UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

OPTIMIZACIÓN DE TÉCNICAS DE BALANCE DE DATOS PARA CLASIFICADOR DE CURVAS DE LUZ BASADO EN XGBOOST

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

VICENTE NOLASCO CARAGOL DÍAZ

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

MIEMBROS DE LA COMISIÓN:
IGNACIO REYES JAINAGA
FRANCISCO FÖRSTER BURÓN

Este trabajo fue financiado por ANID, Iniciativa Científica Milenio, ICN12_009 y Fondecyt 1220829.

SANTIAGO DE CHILE
2023

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: VICENTE NOLASCO CARAGOL DÍAZ
FECHA: 2023
PROF. GUÍA: PABLO ESTÉVEZ V.

OPTIMIZACIÓN DE TÉCNICAS DE BALANCE DE DATOS PARA CLASIFICADOR DE CURVAS DE LUZ BASADO EN XGBOOST

En estudios astronómicos se procesan altos volúmenes de datos de gran tamaño, por lo que para el procesamiento de éstos se utilizan algoritmos de aprendizaje de máquinas. Sin embargo, para el problema de clasificación de eventos astronómicos se presenta la problemática del desbalance en el número de datos que se tiene de cada tipo de evento, lo que puede provocar que algoritmos de clasificación presenten un sesgo en su desempeño. Por esto, en el presente trabajo se estudiaron distintas metodologías de balance de datos en conjunto a un clasificador *XGBoost* para afrontar esta problemática y disminuir el sesgo hacia los eventos más comunes al clasificar sus curvas de luz. Se analizó una relación entre dicho sesgo con la variación del parámetro de profundidad máxima del clasificador, observándose un menor sesgo cuando se disminuye dicho parámetro. Se estudió también la modificación de un algoritmo *Gradient Boosting* al implementar balance de datos en la construcción de sus árboles mediante *bootstrapping*, con lo que se pudo observar una mejor clasificación en las clases menos representadas. También se utilizaron datos de clase transiente generados sintéticamente para estudiar el desempeño de *XGBoost* con un conjunto de datos balanceados mediante esta metodología, lo que permitió una mejor clasificación de datos transientes en comparación a las otras metodologías utilizadas junto a *XGBoost*. Finalmente, los resultados obtenidos son comparados con los obtenidos con *Balanced Random Forest*, algoritmo utilizado por el clasificador de curvas de luz del broker ALeRCE.

Tabla de Contenido

1. Introducción

1.1. Motivación

La inteligencia artificial es un campo de estudio que abarca los diversos algoritmos destinados a hacer que una máquina muestre capacidades similares a las de un ser humano en términos como resolución de problemas y toma de decisiones. Así, se logra que la máquina adopte un comportamiento más próximo al de una entidad inteligente. Dentro de este ámbito se encuentra el concepto de aprendizaje de máquinas o *Machine Learning*, que se refiere a las estrategias mediante las cuales una máquina aprende a realizar una tarea específica sin seguir instrucciones explícitas proporcionadas por un humano, sino mediante un algoritmo ajustado a través del procesamiento de datos. Los algoritmos de inteligencia artificial y aprendizaje de máquinas se utilizan en diversas áreas, como la salud, las finanzas, la ciberseguridad y el comercio.

Otra área en donde es común el uso de algoritmos de aprendizaje de máquinas es la astronomía, en donde el aumento exponencial de datos recopilados por los telescopios en los sondeos astronómicos genera la necesidad de un avance continuo en la capacidad para analizar y clasificar dichos datos. Una dificultad en el análisis de datos astronómicos es la alta cantidad de alertas sobre posibles eventos astrofísicos detectadas por los sondeos, las cuales se generan al observar variaciones de luz en un punto específico. Por esta razón, existen intermediarios, denominados *brokers*, entre los telescopios y el resto de la comunidad astronómica que emplean algoritmos de aprendizaje de máquinas capaces de recibir datos de alertas y determinar a qué tipo de evento astronómico corresponden.

El *broker* ALeRCE [1]-[2]-[3] es una iniciativa chilena que procesa el flujo de alertas de la *Zwicky Transient Facility (ZTF)* y tiene como objetivo futuro procesar las alertas del sondeo *Legacy Survey of Space and Time (LSST)* del Observatorio Vera C. Rubin. Las alertas se clasifican en distintas categorías, divididas en fenómenos transientes, estocásticos y periódicos, mediante un modelo de predicción basado en *Random Forests*. En particular, se utiliza una variante de *Random Forests* llamada *Balanced Random Forest* o *BRF* [4], que permite abordar de manera eficiente uno de los principales desafíos en la clasificación de este conjunto de datos: El desbalance de instancias entre sus clases (Ver Figura 1).

