar Google Colab Pro, ya que esta suscripción brinda acceso a más memoria RAM y a GPU más potentes, lo que permitió realizar entrenamientos y pruebas con un rendimiento adecuado.

5.1 ALGORITMOS APLICADOS E HIPERPARÁMETROS

En la sesión 3.1 se describieron los algoritmos que se aplicaron para el desarrollo de la investigación, se describe los resultados obtenidos:

- Árbol de Decisión: En el caso del árbol de decisión, se optó por max_depth=10 para limitar la profundidad del árbol, con el fin de evitar sobreajuste y mejorar la capacidad de generalización del modelo.
- Random Forest: Se utilizó un modelo de RF con n_estimators=100, lo cual indica que el modelo está compuesto por 100 árboles de decisión. El parámetro max_depth=10 se estableció para limitar la profundidad de cada árbol, evitando así un exceso de ajuste a los datos (*overfitting*).
- Regresión Logística: Para el modelo de RL, se empleó random_state=42 con el fin de obtener resultados consistentes en cada ejecución. Este modelo es menos intensivo

en términos de memoria y cómputo, por lo que no requirió configuraciones adicionales.

- K-Nearest Neighbors: En el modelo de KNN, se seleccionó n_neighbors=3, lo que significa que el modelo considerará los tres vecinos más cercanos para clasificar cada observación.

- Support Vector Machine con Kernel RBF: Para el modelo de SVM, se utilizó el kernel rbf, que es ideal para datos no lineales, permitiendo que el modelo capture relaciones más complejas entre las variables. El parámetro probability=True fue necesario para obtener probabilidades en la salida del modelo, lo cual facilitó la generación de curvas ROC y AUC para evaluar el desempeño en clasificación multiclase.

- XGBoost: El modelo XGBoost se configuró con use_label_encoder=False y eval_metric='mlogloss', ya que XGBoost elimina el uso del label_encoder en versiones recientes, y se seleccionó la métrica mlogloss (log-loss multiclase) por ser una de las más adecuadas en clasificación multiclase.

Cabe aclarar que la base utilizada está desbalanceada (Tabla 1) por lo que se corrieron modelos tanto balanceados (al de menor tamaño) como desbalanceados para ver el ajuste del modelo, utilizando una partición de 80% de entrenamiento y 20% de prueba. Además, Se realizó validación *holdout* aleatoria repetida 10 veces para evaluar la estabilidad de las métricas de los modelos con mejores métricas. En cada iteración, se dividieron los datos (80% entrenamiento, 20% prueba) variando la semilla aleatoria, se entrenó el modelo con hiperparámetros predeterminados, y se calcularon métricas de desempeño como *accuracy*, *F1-score* (macro), precisión y *recall* promedio.

Tabla 1. Distribución total de observaciones por clase

Riesgo	Severidad	Observaciones	%
1	1	3,096,943	53.5%
1	2	523,159	9.0%
1	3	168,401	2.9%
1	4	45,724	0.8%
1	5	16,221	0.3%
1	6	15,142	0.3%
2	1	469,389	8.1%
2	2	114,444	2.0%
2	3	42,322	0.7%
2	4	14,515	0.3%
2	5	4,649	0.1%
2	6	4,255	0.1%
3	1	130,559	2.3%
3	2	119,217	2.1%
3	3	77,563	1.3%
3	4	52,952	0.9%
3	5	8,874	0.2%
3	6	3,935	0.1%
4	1	272,722	4.7%
4	2	156,562	2.7%
4	3	91,376	1.6%
4	4	57,791	1.0%
4	5	16,453	0.3%
4	6	9,248	0.2%
5	1	13,739	0.2%
5	2	19,775	0.3%
5	3	28,553	0.5%
5	4	56,934	1.0%
5	5	26,132	0.5%
5	6	4,043	0.1%
6	1	32,619	0.6%
6	2	30,430	0.5%
6	3	24,598	0.4%
6	4	25,291	0.4%
6	5	12,903	0.2%
6	6	4,524	0.1%
Total		5,791,957	100.0%
mínimo		3,935	

Tabla 2. Distribución de la base balanceada a la de menor tamaño

Riesgo	Severidad	Observaciones	%
1	1	3,935	2.8%
1	2	3,935	2.8%
1	3	3,935	2.8%
1	4	3,935	2.8%
1	5	3,935	2.8%
1	6	3,935	2.8%
2	1	3,935	2.8%
2	2	3,935	2.8%
2	3	3,935	2.8%
2	4	3,935	2.8%
2	5	3,935	2.8%
2	6	3,935	2.8%
3	1	3,935	2.8%
3	2	3,935	2.8%
3	3	3,935	2.8%
3	4	3,935	2.8%
3	5	3,935	2.8%
3	6	3,935	2.8%
4	1	3,935	2.8%
4	2	3,935	2.8%
4	3	3,935	2.8%
4	4	3,935	2.8%
4	5	3,935	2.8%
4	6	3,935	2.8%
5	1	3,935	2.8%
5	2	3,935	2.8%
5	3	3,935	2.8%
5	4	3,935	2.8%
5	5	3,935	2.8%
5	6	3,935	2.8%
6	1	3,935	2.8%
6	2	3,935	2.8%
6	3	3,935	2.8%
6	4	3,935	2.8%
6	5	3,935	2.8%
6	6	3,935	2.8%
Total		141,660	100.0%
mínimo		3,935	

La Tabla 3 presenta el resumen de los resultados de la métrica *F1 Score* de mayor a menor según el desempeño de los modelos aplicados. Las tablas 4 a 13 presentan el detalle de las salidas de cada modelo aplicado con las métricas de *precision, recall y accuracy,* así como los hiperparámetros utilizados en cada caso.

Tabla 3. Resumen de resultados F1 score

# Modelo	Algoritmo	Balanceo	F1 (score) macro avg	F1 (score) weighted avg
1	XGBoost	Balanceado	1	1
		SIN	1	1
2	Random Forest	Balanceado	0,81	0,81
		SIN	0,62	0,93
3	Decision Tree	Balanceado	0,72	0,72
		SIN	0,82	0,96
4	KNN	Balanceado	0,66	0,66
		SIN	N.A	N.A
5	Regresion logistica	Balanceado	0,57	0,57
		SIN	0,23	0,74
6	SVM	Balanceado	0,57	0,57
		SIN	N.A	N.A

Tabla 4. Resultados de modelos corridos XGboost

Modelo 1					
Xgboost sin balancear					
https://colab.research.google.com/drive/1QQ9UlirQ2AnQ4Nj462bcraiuH2l1uEJi?usp=sharing					
Se usó XGBoosting con los hiperparámetros use_label_encoder=False (desactiva el codificador interno para evitar advertencias) y eval_metric='mlogloss' (calcula qué tan lejos están las predicciones de las probabilidades reales). Los demás parámetros se mantuvieron en sus valores predeterminados, es decir, no se introdujeron en el código, pero se usaron por defecto: learning_rate=0.3 (tasa de aprendizaje), n_estimators=100 (número de árboles), max_depth=6 (profundidad máxima del árbol), subsample=1 (fracción de muestras usadas por árbol), colsample_bytree=1 (fracción de características usadas por árbol) y objective='binary:logistic' (clasificación binaria). La base de datos no fue balanceada y contiene 36 categorías (0 para 1.1 y 35 para 6.6). Se entrenó el modelo con el 80% de los datos y se probó con el 20% restante, correspondiente a 1,158,392 registros.					
	Precision	Recall	Accuracy	F1 score	Support (20%)
			1		
macro avg	1	1		1	1,158,392
weighted avg	1	1		1	1,158,392
Modelo 1					
XGboost balanceada al tamaño de la menor clase					
https://colab.research.google.com/drive/1SyYrrAXrxl2x72iEA8YYz0WiQJwLUfzN?usp=sharing					
Se usó XGBoosting con los hiperparámetros use_label_encoder=False (desactiva el codificador interno para evitar advertencias) y eval_metric='mlogloss' (calcula qué tan lejos están las predicciones de las probabilidades reales). Los demás parámetros se mantuvieron en sus valores predeterminados, es decir, no se introdujeron en el código, pero se usaron por defecto: learning_rate=0.3 (tasa de aprendizaje), n_estimators=100 (número de árboles), max_depth=6 (profundidad máxima del árbol), subsample=1 (fracción de muestras usadas por árbol), colsample_bytree=1 (fracción de características usadas por árbol) y objective='binary:logistic' (clasificación binaria). La base de datos fue balanceada a 3,935 por clase y contiene 36 categorías (0 para 1.1 y 35 para 6.6). Se entrenó el modelo con el 80% de los datos y se probó con el 20% restante, correspondiente a 28,332 registros.					
	Precision	Recall	Accuracy	F1 score	Support (20%)
			1		
macro avg	1	1		1	28,332
weighted avg	1	1		1	28,332

Tabla 5. Randon Holdout XGBoost

Random Holdout Modelo 1					
Xgboost sin balancear					
https://colab.research.google.com/drive/1nRA5tQyPPrwdg11fTwnSyJ6eee2SPTzi?usp=sharing					
Se realizó validación holdout aleatoria repetida 10 veces para evaluar la estabilidad de las métricas del modelo XGboost sin balancear. En cada iteración, se dividieron los datos (80% entrenamiento, 20% prueba) variando la semilla aleatoria, se entrenó el modelo con hiperparámetros predeterminados, y se calcularon métricas de desempeño como accuracy, F1-score (macro), precisión y recall promedio. Finalmente, se calcularon los promedios y desviaciones estándar de estas métricas para analizar la consistencia del modelo en diferentes particiones. El número de observaciones en cada partición es constante en las 10 pruebas. Lo único que varía es qué observaciones específicas caen en el conjunto de entrenamiento o prueba, determinado por la semilla aleatoria (random_state=i).					
	Precision (macro avg)	Recall (macro avg)	Accuracy	F1 score (macro avg)	n
AVG	0.9996	0.9995	1	0.9996	10
STD	0.0001	0.0001	0	0.0001	10

Random Holdout Modelo 1					
XGboost balanceada al tamaño de la menor clase					
https://colab.research.google.com/drive/1TsVyGCZCyDCXDSGZ7pZP_fk9Oz4Hsuxf?usp=sharing					
Se realizó validación holdout aleatoria repetida 10 veces para evaluar la estabilidad de las métricas del modelo XGboost balanceado a la menor clase. En cada iteración, se dividieron los datos (80% entrenamiento, 20% prueba) variando la semilla aleatoria, se entrenó el modelo con hiperparámetros predeterminados, y se calcularon métricas de desempeño como accuracy, F1-score (macro), precisión y recall promedio. Finalmente, se calcularon los promedios y desviaciones estándar de estas métricas para analizar la consistencia del modelo en diferentes particiones. El número de observaciones en cada partición es constante en las 10 pruebas. Lo único que varía es qué observaciones específicas caen en el conjunto de entrenamiento o prueba, determinado por la semilla aleatoria (random_state=i).					
	Precision	Recall	Accuracy	F1 score	Support (20%)
AVG	0.9986	0.9986	0.9986	0.9986	10
STD	0.0002	0.0002	0.0002	0.0002	10

Tabla 6. Resultados de modelos corridos Random Forest

Modelo 2					
Random forest balanceada al tamaño de la menor clase					
https://colab.research.google.com/drive/1bYGGwxPz2L0qlZX5j6r8VZYBJNrWsuz3?usp=sharing					
Se usó un modelo Random Forest con los hiperparámetros n_estimators=100 (número de árboles), max_depth=10 (profundidad máxima de cada árbol) y random_state=42 (para garantizar reproducibilidad). Además, se utilizó n_jobs=-1 (para aprovechar todos los núcleos disponibles y acelerar el entrenamiento). Los demás parámetros se mantuvieron en sus valores predeterminados, es decir, no se introdujeron en el código pero se usaron por defecto, como criterion='gini' (criterio para medir la calidad de la división) y bootstrap=True (uso de muestreo con reemplazo). La base de datos fue balanceada a 3,935 observaciones por clase y contiene 36 categorías (1 para 1.1 y 36 para 6.6). Se entrenó el modelo con el 80% de los datos y se probó con el 20% restante, correspondiente a 28,332 registros.					
	Precision	Recall	Accuracy	F1 score	Support (20%)
			0.81		
macro avg	0.83	0.81		0.81	28,332
weighted avg	0.83	0.81		0.81	28,332
Modelo 2					
Random forest sin balancear					
https://colab.research.google.com/drive/1yKbbHQkhu3ngO3y8MggIzLA5j41zrIeN?usp=sharing					
Se usó un modelo Random Forest con los hiperparámetros n_estimators=100 (número de árboles), max_depth=10 (profundidad máxima de cada árbol) y random_state=42 (para garantizar reproducibilidad). Además, se utilizó n_jobs=-1 (para aprovechar todos los núcleos disponibles y acelerar el entrenamiento). Los demás parámetros se mantuvieron en sus valores predeterminados, es decir, no se introdujeron en el código pero se usaron por defecto, como criterion='gini' (criterio para medir la calidad de la división) y bootstrap=True (uso de muestreo con reemplazo). La base de datos no es balanceada y contiene 36 categorías (1 para 1.1 y 36 para 6.6). Se entrenó el modelo con el 80% de los datos y se probó con el 20% restante, correspondiente a 1,158,392 registros.					
	Precision	Recall	Accuracy	F1 score	Support (20%)
			0.94		
macro avg	0.86	0.58		0.62	1,158,392
weighted avg	0.94	0.94		0.93	1,158,392

Tabla 7. Randon Holdout Random Forest

Random Holdout Modelo 2					
Random forest balanceada al tamaño de la menor clase					
https://colab.research.google.com/drive/1DjOpb3O8t3DTueMe8t9V3qPk7iyp2AG0?usp=sharing					
Se realizó validación holdout aleatoria repetida 10 veces para evaluar la estabilidad de las métricas del modelo Random Forest balanceado a la menor clase. En cada iteración, se dividieron los datos (80% entrenamiento, 20% prueba) variando la semilla aleatoria, se entrenó el modelo con los hiperparámetros n_estimators=100 (número de árboles), max_depth=10 (profundidad máxima de los árboles) y n_jobs=-1 (uso de todos los núcleos disponibles), manteniendo los demás en sus valores predeterminados. Se calcularon métricas de desempeño como accuracy, F1-score (macro), precisión y recall promedio. Finalmente, se calcularon los promedios y desviaciones estándar de estas métricas para analizar la consistencia del modelo en diferentes particiones. El número de observaciones en cada partición es constante en las 10 pruebas. Lo único que varía es qué observaciones específicas caen en el conjunto de entrenamiento o prueba, determinado por la semilla aleatoria (random_state=i).					
	Precision	Recall	Accuracy	F1 score	n
AVG	0.8367	0.8232	0.8235	0.8214	10
STD	0.0055	0.0062	0.0063	0.0063	10

Random Holdout Modelo 2					
Random forest sin balancear					
https://colab.research.google.com/drive/1AnD2chJayFbq2gK2XWY3Pdo-jpgPNF4q?usp=sharing					
Se realizó validación holdout aleatoria repetida 10 veces para evaluar la estabilidad de las métricas del modelo Random Forest desbalanceado. En cada iteración, se dividieron los datos (80% entrenamiento, 20% prueba) variando la semilla aleatoria, se entrenó el modelo con los hiperparámetros n_estimators=100 (número de árboles), max_depth=10 (profundidad máxima de los árboles) y n_jobs=-1 (uso de todos los núcleos disponibles), manteniendo los demás en sus valores predeterminados. Se calcularon métricas de desempeño como accuracy, F1-score (macro), precisión y recall promedio. Finalmente, se calcularon los promedios y desviaciones estándar de estas métricas para analizar la consistencia del modelo en diferentes particiones. El número de observaciones en cada partición es constante en las 10 pruebas. Lo único que varía es qué observaciones específicas caen en el conjunto de entrenamiento o prueba, determinado por la semilla aleatoria (random_state=i).					
	Precision	Recall	Accuracy	F1 score	Support (20%)
AVG	0.8702	0.5876	0.9384	0.6257	10
STD	0.0102	0.0042	0.0013	0.0039	10

Tabla 8. Resultados de modelos corridos Decision Tree

Modelo 3					
Decision tree sin balancear					
https://colab.research.google.com/drive/1z1lVQlZMBeCsAKG45li8rPD-sr_jFJAf?usp=sharing					
Se usó un modelo Decision Tree con los hiperparámetros max_depth=10 (profundidad máxima del árbol) y random_state=42 (para garantizar reproducibilidad). Los demás parámetros se mantuvieron en sus valores predeterminados, es decir, no se introdujeron en el código pero se usaron por defecto, como criterion='gini' (criterio para medir la calidad de la división) y splitter='best' (elige la mejor división en cada nodo). La base de datos no es balanceada y contiene 36 categorías (1 para 1.1 y 36 para 6.6). Se entrenó el modelo con el 80% de los datos y se probó con el 20% restante, correspondiente a 1,158,392 registros.					
	Precision	Recall	Accuracy	F1 score	Support (20%)
			0.96		
macro avg	0.89	0.76		0.82	1,158,392
weighted avg	0.96	0.96		0.96	1,158,392
Modelo 3					
Decision tree balanceada al tamaño de la menor clase					
https://colab.research.google.com/drive/1LLSEHd2NHH9OfRVSd50jY47tOy-6Imq9?usp=sharing					
Se usó un modelo Decision Tree con los hiperparámetros max_depth=10 (profundidad máxima del árbol) y random_state=42 (para garantizar reproducibilidad). Los demás parámetros se mantuvieron en sus valores predeterminados, es decir, no se introdujeron en el código pero se usaron por defecto, como criterion='gini' (criterio para medir la calidad de la división) y splitter='best' (elige la mejor división en cada nodo). La base de datos está balanceada a la menor clase y contiene 36 categorías (1 para 1.1 y 36 para 6.6). Se entrenó el modelo con el 80% de los datos y se probó con el 20% restante, correspondiente a 28,332 registros.					
	Precision	Recall	Accuracy	F1 score	Support (20%)
			0.72		
macro avg	0.76	0.72		0.72	28,332
weighted avg	0.76	0.72		0.71	28,332

Tabla 9. Randon Holdout Decision Tree

Random Holdout Modelo 3					
Decision tree sin balancear					
https://colab.research.google.com/drive/1RzCPg0Uy9oS0mUfeP4X4YQfgikL4IS2g?usp=sharing					
Se realizó validación holdout aleatoria repetida 10 veces para evaluar la estabilidad de las métricas del modelo de árbol de decisión desbalanceado. En cada iteración, se dividieron los datos (80% entrenamiento, 20% prueba) variando la semilla aleatoria, se entrenó el modelo con los hiperparámetros n_estimators=100 (número de árboles), max_depth=10 (profundidad máxima de los árboles) y n_jobs=-1 (uso de todos los núcleos disponibles), manteniendo los demás en sus valores predeterminados. Se calcularon métricas de desempeño como accuracy, F1-score (macro), precisión y recall promedio. Finalmente, se calcularon los promedios y desviaciones estándar de estas métricas para analizar la consistencia del modelo en diferentes particiones. El número de observaciones en cada partición es constante en las 10 pruebas. Lo único que varía es qué observaciones específicas caen en el conjunto de entrenamiento o prueba, determinado por la semilla aleatoria (random_state=i).					
	Precision	Recall	Accuracy	F1 score	Support (20%)
AVG	0.8963	0.7878	0.9605	0.8228	10
STD	0.0016	0.0018	0.0002	0.0032	10

Random Holdout Modelo 3					
Decision tree balanceada al tamaño de la menor clase					
https://colab.research.google.com/drive/1eQCBhQskg-5TiU381-SA3IkhqCwxfiAi?usp=sharing					
Se realizó validación holdout aleatoria repetida 10 veces para evaluar la estabilidad de las métricas del modelo de árbol de decisión balanceado. En cada iteración, se dividieron los datos (80% entrenamiento, 20% prueba) variando la semilla aleatoria, se entrenó el modelo con los hiperparámetros n_estimators=100 (número de árboles), max_depth=10 (profundidad máxima de los árboles) y n_jobs=-1 (uso de todos los núcleos disponibles), manteniendo los demás en sus valores predeterminados. Se calcularon métricas de desempeño como accuracy, F1-score (macro), precisión y recall promedio. Finalmente, se calcularon los promedios y desviaciones estándar de estas métricas para analizar la consistencia del modelo en diferentes particiones. El número de observaciones en cada partición es constante en las 10 pruebas. Lo único que varía es qué observaciones específicas caen en el conjunto de entrenamiento o prueba, determinado por la semilla aleatoria (random_state=i).					
	Precision	Recall	Accuracy	F1 score	Support (20%)
AVG	0.7668	0.7291	0.7292	0.7281	10
STD	0.0048	0.0048	0.0075	0.0074	10

Tabla 10. Resultados de modelos corridos KNN

Modelo 4					
KNN 3 balanceada al tamaño de la menor clase					
https://colab.research.google.com/drive/1FOAP6LNRS1FhpWov8i0wVj5WuP7Rqznn?usp=sharing					
Se usó un modelo K-Nearest Neighbors (KNN) con el hiperparámetro n_neighbors=3 (considera los 3 vecinos más cercanos para clasificar cada punto). Los demás parámetros se mantuvieron en sus valores predeterminados, es decir, no se introdujeron en el código pero se usaron por defecto, como metric='minkowski' (métrica para calcular la distancia) y weights='uniform' (todos los vecinos tienen el mismo peso). La base de datos es balanceada a la de menor clase y contiene 36 categorías (1 para 1.1 y 36 para 6.6). Se entrenó el modelo con el 80% de los datos y se probó con el 20% restante, correspondiente a 28,332 registros.					
	Precision	Recall	Accuracy	F1 score	Support (20%)
			0.67		
macro avg	0.67	0.67		0.66	28,332
weighted avg	0.67	0.67		0.66	28,332