Debido a que eventos de ciertas clases astronómicas tienen una probabilidad de ocurrencia diferente a otras, se tiene una cantidad distinta de instancias de cada clase en el conjunto de datos, lo que puede llevar a que un modelo de aprendizaje de máquinas presente un sesgo hacia la clase mayoritaria, ocasionando un mal desempeño al momento de clasificar instancias de las clases minoritarias, instancias que son, generalmente, más relevantes de identificar.

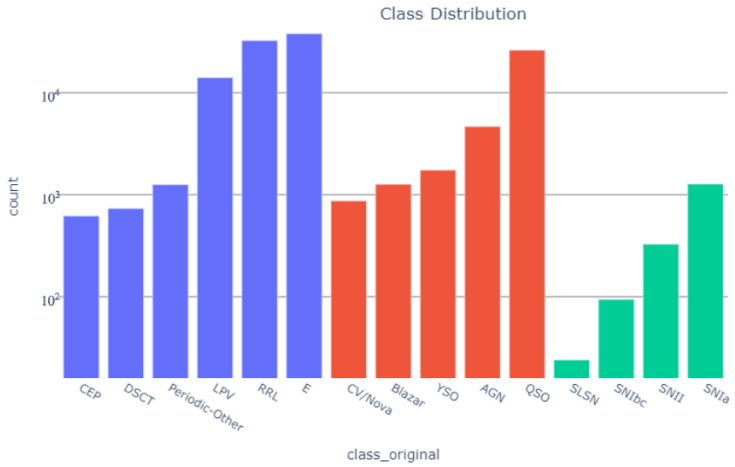


Figura 1: Distribución de números de ejemplos por clases de objetos astronómicos en el conjunto de datos utilizado.

Frente a esta problemática se han desarrollado distintas estrategias que tienen como objetivo disminuir el sesgo que se genera hacia la clase mayoritaria, ya sean modificaciones al algoritmo de aprendizaje de máquinas o disminución artificial del desbalance de clases en el conjunto de datos, ya sea mediante la reducción del número de instancias de la clase mayoritaria o el aumento de forma sintética del número de instancias de la clase minoritaria.

Entre estas estrategias se encuentra el algoritmo anteriormente mencionado *Balanced Random Forest*, que emplea un balance artificial de los datos en distintas iteraciones para lograr una clasificación con un sesgo notablemente menor hacia la clase mayoritaria. Por otro lado, otro modelo de aprendizaje de máquinas utilizado comúnmente para tareas de clasificación en distintas áreas es el denominado *Extreme Gradient Boost (XGBoost)*, el cual ha presentado un alto desempeño en el problema de clasificación de curvas de luz [5]. Sin embargo, se observa al analizar los resultados en distintas métricas de clasificación que este desempeño está sujeto a un sesgo hacia las clases mayoritarias, obteniéndose así un porcentaje alto de predicciones erróneas al identificar instancias de las clases minoritarias.

Las herramientas disponibles en *XGBoost* de forma nativa para afrontar el problema de sesgo en la clasificación son limitadas, por lo que en el presente trabajo se estudiará el uso de diferentes estrategias de balance de datos en conjunto con el modelo *XGBoost*.

1.2. Identificación del Problema

Una curva de luz corresponde al brillo en función del tiempo de un objeto astronómico determinado. Estas se construyen utilizando fotometría mediante la medición de la diferencia entre una observación determinada con una imagen de referencia en distintas bandas de frecuencia. La forma en que distintos intervalos de frecuencia se dividen en bandas varía, existiendo diferentes sistemas para definir dichas bandas. Uno de estos sistemas se denomina *ugriz* debido a que realiza

observaciones de luz en las bandas ultravioleta (u), verde (g), roja (r), cercano a infrarrojo (i) e infrarrojo (z) [6]. Para procesar el flujo de alertas de ZTF, ALeRCE hace uso de las bandas g y r [2].

Cabe notar que el brillo de un objeto astronómico puede medirse según la magnitud o según el flujo, siendo la primera una medición en escala logarítmica en la que un valor menor refleja un mayor brillo del objeto observado, mientras que el flujo corresponde a la cantidad de energía por unidad de área por unidad de tiempo. La relación entre estas dos mediciones es la siguiente:

$$m = -2{,}5log(\frac{F}{F_0}), \tag{1}$$

donde m corresponde a la magnitud aparente, F al flujo y F_0 a un flujo de referencia [7].