Tabla 11. Resultados de modelos corridos Regresión Logística balanceada

Modelo 5					
Regresión logística balanceada al tamaño de la menor clase					
https://colab.research.google.com/drive/1R8Bx38BsZzq6_a7s7fPKKTWazAdG9_6X?usp=sharing					
Se usó un modelo de Regresión Logística con el hiperparámetro random_state=42 (para garantizar reproducibilidad). Los demás parámetros se mantuvieron en sus valores predeterminados, es decir, no se introdujeron en el código pero se usaron por defecto, como solver='lbfgs' (algoritmo para optimización), penalty='l2' (regularización L2) y max_iter=100 (máximo de iteraciones para la convergencia). La base de datos es balanceada a la de menor clase y contiene 36 categorías (1 para 1.1 y 36 para 6.6). Se entrenó el modelo con el 80% de los datos y se probó con el 20% restante, correspondiente a 28,332 registros.					
	Precision	Recall	Accuracy	F1 score	Support (20%)
			0.57		
macro avg	0.57	0.57		0.57	28,332
weighted avg	0.57	0.57		0.57	28,332

Tabla 12. Resultados modelos corridos Regresión Logística Desbalanceada

Modelo 5					
Regresión logística desbalanceada					
https://colab.research.google.com/drive/1-5x5DLECd5TK1yn-FYFKuIHxk7OhglqT?usp=sharing					
Se usó un modelo de Regresión Logística con el hiperparámetro random_state=42 (para garantizar reproducibilidad). Los demás parámetros se mantuvieron en sus valores predeterminados, es decir, no se introdujeron en el código pero se usaron por defecto, como solver='lbfgs' (algoritmo para optimización), penalty='l2' (regularización L2) y max_iter=100 (máximo de iteraciones para la convergencia). La base de datos no es balanceada y contiene 36 categorías (1 para 1.1 y 36 para 6.6). Se entrenó el modelo con el 80% de los datos y se probó con el 20% restante, correspondiente a 1,158,392 registros.					
	Precision	Recall	Accuracy	F1 score	Support (20%)
			0.77		
macro avg	0.27	0.23		0.23	1,158,392
weighted avg	0.73	0.77		0.74	1,158,392

Tabla 13. Resultados de modelos corridos SVM

Modelo 6					
SVM balanceada al tamaño de la menor clase					
https://colab.research.google.com/drive/1_QB3DqDOFmSapx4dsA9HZdAPJHBvhSa5?usp=sharing					
Se usó un modelo SVM (Support Vector Machine) con el hiperparámetro kernel='rbf' (función de base radial como núcleo), probability=True (permite calcular probabilidades para las clases) y random_state=42 (para garantizar reproducibilidad). Los demás parámetros se mantuvieron en sus valores predeterminados, es decir, no se introdujeron en el código pero se usaron por defecto, como C=1.0 (regularización) y gamma='scale' (ajuste automático del parámetro gamma). La base de datos es balanceada a la de menor clase y contiene 36 categorías (1 para 1.1 y 36 para 6.6). Se entrenó el modelo con el 80% de los datos y se probó con el 20% restante, correspondiente a 28,332 registros.					
	Precision	Recall	Accuracy	F1 score	Support (20%)
			0.58		
macro avg	0.59	0.58		0.57	28,332
weighted avg	0.59	0.58		0.57	28,332

En este análisis, los modelos se probaron tanto con datos desbalanceados como balanceados al tamaño de la menor clase. En general, el balanceo de las clases ayuda a mejorar la macro *average* (macro avg) y la *weighted average* (weighted avg) de métricas como precisión, *recall* y *F1-score*, lo que indica un mejor rendimiento en todas las clases en lugar de favorecer únicamente a las clases mayoritarias. En la regresión logística desbalanceada, la macro avg es muy baja (0.27 en precisión y 0.23 en *recall*), lo que indica un rendimiento deficiente en las clases minoritarias. Sin embargo, al balancear, el rendimiento en macro avg sube a 0.57, mostrando una mejora notable en la cobertura de todas las clases.

Respecto a KNN y SVM, al ser balanceados, muestran un mejor rendimiento en macro avg y weighted avg, lo que sugiere que el modelo tiene una mejor representación de las clases minoritarias después del balanceo, aunque estos algoritmos aún no alcanzan el rendimiento óptimo en comparación con *Random Forest*. Sobre este último, demuestra un rendimiento

sólido, especialmente en los datos balanceados. Con una *accuracy* de 0.81, junto con macro avg y weighted avg de 0.83, el *Random Forest* balanceado logra captar de manera más precisa tanto las clases minoritarias como las mayoritarias. Este resultado sugiere que *Random Forest* es particularmente eficaz en problemas de clasificación con múltiples clases y desequilibrios de clase, dado que su estructura basada en múltiples árboles ayuda a mitigar los sesgos hacia las clases mayoritarias.

Por su parte, el árbol de decisión, cuando se balancea al tamaño de la menor clase, muestra un rendimiento aceptable, con una *accuracy* de 0.72 y un macro avg de 0.76. Sin embargo, cuando el árbol de decisión se ejecuta sin balanceo, el *accuracy* es inusualmente alto (0.96), lo cual es una señal de que el modelo está sesgado hacia las clases mayoritarias. El balanceo ayuda a reducir este sesgo, asegurando que el modelo no dependa únicamente de las clases con más frecuencias. Es de resaltar que el modelo *XGBoost* muestra resultados casi perfectos en la clasificación (*accuracy*, macro avg, y weighted avg de 1.0 en todas las métricas) tanto en datos balanceados como desbalanceados.

6. SELECCIÓN DEL MEJOR MODELO

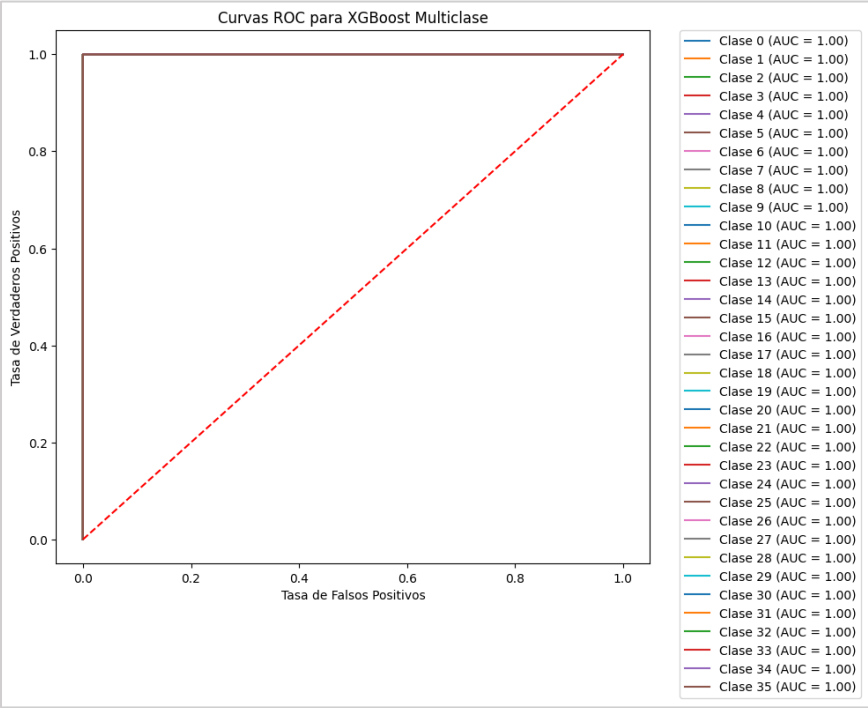
Para llegar al cumplimiento del objetivo 3 del proyecto, se consolidó la validación del modelo *XGBoost* sin balancear, el cual demostró ser la opción más estable y eficaz para automatizar la estratificación de riesgo utilizada por la organización. Dada la magnitud de la base de datos (más de cinco millones de registros y 42 características) y las limitaciones de trabajar en un equipo personal (con 7 GB de RAM personal), se utilizó Google Colab Pro (una versión paga) para contar con mayores capacidades de procesamiento (hasta 50 GB).

Aun así, la aplicación de técnicas más exigentes como validación cruzada o búsquedas extensas de hiperparámetros implicaba un costo de ejecución y monetario elevado. Por este motivo, y como ya se mostró en la sección anterior, se optó por validación mediante particiones aleatorias (*random hold-out*), lo que permitió evaluar la estabilidad del modelo sin comprometer la viabilidad del proyecto. Los resultados obtenidos con *XGBoost* sin balancear no solo fueron sobresalientes desde el punto de vista técnico, sino que reflejaron con precisión la clasificación discrecional aplicada previamente por expertos, lo que respalda su fidelidad al criterio clínico institucional.

Como parte de la evaluación final del modelo *XGBoost* sin balancear, se construyeron curvas ROC para las 36 clases, obteniendo en todos los casos un AUC igual a 1.00. Esta métrica indica que el modelo distingue perfectamente entre clases, sin errores de clasificación. Aunque este comportamiento puede parecer atípico, se ha verificado que se mantiene incluso al aplicarse sobre nuevos datos (cuarto trimestre de 2023), lo cual respalda la estabilidad y capacidad de generalización del modelo dentro del mismo sistema de etiquetado. Este comportamiento es especialmente destacable, ya que demuestra que *XGBoost* no solo se ajusta de manera consistente a los datos originales utilizados en el entrenamiento, sino que también mantiene niveles equivalentes de precisión al ser aplicado sobre registros correspondientes a un período posterior dentro del mismo sistema de información. Esto reafirma su estabilidad y su capacidad para generalizar bajo

condiciones operativas reales, lo cual es relevante para su uso continuado en tareas de clasificación multiclase sin necesidad de recalibraciones frecuentes.

Ilustración 1. Resultados sobre 80% de prueba



Los resultados obtenidos con el modelo *XGBoost* sin balancear al ser evaluado con datos del tercer trimestre de 2022 generaron interés por comprobar su estabilidad en otros periodos. Para ello, se aplicó el mismo modelo sobre un nuevo conjunto de datos correspondiente al cuarto trimestre de 2023, con el fin de observar si su rendimiento se mantenía y explorar qué variables explicaban en mayor medida la clasificación. La base de entrenamiento del tercer trimestre de 2022 contenía información de 5.791.957 personas únicas, mientras que la base de validación del cuarto trimestre de 2023 incluyó 5.955.768 registros individuales, sobre los cuales se aplicó el 100% del modelo ya entrenado para evaluar su desempeño en un conjunto completamente nuevo.