ALeRCE cuenta con un clasificador de curvas de luz [2] y un clasificador de *stamps* [3] con los que se procesa el flujo de alertas de ZTF utilizando una taxonomía definida según consideraciones astrofísicas (Ver Figura 2). Para el caso de las curvas de luz, su taxonomía se divide en 3 clases principales, cada una con un distinto número de sub-clases.

- Objetos periódicos: *Long-Period Variable* (LPV) que incluye estrellas variables regulares, semi-regulares, irregulares, *RR Lyrae* (RRL), *Cepheid* (CEP), *eclipsing binary* (E), δ *Scuti* (DSCT), y otras estrellas variables (Periodic-Other).
- Objetos estocásticos: *Active galactic nuclei* (AGN), quasar tipo 1 (QSO), blazar, *young stellar object* (YSO), y *cataclysmic variable/nova* (CV/Nova).
- Objetos transientes: Supernovas tipo Ia (SNIa), Ibc (SNIbc), II (SNII) y superluminosa (SLSN).

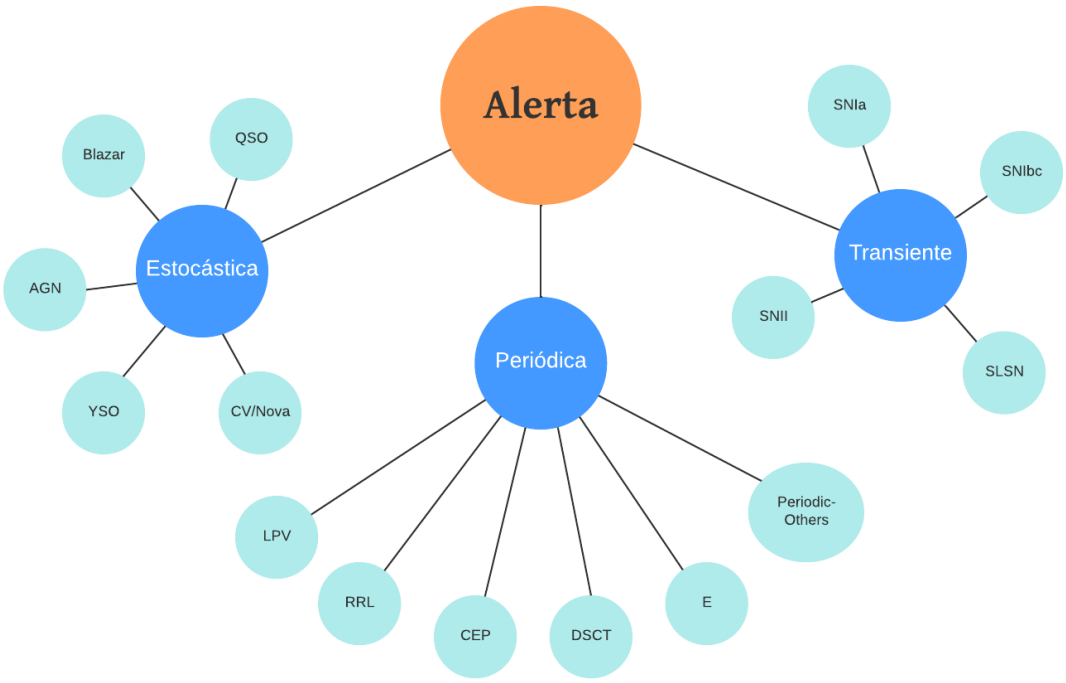


Figura 2: Taxonomía de clases utilizada por ALeRCE

La diferencia de probabilidad de ocurrencia de dichos eventos astronómicos conlleva un alto nivel de desbalance entre clases dentro del conjunto de datos utilizado por ALeRCE para el entrenamiento del clasificador de curvas de luz descrito en [2]. De un total de 123.496 curvas de luz, se tienen 87.065 instancias correspondientes a objetos periódicos, 34.713 a objetos estocásticos, y 1.718 a objetos transientes. Así también existe un alto porcentaje de desbalance de las ocurrencias de sub-clases dentro de una misma clase, teniendo un 43,5 % de instancias de la sub-clase mayoritaria y 0,7 % de la sub-clase minoritaria dentro de los objetos periódicos, mientras que para objetos estocásticos esta diferencia es de un 75,4 % en comparación con un 2,5 %, y un 74,0 % en comparación a un 1,4 % en objetos transientes.

De estas curvas de luz se extraen 152 características diferentes, que incluyen datos extraídos de observaciones de ZTF en las bandas g y r individualmente, características extraídas de ambas bandas, y metadatos que incluyen características como las coordenadas galácticas, colores obtenidos del catálogo *AllWISE*, entre otros.