Tabla 14. Resultados del modelo de 2022Q3 sobre datos de 2023Q4

	Precision	Recall	F1 score	N (100%)
macro avg	1	1	1	5,955,768
weighted avg	1	1	1	5,955,768

Si bien el modelo *XGBoost* sin balancear mostró métricas cercanas a la perfección en casi todas las clases al ser aplicado sobre los datos del cuarto trimestre de 2023, es importante resaltar que en algunos casos puntuales se observaron valores ligeramente inferiores al 100% (Tabla 15). Por ejemplo, en las clases 10, 16, 17, 22 y 23, se registraron métricas de precisión o *recall* entre el 89% y el 99%. Estos pequeños desajustes no representan una debilidad, sino todo lo contrario: confirman que el modelo está operando con realismo, reconociendo correctamente la variabilidad natural en los datos y los posibles solapamientos entre clases. El hecho de que no todas las predicciones sean perfectas refuerza la idea de que el modelo no está memorizando, sino aprendiendo reglas generales con una capacidad sólida de generalización, lo cual resulta valioso para su aplicación práctica y sostenible en el tiempo.

En este sentido, respecto a los objetivos 3.1 y 3.2 del proyecto, se concluye que el modelo *XGBoost* sin balancear conserva un rendimiento prácticamente idéntico al aplicarse en datos correspondientes a otro trimestre. El modelo mantiene una exactitud cercana al 100%, junto con valores promedio de precisión, *recall* y *F1-score* también cercanos a 1. La matriz de confusión obtenida es casi perfectamente diagonal, lo que indica que todas las clases, incluso las menos representadas, fueron clasificadas con alta precisión. Esto sugiere que el modelo generaliza correctamente a nuevos datos sin requerir reentrenamiento, cumpliendo así con el objetivo 3.1 al mantener un desempeño estable en diferentes periodos.

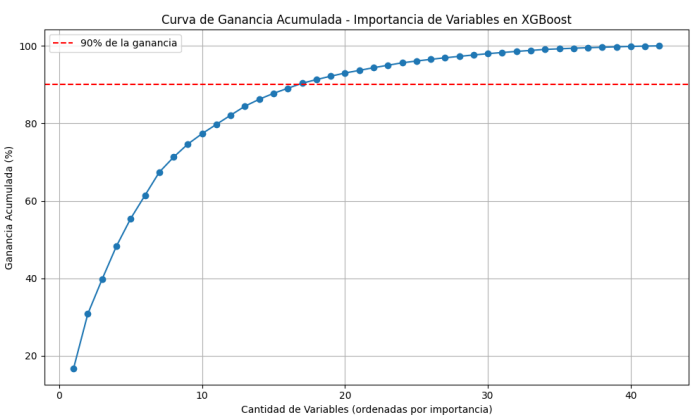
Tabla 15. Distribución resultados por clase en base nueva de prueba 2023T4

Risk - Severity	Class	precision	recall	f1-score	support
1.1	0	1	1	1	2,747,379
1.2	1	1	1	1	592,797
1.3	2	1	1	1	207,797
1.4	3	1	1	1	83,136
1.5	4	1	1	1	45,779
1.6	5	1	1	1	17,842
2.1	6	1	1	1	428,272
2.2	7	1	1	1	134,047
2.3	8	1	1	1	54,256
2.4	9	1	1	1	31,130
2.5	10	1	0.99	1	20,903
2.6	11	1	1	1	4,887
3.1	12	1	1	1	98,896
3.2	13	1	1	1	95,896
3.3	14	1	1	1	69,594
3	15	1	1	1	71,102
3.5	16	0.89	1	0.94	48,249
3.6	17	0.99	0.99	0.99	4,356
4.1	18	1	1	1	309,887
4.1	19	1	1	1	170,730
4.3	20	1	1	1	113,591
4.4	21	1	1	1	108,325
4.5	22	1	0.94	0.97	98,660
4.6	23	1	0.99	1	13,382
5.1	24	1	1	1	13,456
5.2	25	1	1	1	16,240
5.3	26	1	1	1	22,919
5.4	27	1	1	1	59,623
5.5	28	1	1	1	88,440
5.6	29	1	1	1	6,660
6.1	30	1	1	1	32,366
6.2	31	1	1	1	26,082
6.3	32	1	1	1	22,183
6.4	33	1	1	1	31,747
6.5	34	1	1	1	58,080
6.6	35	1	1	1	7,079

En cuanto al objetivo 3.2, se evidenció una ganancia práctica al conservar el modelo sin balancear de *XGBoost*, ya que evita procesos adicionales de rebalanceo que no aportaron mejoras significativas para este algoritmo específico. Esta decisión permitió optimizar los recursos computacionales y económicos sin sacrificar calidad en la clasificación. Además, se observó que 20 de las 42 variables explicaban más del 90% de la ganancia total del modelo (Ilustración 2), lo cual sugiere que una parte sustancial del poder predictivo del algoritmo se concentra en un subconjunto de variables relevantes. En el contexto de modelos basados en árboles de decisión como *XGBoost*, la ganancia (*gain*) mide cuánto mejora el modelo al dividirse usando una variable específica. Es una medida directa de la

contribución que cada característica hace a la reducción del error durante el proceso de entrenamiento, por lo que permite determinar qué variables fueron realmente determinantes en la clasificación realizada por el modelo.

Ilustración 2. Curva de ganancia acumulada de las características



Top 20 variables por ganancia:

	Variable	Ganancia
0	urgencias	2694.044922
1	num_cronicas	2267.751953
2	edad	1436.518799
3	n_condicionantes	1369.975098
4	SaludMental	1149.053345
5	medicamentos_cronicos	962.265442
6	Cancer	961.690002
7	Obesidad	624.804260
8	Asma	543.889954
9	Gastrointestinal	432.762451
10	hospitalizaciones	385.749329
11	OtrasHuerfanas	381.172180
12	c_HTA	373.959412
13	Diabetes	286.717712
14	consultas_especialista	245.904617
15	HTA	210.783554
16	c_Diabetes	208.397949
17	c_VIH	152.111191
18	c_SaludMental	140.083847
19	c_ERC	131.164139

Las variables que componen el 90% de la ganancia del modelo reflejan con solidez las dimensiones clínicas y de uso de servicios más críticas en la estratificación del riesgo, destacándose factores como la frecuencia de urgencias, la carga de enfermedad crónica, la edad, y el número de condicionantes, así como el control de condiciones específicas como cáncer, obesidad, salud mental y uso de medicamentos crónicos. Esta concentración de relevancia sugiere que el modelo no solo es eficiente, sino también clínicamente coherente con los criterios de gestión de riesgo utilizados por la organización.

En la sección de Anexos se incluye un resumen detallado del código correspondiente al modelo seleccionado (*XGBoost* sin balancear), así como de los demás algoritmos implementados durante el desarrollo del proyecto, entre ellos *Random Forest, Decision Tree*, Regresión Logística, SVM y KNN. También se documenta la validación realizada

mediante repeticiones de particiones aleatorias (*random hold-out*), empleada como estrategia de evaluación ante limitaciones computacionales. Este resumen tiene como propósito facilitar la comprensión del flujo técnico, sin mostrar el código completo ni comprometer el acceso a las bases anonimizadas de la organización, el cual puede consultarse bajo solicitud mediante el enlace restringido al cuaderno de Google Colab.

7. CONCLUSIONES Y TRABAJOS FUTUROS

El objetivo del proyecto fue identificar un modelo de aprendizaje automático capaz de medir y estratificar el riesgo en salud de más de cinco millones de afiliados de una aseguradora, con el fin de facilitar su gestión efectiva. El análisis comparativo de clasificadores supervisados permitió seleccionar al modelo *XGBoost* sin balancear como la alternativa con mejor desempeño para replicar la lógica de estratificación actual de la empresa.

Este modelo fue capaz de aprender las reglas de clasificación institucionales con precisión, alcanzando métricas perfectas (AUC = 1.0) en todos los grupos de riesgo y severidad tanto en la base de entrenamiento como en una base nueva correspondiente a otro periodo (2023T4). No obstante, también se observó que la versión balanceada del modelo *XGBoost* ofrecía resultados equivalentes, lo cual sugiere que desde el punto de vista técnico cualquiera de los dos modelos podría ser implementado. Esta decisión debe ser discutida con la empresa, en función de las capacidades computacionales y del proceso organizacional de toma de decisiones.

Otros modelos, como el *Random Forest*, también mostraron un rendimiento destacado, y pueden ser utilizados para complementar y perfeccionar los criterios de clasificación. La principal limitación del proyecto fue la demanda de recursos computacionales para la ejecución de los algoritmos; sin embargo, esta fue resuelta de manera sostenible mediante el uso de Google Colab Plus.

Un aspecto relevante del proyecto es que demuestra cómo un modelo determinista, basado en reglas de clasificación predefinidas por expertos, puede ser replicado mediante aprendizaje automático siempre que la información de entrada esté debidamente estructurada y etiquetada. Esta capacidad de la máquina para "aprender" decisiones institucionales codificadas previamente pone en evidencia la utilidad de los algoritmos supervisados en entornos donde ya existen criterios operativos bien definidos. En ese

sentido, se considera que el objetivo general del proyecto se cumple: se identificó y evaluó un modelo que permite clasificar niveles de riesgo en salud, alineado con el enfoque actual de la organización, con potencial de ser implementado como herramienta de apoyo para la gestión poblacional en salud.

Desde el inicio del proyecto, los autores no contaban con experiencia previa en aprendizaje automático supervisado ni en el uso de plataformas colaborativas como Google Colab Pro. A lo largo del proceso, adquirieron competencias técnicas para entrenar, evaluar y comparar distintos algoritmos de clasificación, reconociendo tanto su potencial como sus limitaciones. Aunque los resultados del modelo *XGBoost* sin balancear fueron sobresalientes, se tomaron precauciones metodológicas como la validación con datos de otro periodo, lo que permitió comprobar que el modelo generaliza adecuadamente. Esta experiencia permitió a los autores valorar la aplicabilidad real de estas herramientas en el entorno de salud y su posible integración con procesos institucionales existentes.

Para trabajos futuros, se contempla la posibilidad de presentar estos resultados al equipo de tecnología de la compañía, con el fin de explorar su eventual integración en los sistemas de gestión del riesgo. Este paso implicaría revisar los requisitos técnicos para el despliegue del modelo en ambientes productivos y su articulación con los procesos actuales de estratificación.

Asimismo, se vislumbran nuevas líneas de investigación, entre ellas el desarrollo de modelos que no solo clasifiquen el riesgo actual de un paciente, sino que estimen la probabilidad de transición entre niveles de riesgo en periodos futuros. Esto permitiría anticipar cambios en el estado de salud de los afiliados y diseñar intervenciones más oportunas desde una perspectiva predictiva y poblacional.

Dado que estos modelos están en constante evolución y generan cada vez más información, el siguiente paso será explorar técnicas de aprendizaje no supervisado que permitan identificar nuevas formas de segmentación de pacientes aún no contempladas.

8. REFERENCIAS BIBLIOGRÁFICAS

[1] Keralty, «Quiénes somos,» Keralty, 2024. [En línea]. Available: https://www.keralty.com/sobre-keralty. [Último acceso: mayo 2024].

[2] IBM, «What is supervised learning?,» IBM, 2024. [En línea]. Available: https://www.ibm.com/topics/supervised-learning. [Último acceso: mayo 2024].

[3] República de Colombia, «Artículo 178, Ley 100,» Congreso de la República, 1993. [En línea]. Available: https://www.suin-juriscol.gov.co/viewDocument.asp?ruta=Leyes/1635955. [Último acceso: junio 2024].

[4] MinSalud, «Resolución 1035 de 2022,» Ministerio de Salud y Protección Social, República de Colombia., 14 junio 2022. [En línea]. Available: https://www.minsalud.gov.co/Normatividad_Nuevo/Resoluci%C3%B3n%20No.%201035%20de%202022.pdf. [Último acceso: mayo 2024].

[5] Keralty, «RiSe-K groups for patient's stratification according multimorbidity and severity,» Publicación interna, Bogotá D.C., 2024.

[6] A. L. Cox, Risk Analysis of Complex and Uncertain Systems, Springer, 2009.

[7] G. Rose, The Strategy of Preventive Medicine, Oxford University Press., 2001.

[8] World Health Organization (WHO). , Global Health Risks: Mortality and Burden of Disease Attributable to Selected Major Risks., 2009.

[9] J. S. A. R. F. E. J. e. a. Hughes, Clinical Risk Groups (CRGs): A classification system for risk-adjusted capitation-based payment and health care management. Medical Care., Medical , 2004.

[10] J. P. S. B. H. S. D. M. &. M. L. M. Weiner, Development and application of a population-oriented measure of ambulatory care case-mix., Medical Care, 1991.

[11] M. Porter, «What is value in health care,» *N Engl J Med,* vol. 363, nº 26, pp. 2477-2481, 2010.

[12] L. Eng Sing, K. Hui Li, H. Elaine Qiao-Ying, T. Sok Huang, W. Fang Yan, R. Bridget L., F. Martin y S. Moira, «Evaluation of the implementation of a large-scale quality improvement programme: protocol for a mixed-methods study,» *BMJ Open,* vol. 11, nº 5, p. e041219, 2021.

[13] Change healthcare, «Risk Adjustment - Frequently Asked Questions,» changehealthcare, 2024. [En línea]. Available: https://www.changehealthcare.com/insights/risk-adjustment-faq. [Último acceso: junio 2024].

[14] G. R. Alexánder, «Marco conceptual y legal sobre la gestión de riesgo en Colombia: Aportes para su implementación,» *Monitor estratégico,* nº 5, pp. 4-11, 2014.

[15] S. B. Kotsiantis, I. Zaharakis y P. Pintelas, Supervised machine learning: A review of classification techniques, Amsterdam: IOS Press, 2007.

[16] Q. J. Ross, «Induction of decision trees,» *Machine Learning,* vol. 1, pp. 81 - 106, 1986.

[17] L. Breiman, «Random forests,» *Machine learning,* vol. 45, pp. 5 - 32, 2001.

[18] D. Cox, «The regression analysis of binary sequences,» *Journal of the Royal Statistical Society Series B: Statistical Methodology,* vol. 20, nº 2, pp. 215 - 232, 1958.

[19] T. Cover y P. Hart, «Nearest neighbor pattern classification.,» *IEEE transactions on information theory,* vol. 13, nº 1, pp. 21 - 27, 1967.

[20] C. Cortes y V. Vapnik, «Support-vector networks.,» *Machine learning,* vol. 20, pp. 273 - 297, 1995.

[21] T. Chen y C. Guestrin, «Xgboost: A scalable tree boosting system.,» *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining,* 2016.

[22] M. Sokolova y G. Lapalme, «A systematic analysis of performance measures for classification tasks.,» *Information processing & management,* vol. 45, nº 4, pp. 427 - 437, 2009.

[23] C. Luo, Y. Zhu, Z. Zhu, R. Li, G. Chen y Z. Wang, «A machine learning-based risk stratification tool for in-hospital mortality of intensive care unit patients with heart failure,» *Journal of translational medicine,* vol. 20, nº 1, p. 136, 2022.

[24] H. Goel y D. Kumar, «Data Mining in Healthcare using Machine Learning Techniques,» *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS). IEEE,,* pp. 25 - 29, 2023.

[25] V. Caballer-Tarazona, N. Guadalajara-Olmeda y D. Vivas-Consuelo, «Predicting healthcare expenditure by multimorbidity groups,» *Health Policy,* vol. 123, nº 4, pp. 427 - 434, 2019.

[26] L. D. Liao, A. Ferrara, M. B. Greenberg, A. L. Ngo, J. Feng, Z. Zhang, P. T. Bradshaw, A. E. Hubbard y Y. Zhu, «Development and validation of prediction models for gestational diabetes treatment modality using supervised machine learning: a population-based cohort study,» *BMC medicine,* vol. 20, nº 1, p. 307, 2022.

[27] D. K. Ming, B. Hernandez, S. Sangkaew, N. L. Vuong, P. K. Lam, N. M. Nguyet, D. T. H. Tam, D. T. Trung, N. T. H. Tien, N. M. Tuan, N. V. V. Chau, C. T. Tam, H. Q. Chanh y H. Trieu, «Applied machine learning for the risk-stratification and clinical decision support of hospitalised patients with dengue in Vietnam,» *PLOS Digital Health,* vol. 1, nº 1, p. e0000005, 2022.

9. ANEXOS

9.1 Flujo técnico resumido de los modelos y generalidades de su código

El desarrollo completo del código fuente, incluyendo la carga de datos, procesamiento, entrenamiento de cada modelo, evaluación y validación externa, se encuentra disponible en el siguiente cuaderno de Google Colab:

A continuación, el flujo en un cuaderno de tipo Jupyter notebbok en Google colab para Python:

9.2 Modelo XGBOOST Sin Balancear

Enlace al cuaderno:
https://colab.research.google.com/drive/1QQ9UlirQ2AnQ4Nj462bcraiuH2l1uEJi?usp=sharing

Este cuaderno tiene acceso restringido por razones de confidencialidad de datos. Cualquier persona interesada en revisarlo podrá solicitar acceso al autor principal del trabajo. No obstante, para facilitar la comprensión del proceso, se ha incluido en este apéndice un resumen detallado de los pasos ejecutados y del código más relevante.

1. Importación de librerías

Scikit-learn es una biblioteca de Python ampliamente usada para aprendizaje automático. En este proyecto se utilizó para los diferentes algoritmos de aprendizaje supervisado:

```
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

Se cargan librerías para manipulación de datos, modelado, métricas, visualización y codificación de variables categóricas.

2. Carga de datos de entrenamiento (2022-T3)

```
df = pd.read_csv("datos_entrenamiento_2022_T3.csv")
```

Se carga la base estructurada correspondiente al tercer trimestre de 2022, previamente anonimizada y etiquetada por la organización.

3. Revisión y tratamiento de datos faltantes

df.isnull().sum()

df.fillna("No Aplica", inplace=True)

Se identifican los atributos con valores nulos. En variables relacionadas con control de enfermedades, los valores faltantes se reemplazan por "No Aplica", preservando registros valiosos sin distorsionar el modelo.

4. Transformación de variables categóricas a numéricas

le = LabelEncoder()

df["columna_codificada"] = le.fit_transform(df["columna_categorica"])

Se convierte información categórica (como estado de control de enfermedades) en valores numéricos necesarios para el entrenamiento del modelo.

5. Separación de datos en entrenamiento y prueba

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

División de la base en conjuntos de entrenamiento y prueba con partición aleatoria estratificada para garantizar representatividad.

6. Entrenamiento del modelo XGBoost

modelo = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', max_depth=10, random_state=42)

modelo.fit(X_train, y_train)

Entrenamiento del modelo con el algoritmo XGBoost, configurado para clasificación multiclase con la métrica mlogloss (log-loss multiclase). Se estableció una profundidad máxima de 10, se desactivó el codificador interno de etiquetas y se definió una semilla para reproducibilidad. Los hiperparámetros fueron definidos manualmente según criterios prácticos.

7. Evaluación del modelo con datos de prueba (2022-T3)

y_pred = modelo.predict(X_test)

print(classification_report(y_test, y_pred))

Se evalúan métricas como accuracy, precisión, recall y F1-score para comprobar el rendimiento inicial del modelo.

8. Matriz de confusión

sns.heatmap(confusion_matrix(y_test, y_pred), annot=False, cmap="Blues")

Visualización del comportamiento del modelo por clase. Ayuda a identificar posibles errores sistemáticos.

9. Curva ROC (evaluación multiclase)

fpr, tpr, thresholds = roc_curve(...)

plt.plot(fpr, tpr)

Evaluación complementaria del rendimiento del modelo, útil para determinar la discriminación entre clases.

10. Cálculo de ganancia por variable

importancia = modelo.get_booster().get_score(importance_type='gain')

Se calcula la ganancia por variable, que representa la mejora promedio en el modelo al dividir con cada característica. Permite identificar las variables más relevantes.

11. Carga de datos de validación (2023-T4)

df_valid = pd.read_csv("datos_validacion_2023_T4.csv")

Se carga la nueva base de datos correspondiente al cuarto trimestre de 2023. Esta base se utiliza en su totalidad como conjunto de validación externa.

12. Predicción y evaluación sobre la base 2023-T4

y_pred_2023 = modelo.predict(X_valid)

print(classification_report(y_valid, y_pred_2023))

Se aplica el modelo previamente entrenado sobre la nueva base. Las métricas confirman su estabilidad y capacidad de generalización a datos de otro período.

9.3 Modelo XGBOOST Balanceado

Enlace al cuaderno:
https://colab.research.google.com/drive/1SyYrrAXrxl2x72iEA8YYz0WiQJwLUfzN?usp=sharing

Este cuaderno tiene acceso restringido por razones de confidencialidad de datos. Cualquier persona interesada en revisarlo podrá solicitar acceso al autor principal del trabajo.

Sobre este modelo balanceado vale la pena aclarar que se probó también una versión del modelo XGBoost balanceada al tamaño de la clase minoritaria. Aun con este ajuste, el modelo mantuvo un buen desempeño en todas las clases, lo que indica que logra aprender adecuadamente la lógica de clasificación, incluso cuando se corrige el desbalance natural de la base de datos.

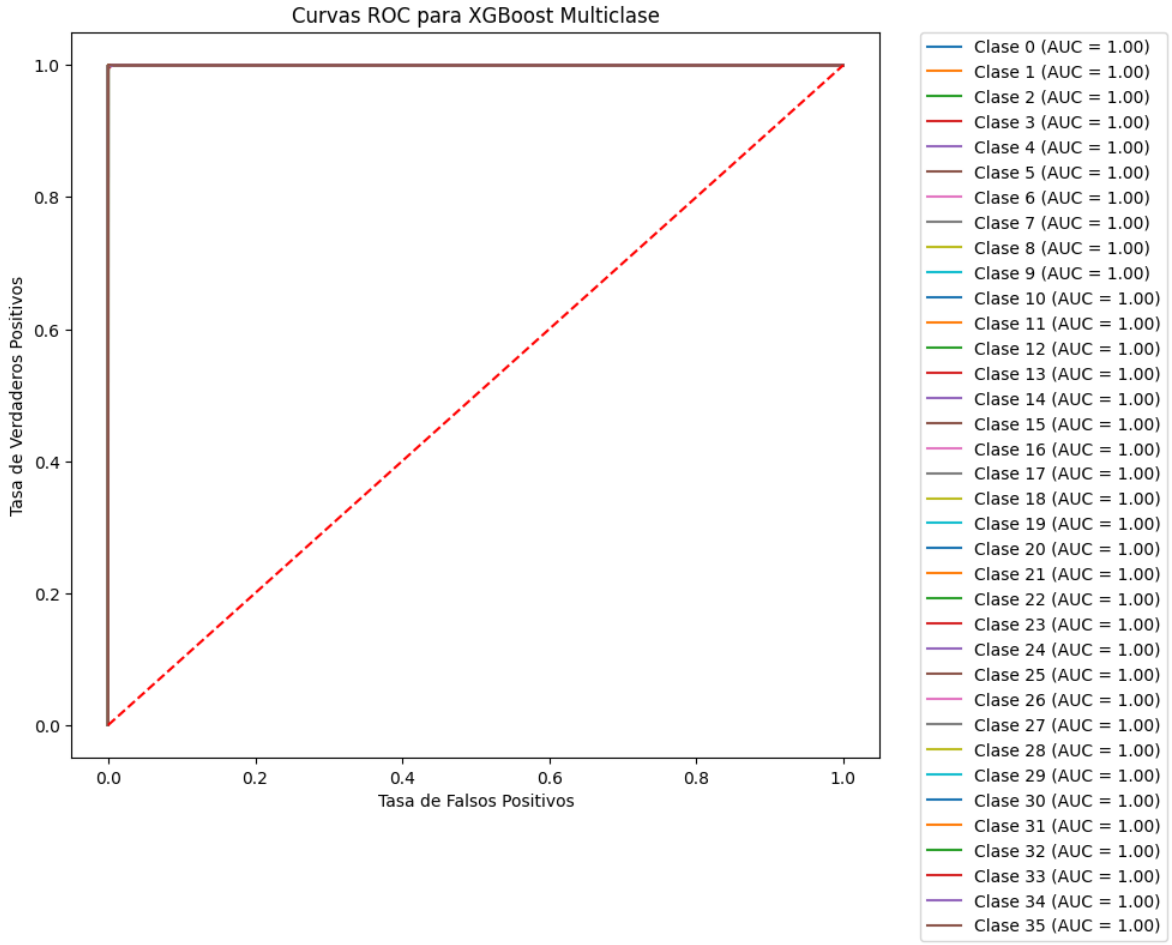


Ilustración 3 Curva ROC XGBoost balanceado

Para este código, lo único que cambia respecto al desbalanceado es que entre las secciones 4 y 5 va el balanceo de clases al tamaño de la menor clase:

```
from sklearn.utils import resample

# Concatenar datos y etiquetas para facilitar el balanceo
```

```
data_balanceo = pd.concat([X, y], axis=1)

# Agrupar por clase y aplicar muestreo aleatorio estratificado hacia la menor clase

clase_min = data_balanceo["RiesgoSeveridad"].value_counts().min()

df_balanceado = data_balanceo.groupby("RiesgoSeveridad", group_keys=False).apply(lambda x:
x.sample(clase_min, random_state=42))

# Separar nuevamente X e y

X = df_balanceado.drop("RiesgoSeveridad", axis=1)

y = df_balanceado["RiesgoSeveridad"]
```

Se aplicó un muestreo estratificado hacia la clase minoritaria para construir una base balanceada, asegurando que todas las clases tengan el mismo número de registros.

9.4 Random Hold-out XGBOOST

Enlace al cuaderno (Sin balancear):

https://colab.research.google.com/drive/1nRA5tQyPPrwdg11fTwnSyJ6eee2SPTzi?usp=sharing

Enlace al cuaderno (Balanceado):

https://colab.research.google.com/drive/1TsVyGCZCyDCXDSGZ7pZP_fk9Oz4Hsuxf?usp=sharing

Estos cuadernos tienen acceso restringido por razones de confidencialidad de datos. Cualquier persona interesada en revisarlo podrá solicitar acceso al autor principal del trabajo.

El procedimiento a realizar fue el siguiente y parte de la base de lo planteado en el código presentado en el XGboost sin balancear y balanceado:

```
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, accuracy_score, f1_score
import numpy as np

accuracy_scores = []
f1_scores = []
precision_scores = []
recall_scores = []

for i in range(10):
    X_train, X_test, y_train, y_test = train_test_split(
        data2.drop('riesgoseveridad_numerica_xboost', axis=1),
        data2['riesgoseveridad_numerica_xboost'],
```

```
    test_size=0.2,
    random_state=i
  )

  xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', random_state=42)
  xgb_model.fit(X_train, y_train)
  y_pred = xgb_model.predict(X_test)

  acc = accuracy_score(y_test, y_pred)
  f1 = f1_score(y_test, y_pred, average='macro')
  report = classification_report(y_test, y_pred, output_dict=True)
  precision = report['macro avg']['precision']
  recall = report['macro avg']['recall']

  accuracy_scores.append(acc)
  f1_scores.append(f1)
  precision_scores.append(precision)
  recall_scores.append(recall)

print(f"Accuracy promedio: {np.mean(accuracy_scores):.4f}")
print(f"F1-Score promedio: {np.mean(f1_scores):.4f}")
print(f"Precisión promedio: {np.mean(precision_scores):.4f}")
print(f"Recall promedio: {np.mean(recall_scores):.4f}")
```

Para verificar la estabilidad del modelo, se aplicó una validación tipo random hold-out repetida diez veces. En cada iteración, se dividió la base en entrenamiento (80%) y prueba (20%) con una semilla aleatoria distinta, y se entrenó un nuevo modelo XGBoost, tanto para el caso balanceado como el no balanceado. Se calcularon las métricas principales (accuracy, F1-score, precisión y recall), y al finalizar se obtuvieron los promedios y desviaciones estándar de cada métrica. Esta estrategia, aplicada tanto al modelo balanceado como al no balanceado, permitió observar su rendimiento bajo distintas particiones y comprobar que el comportamiento del modelo es estable y consistente frente a cambios en la distribución de los datos.

9.5 Modelo Random Forest

Enlace al cuaderno (Sin balancear):

https://colab.research.google.com/drive/1yKbbHQkhu3ngO3y8MggIzLA5j41zrIeN?usp=sharing

Enlace al cuaderno (Balanceado):

https://colab.research.google.com/drive/1bYGGwxPz2L0qlZX5j6r8VZYBJNrWsuz3?usp=sharing

A continuación, las aclaraciones sobre el procedimiento realizado en los modelos de Random forest balanceado y no balanceado. El balanceo es el mismo que el presentado en la explicación previa del XGBoost, no obstante respecto a la parte del Entrenamiento del modelo Random Forest, se aclara que se usó el siguiente código:

```
modelo = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42, n_jobs=-1)
modelo.fit(X_train, y_train)
```

Entrenamiento del modelo utilizando un clasificador Random Forest con 100 árboles y profundidad máxima de 10, ajustado previamente para clasificación multiclase. Se utilizó n_jobs=-1 para aprovechar todos los núcleos del procesador y acelerar el entrenamiento.

A diferencia de XGBoost, no se importa con un alias como xgb y todo el flujo usa directamente el nombre RandomForestClassifier del módulo sklearn.ensemble.

En lo relacionado a las métricas de desempeño, se destaca que la curva ROC multiclasificación para el modelo Random Forest sin balancear muestra una alta capacidad de discriminación en la mayoría de las clases, con AUC superiores a 0.90 en casi todas. Sin embargo, se observan algunas clases con AUC bajos (por ejemplo, clases 6, 12 y 30), lo cual indica menor precisión en esas categorías específicas. Esto refleja el impacto del desbalance de clases en el desempeño del modelo.

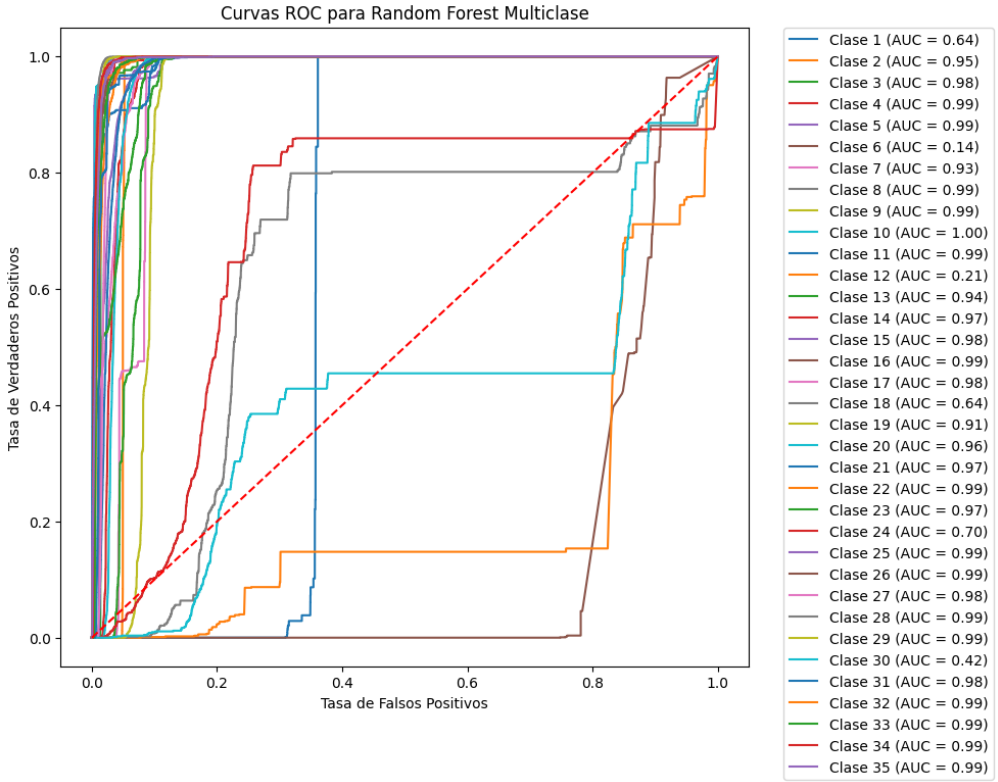


Ilustración 4 Random Forest Sin Balancear

La curva ROC del modelo Random Forest balanceado muestra una mejora general en las clases con bajo desempeño respecto al modelo sin balancear, aunque algunas clases continúan con AUC bajos (como la 6, 12 y 30). En general, el balanceo contribuye a una representación más equitativa entre clases, aumentando la capacidad del modelo para distinguir categorías minoritarias, aunque persisten retos en clases particularmente complejas.

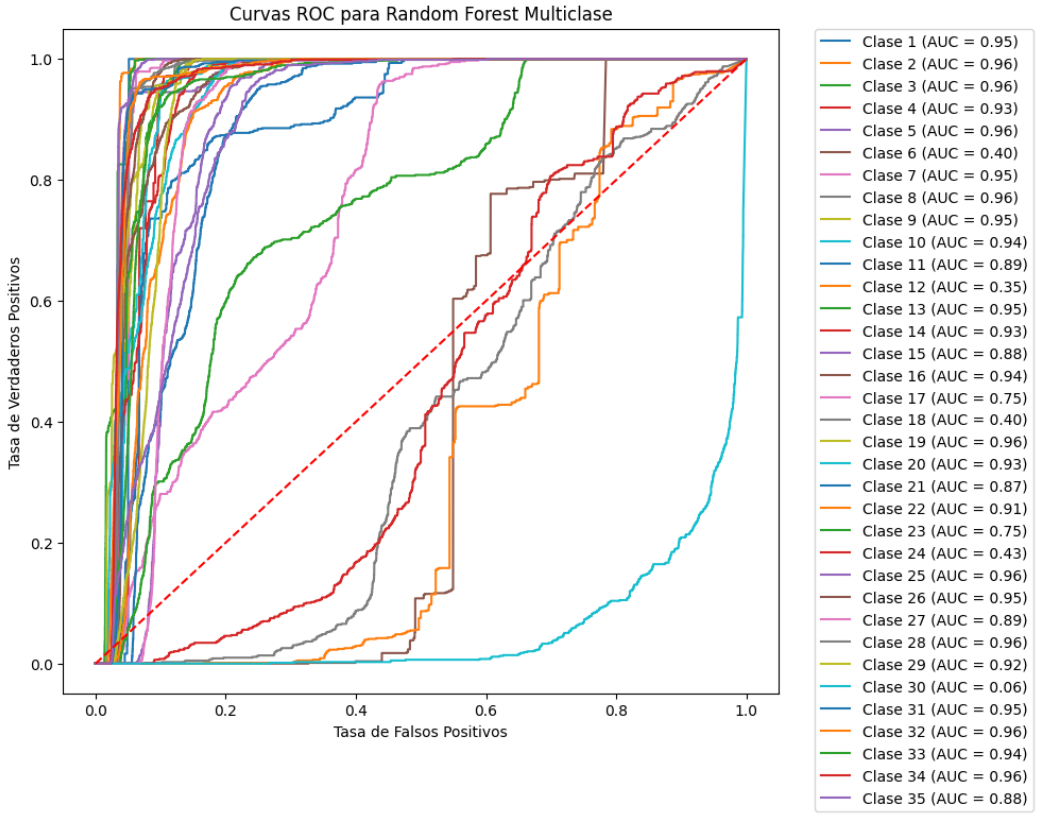


Ilustración 5 Random Forest balanceado

Respecto a lo relacionado con el random holdout, se usó el mismo paso a paso que en XgBoost pero con los pasos respectivos de los modelos corridos

Enlace al cuaderno (Sin balancear):

https://colab.research.google.com/drive/1AnD2chJayFbq2gK2XWY3Pdo-jpgPNF4q?usp=sharing

Enlace al cuaderno (Balanceado):

https://colab.research.google.com/drive/1DjOpb3O8t3DTueMe8t9V3qPk7iyp2AG0?usp=sharing

9.6 Modelo Decision Tree

Enlace al cuaderno (Sin balancear):

https://colab.research.google.com/drive/1z1lVQlZMBeCsAKG45li8rPD-sr_jFJAf?usp=sharing

Enlace al cuaderno (Balanceado):

https://colab.research.google.com/drive/1LLSEHd2NHH9OfRVSd50jY47tOy-6Imq9?usp=sharing

A continuación, se describen los aspectos diferenciadores en la implementación de los modelos de Árbol de Decisión balanceado y no balanceado. El procedimiento de balanceo es exactamente el mismo que el aplicado en los modelos anteriores (XGBoost y Random Forest), mediante muestreo estratificado al tamaño de la clase minoritaria.

Entrenamiento del modelo utilizando un clasificador tipo Árbol de Decisión con una profundidad máxima de 10 y una semilla aleatoria para asegurar la reproducibilidad de resultados.