El modelo de clasificación utilizado en ALeRCE corresponde a cuatro clasificadores basados en *Balanced Random Forest* separados en dos niveles. El nivel superior corresponde a un clasificador para predecir si un objeto pertenece a la clase periódica, estocástica, o transiente. Por otro lado, el nivel inferior consiste en tres clasificadores distintos, entrenados cada uno con los datos de una de las tres clases anteriormente mencionadas que tienen como objetivo predecir la sub-clase de un determinado objeto. Cabe notar que los cuatro clasificadores utilizan las mismas características extraídas de las observaciones.

Finalmente, se calculan las probabilidades de pertenecer a cada clase para un determinado objeto al multiplicar las probabilidades dadas por los clasificadores del nivel inferior con las probabilidades correspondientes dadas por el nivel superior. Con esto, denotando $P(\cdot)$, $P_T(\cdot)$ y $P_B()$ a la probabilidad final, la probabilidad del nivel superior, y la probabilidad del nivel inferior (dada por el clasificador correspondiente) respectivamente, se tendrá que la probabilidad de que un objeto pertenezca a la subclase s, correspondiente a la clase c, estará dada por:

$$P(s) = P_T(c) \cdot P_B(s|c). \tag{2}$$

El conjunto de datos de curvas de luz se construyó en base a etiquetas obtenidas de diversos catálogos que determinaron las clases de objetos astronómicos mediante análisis fotométrico y/o espectroscópico.

1.3. Objetivo General

El objetivo del presente trabajo es optimizar el desempeño del modelo de aprendizaje de máquinas llamado *Extreme Gradient Boost (XGBoost)* para la clasificación de curvas de luz mediante técnicas de balance de datos.

1.4. Objetivos específicos

Para desarrollar este trabajo, se proponen como objetivos específicos:

- Realizar una revisión bibliográfica para determinar el estado del arte en técnicas de entrenamiento de modelos de clasificación con conjuntos de datos desbalanceados, así como también del estado del arte sobre implementación y herramientas del modelo *XGBoost*.
- Analizar el desempeño de las técnicas de balance de datos estudiadas para clasificar las curvas de luz obtenidas del sondeo ZTF, al ser implementadas en un modelo *XGBoost* y *Balanced Random Forest*.
- Optimizar el desempeño de los modelos de predicción al determinar los mejores parámetros e implementaciones de distintas técnicas de balance de datos.
- Evaluar el uso combinado de técnicas de aumento de datos y de balance de datos, así como también técnicas de balance a nivel de algoritmo.

1.5. Organización del presente trabajo

En el capítulo 2 *Marco Teórico y Estado del Arte* se presenta una descripción de los conceptos teóricos relacionados a los modelos de aprendizaje de máquinas, particularmente los relacionados a modelos *Random Forest* y *XGBoost*. También se realiza una revisión de algunas de las principales técnicas del estado el arte para el problema de clasificación de datos desbalanceados. Posteriormente, en el capítulo 3 *Metodología* se describen los procesos y estrategias a utilizar para llevar a cabo los

objetivos del trabajo, especificando también detalles de la configuración seleccionada de los modelos a entrenar. En el capítulo 4 *Resultados y Análisis* se presentan los resultados obtenidos haciendo uso de la metodología anteriormente descrita, así como también una interpretación correspondiente. Posteriormente en base a estos resultados se realiza una prueba de los clasificadores seleccionados al ser utilizados para predecir datos actualizados de ALeRCE, a los que no se ha tenido acceso durante el proceso anterior de obtención de resultados y análisis. Finalmente, las conclusiones finales sobre las interpretaciones obtenidas de los resultados se presentan en el capítulo 5 *Conclusiones*.

2. Marco Teórico y Estado del Arte

2.1. Aprendizaje de Máquinas

El aprendizaje de máquinas corresponde a un área dentro del estudio de la inteligencia computacional que comprende el desarrollo de algoritmos que permiten a un computador "aprender" a resolver un problema determinado sin seguir acciones específicas programadas previamente. Esto se realiza mediante datos de entrenamiento con los que un modelo de aprendizaje de máquinas es capaz de desarrollar su propia estrategia para cumplir una tarea específica. Este tipo de algoritmos son utilizados en un amplio espectro de áreas como la medicina, procesamiento de imágenes, detección de fraudes, reconocimiento de lenguaje, etc.

Diversos modelos de aprendizaje de máquinas han sido desarrollados y estudiados para distintos tipos de necesidades y problemas. Entre estos modelos se encuentran las redes neuronales artificiales, árboles de decisión, máquinas de vectores de soporte (SVM), entre otros. Existen también algoritmos de aprendizaje de máquinas que consisten en variaciones o en uso de múltiples estructuras individuales de dichos modelos, como es el caso de *Balanced Random Forest* y *XGBoost*, que corresponden a algoritmos con estructuras basadas en múltiples árboles de decisión.