```
from sklearn.tree import DecisionTreeClassifier

modelo = DecisionTreeClassifier(max_depth=10, random_state=42)

modelo.fit(X_train, y_train)
```

En términos de desempeño, la curva ROC para el modelo sin balancear evidencia un rendimiento variable entre clases. Algunas categorías muestran AUC altos (por encima de 0.90), pero se observan muchas otras con AUC bajos (como las clases 1, 6, 12, 24 y 30), lo que indica limitaciones al clasificar clases minoritarias o complejas.

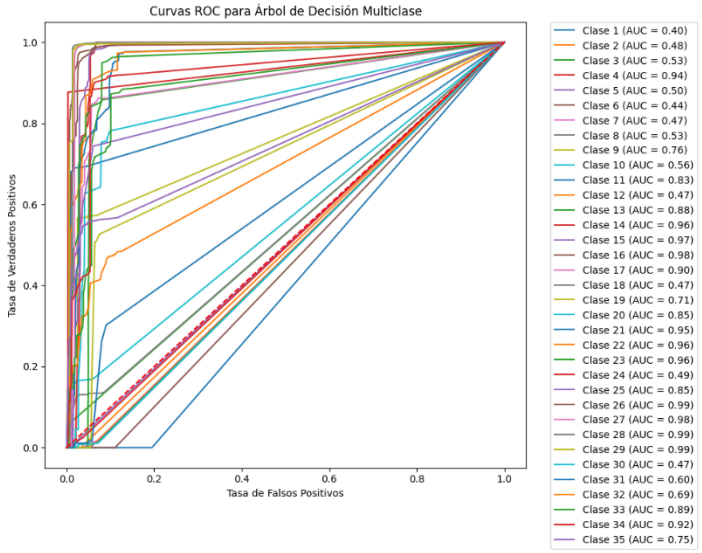


Ilustración 6 Decision Tree sin balancear

Al aplicar balanceo de clases, se observa una mejora en varias de las clases más débiles. Sin embargo, algunas categorías continúan con desempeño bajo, lo cual puede estar asociado a la limitada capacidad del algoritmo para capturar relaciones complejas en datos con alta dimensionalidad o desequilibrios naturales muy marcados.

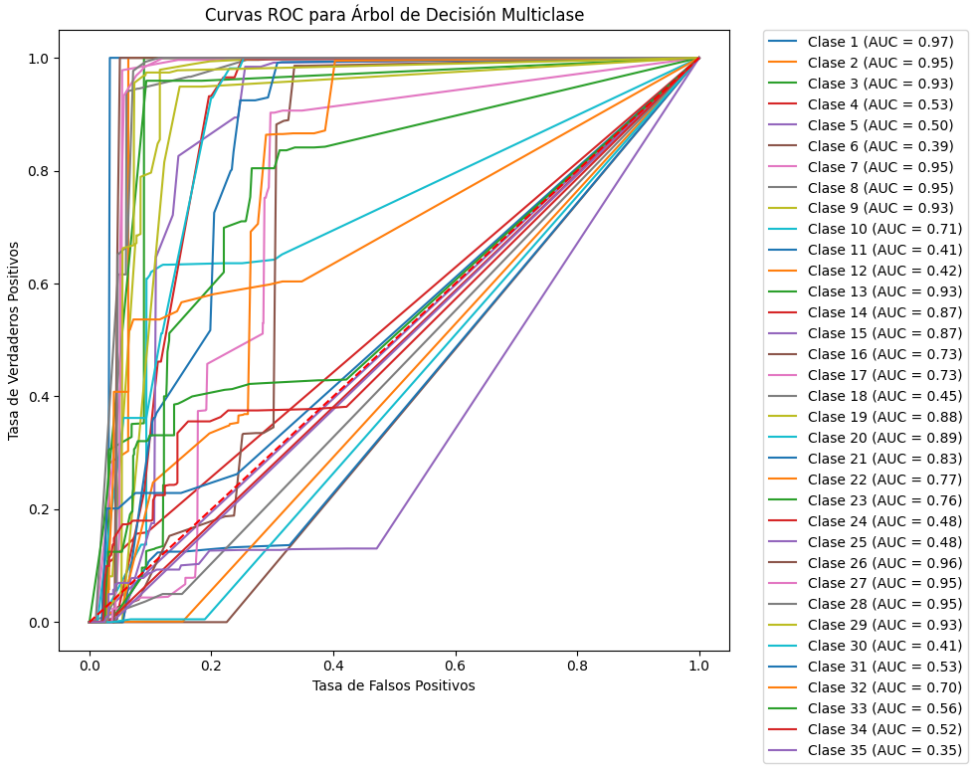


Ilustración 7 Decision Tree Balanceado

Respecto a lo relacionado con el random holdout, se usó el mismo paso a paso que en XgBoost pero con los pasos respectivos de los modelos corridos

Enlace al cuaderno (Sin balancear):

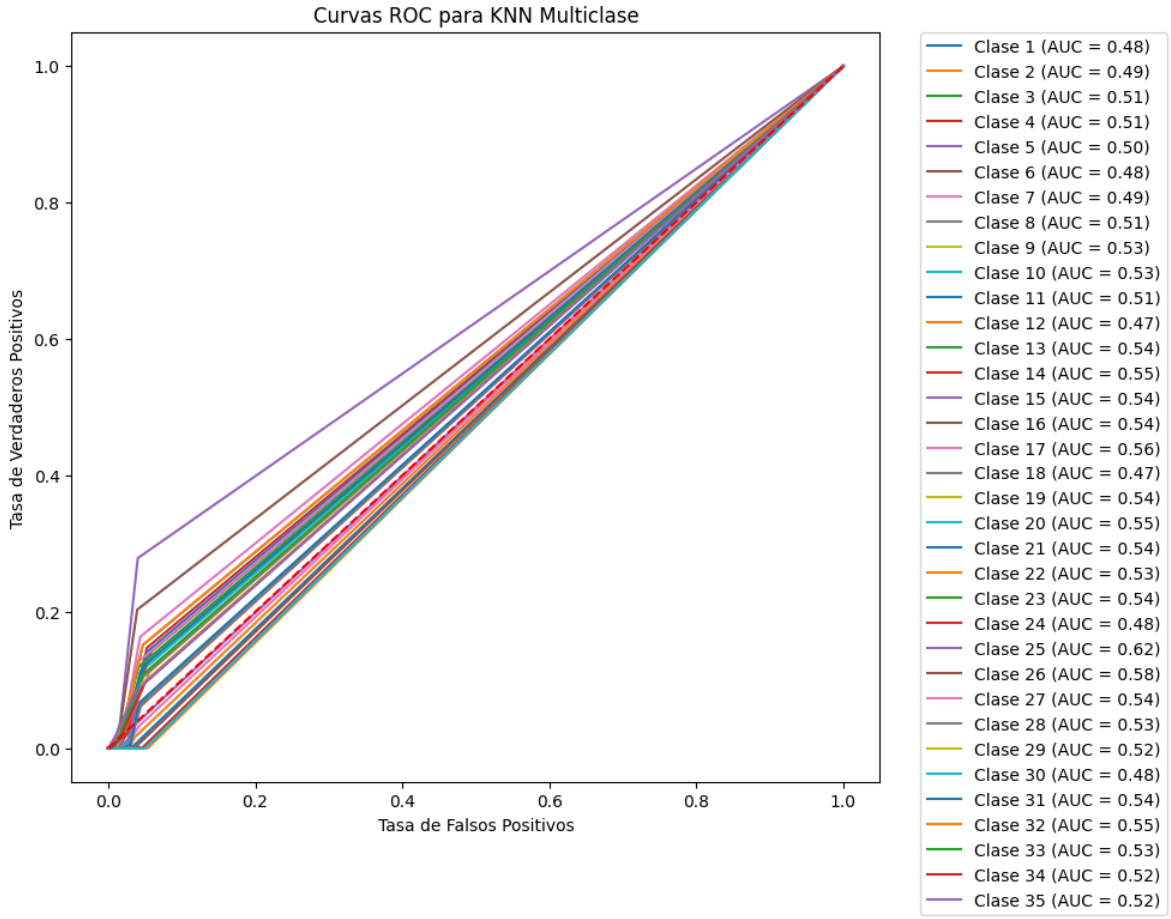
https://colab.research.google.com/drive/1RzCPg0Uy9oS0mUfeP4X4YQfgikL4IS2g?usp=sharing

Enlace al cuaderno (Balanceado):

https://colab.research.google.com/drive/1eQCBhQskg-5TiU381-SA3IkhqCwxfiAi?usp=sharing

9.7 Modelo KNN

Enlace al cuaderno (Balanceado):
https://colab.research.google.com/drive/1FOAP6LNRS1FhpWov8i0wVj5WuP7Rqznn?usp=sharing



Debido a las limitaciones de procesamiento computacional y al elevado tamaño de la base de datos original, no fue posible ejecutar satisfactoriamente el modelo KNN sobre la base desbalanceada. El algoritmo requería altos recursos para calcular distancias entre millones de registros, lo que resultó en fallas de ejecución. Por ello, se optó por evaluar únicamente una versión balanceada del modelo. Aun así, los resultados obtenidos muestran desempeños bajos en general, con curvas ROC planas y AUC cercanos a 0.5 en casi todas las clases, indicando que el modelo no logra distinguir adecuadamente entre categorías. Dado este bajo rendimiento, no se implementó validación adicional tipo random holdout para este modelo.

9.8 Modelo Logit

Enlace al cuaderno (Sin balancear):

https://colab.research.google.com/drive/1-5x5DLECd5TK1yn-FYFKuIHxk7OhglqT?usp=sharing

Enlace al cuaderno (Balanceado):

https://colab.research.google.com/drive/1R8Bx38BsZzq6_a7s7fPKKTWazAdG9_6X?usp=sharing

Este modelo fue implementado para evaluar la capacidad de la regresión logística en la clasificación multiclase de niveles de riesgo en salud. La versión sin balancear se entrenó directamente sobre la distribución original de clases, mientras que para la versión balanceada se aplicó el mismo procedimiento descrito previamente para otros modelos (muestreo estratificado hacia la clase minoritaria antes de la división en entrenamiento y prueba).

En cuanto al entrenamiento del modelo, se utilizó el siguiente código:

```
from sklearn.linear_model import LogisticRegression
modelo = LogisticRegression(max_iter=1000, solver='lbfgs', multi_class='multinomial')
modelo.fit(X_train, y_train)
```

Se realizó el entrenamiento del modelo de regresión logística con ajuste para clasificación multiclase mediante la opción multi_class='multinomial', utilizando el algoritmo lbfgs y un máximo de 1000 iteraciones para asegurar la convergencia del modelo. En lo relativo al desempeño, se observa que el modelo balanceado presentó mejores resultados en algunas clases que eran menos representadas, mostrando mejoras en sus valores AUC. Sin embargo, persisten algunas clases con bajo rendimiento (por ejemplo, clases 6, 12 y 30), lo cual sugiere que, aunque el balanceo mejora la representación global, la regresión logística tiene limitaciones cuando se enfrenta a clases con patrones no lineales o poca separación.

La primera gráfica muestra la curva ROC multiclase para el modelo balanceado, con AUC mayor a 0.90 en varias clases, pero también presencia de valores por debajo de 0.50 en clases complejas.

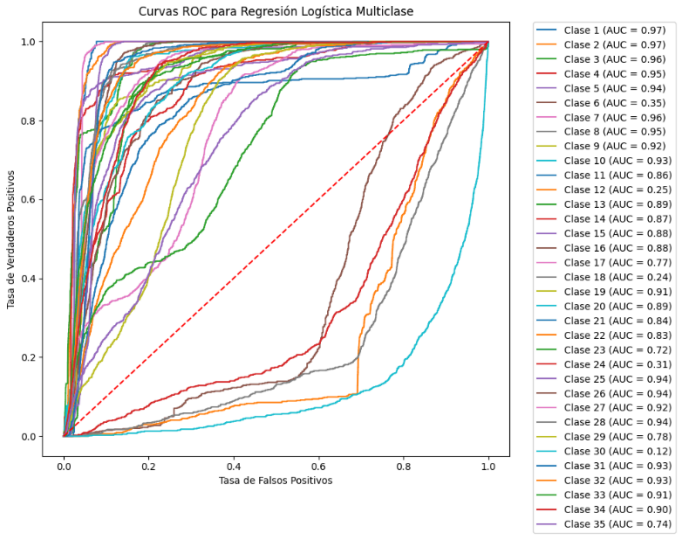


Ilustración 8 Logit balanceado

En la segunda imagen, correspondiente al modelo sin balancear, aunque se logra un buen rendimiento en algunas clases mayoritarias, se evidencia una pérdida significativa de precisión en

clases minoritarias (AUC < 0.40 en varias), reflejando el sesgo del modelo hacia las clases predominantes. Aun así, debido a su rendimiento aceptable y facilidad de entrenamiento, el modelo fue útil como línea base de comparación frente a algoritmos más complejos.

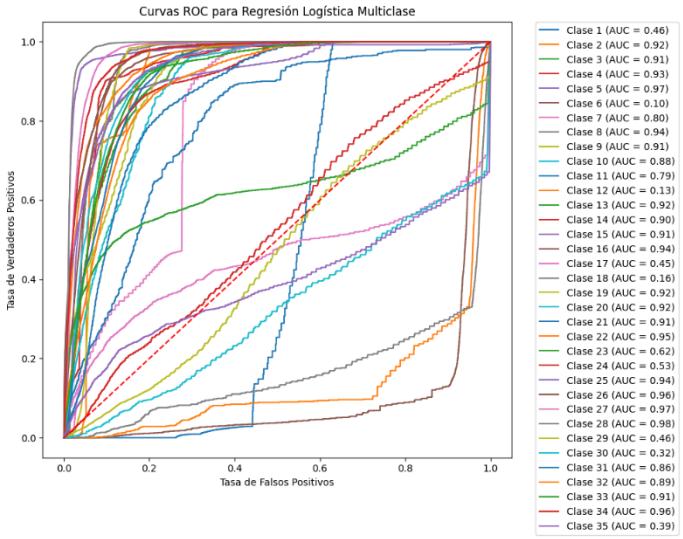


Ilustración 9 Logit sin balancear

Dado este rendimiento, tampoco se implementó validación adicional tipo random holdout para este modelo.

9.9 Modelo SVM

Enlace al cuaderno (balanceado):
https://colab.research.google.com/drive/1_QB3DqDOFmSapx4dsA9HZdAPJHBvhSa5?usp=sharing

El modelo de Máquina de Vectores de Soporte (SVM) se probó únicamente en su versión balanceada, debido a que la ejecución sobre la base desbalanceada no fue posible por las limitaciones de procesamiento asociadas al tamaño y complejidad de los datos. A pesar de su elevado costo computacional, el modelo balanceado logró desempeños aceptables, con métricas AUC superiores a 0.90 en la mayoría de clases, aunque con menor rendimiento en algunas categorías específicas como las clases 6, 12, 18 y 30. Por estos resultados, y considerando su exigencia en recursos y el tiempo de entrenamiento, este modelo no fue incluido en procesos de validación tipo random holdout.

Entrenamiento del modelo SVM balanceado:

```
from sklearn.svm import SVC
modelo = SVC(kernel='rbf', probability=True, random_state=42)
modelo.fit(X_train, y_train)
```

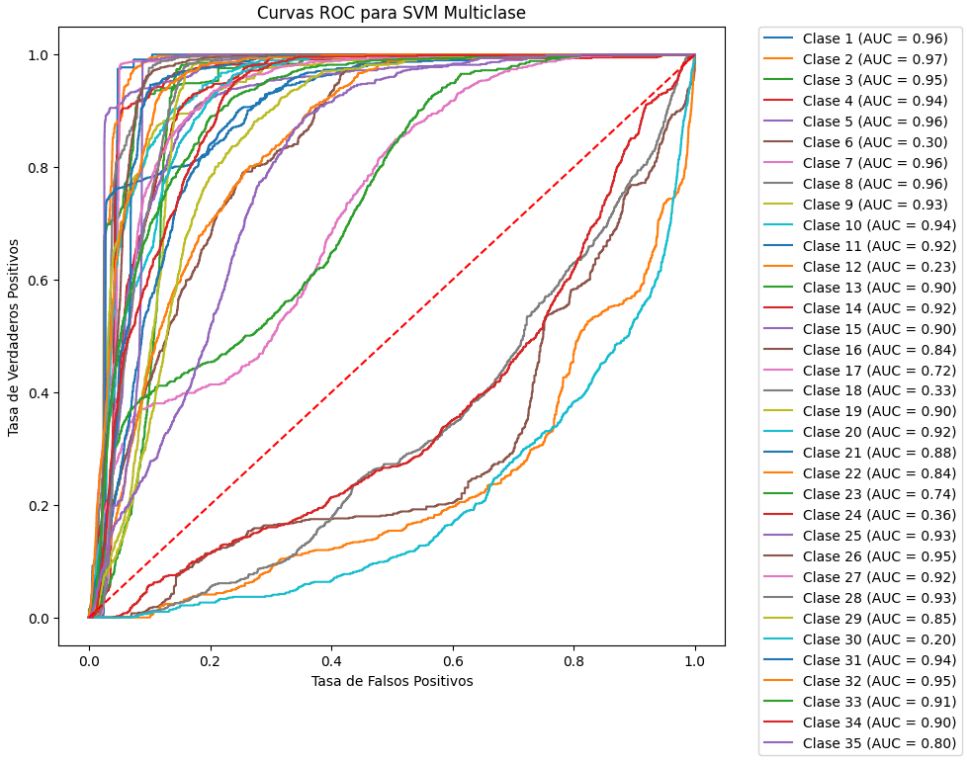


Ilustración 10 SVN balanceado

Se utilizó un modelo SVM con kernel RBF (función base radial), adecuado para capturar relaciones no lineales entre las variables. El parámetro probability=True fue necesario para calcular las curvas ROC por clase. El entrenamiento se realizó sobre la base previamente balanceada y codificada, con una semilla para asegurar reproducibilidad.

Fin del documento