Los algoritmos de aprendizaje de máquinas tienen la ventaja de que debido a la capacidad de rápido procesamiento de datos de los computadores, un modelo entrenado puede tomar decisiones de forma rápida, especialmente al trabajar con datos de gran volumen. Este es el caso de gran parte de los problemas de procesamiento de datos relacionados con astronomía, debido a que mediante los sondeos realizados por telescopios se obtienen alta cantidad de datos cuyo análisis manual conllevaría una alta carga de tiempo.

Aunque el estudio de inteligencia artificial y aprendizaje de máquinas comprende algoritmos empleados para distintos fines, en este trabajo se hará énfasis en los modelos de clasificación supervisados, correspondientes a algoritmos que se entrenan en base a un conjunto de datos en el cual cada instancia corresponde a una determinada clase, para así procesar datos desconocidos y predecir a qué clase corresponde cada instancia.

2.2. Árboles de decisión

Los árboles de decisión corresponden a herramientas utilizadas en distintos ámbitos, principalmente administrativos, que consisten en "mapas" de decisiones secuenciales en forma de árbol diseñados para obtener un valor o acción de salida en función de un conjunto de características de entrada (Ver Figura 3). En el contexto de aprendizaje de máquinas se construyen árboles mediante algoritmos que permiten determinar la estructura de éste que permite una mejor clasificación de los datos. Se componen de un nodo inicial denominado *raíz* en el que se toma una decisión con respecto a los datos de entrada y se continúa hacia el nodo correspondiente al resultado de dicha decisión. Nodos siguientes a la raíz que contienen otra toma de decisión se denominan *ramas*, y en un punto se llega a ramas cuya respuesta determina una salida. Estos puntos que contienen la información de salida del árbol se denominan nodos terminales [8].

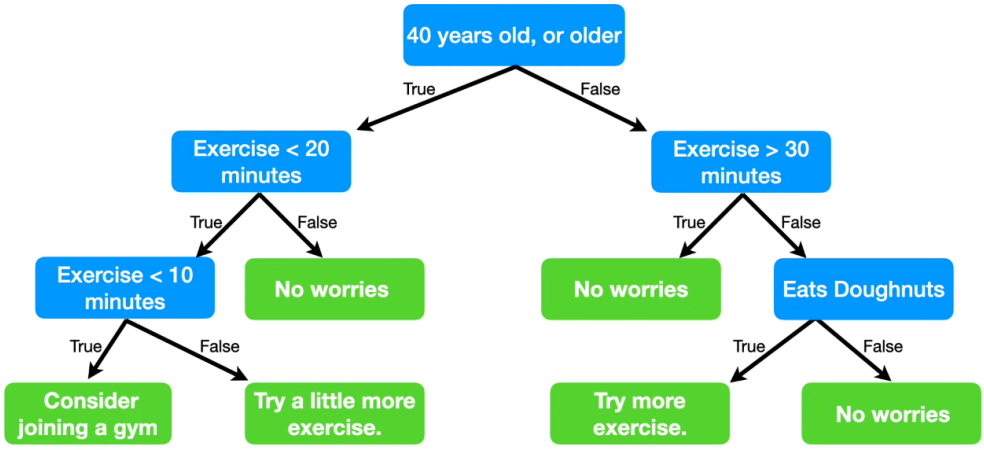


Figura 3: Ejemplo de árbol de decisión, en el cual se toma una decisión sobre si se debe considerar realizar actividad física en base a datos de entrada (características de una persona). *(Fuente: StatQuest)*

En el área de aprendizaje de máquinas, los árboles de decisión se utilizan para clasificar datos o realizar predicciones de valores. Para esto se entrena el modelo con el objetivo de que éste estructure las decisiones de cada nodo de forma que el árbol de decisión realice una predicción o clasificación efectiva de la salida a partir de los datos de entrada. Debido a que este algoritmo se entrena mediante el uso de datos cuya salida correspondiente es conocida, se denomina un modelo de *aprendizaje supervisado*.

Para construir un árbol de forma óptima, se utiliza el algoritmo *Classification and Regression Tree* o CART [9], que consiste principalmente en:

- Determinar el mejor punto de separación para cada variable de la entrada, es decir, el valor de una variable a partir del cual una entrada toma la decisión de pasar de una rama del nodo o a otra. Existen distintas métricas para determinar qué punto de separación es mejor, y varían para los árboles de clasificación (Árboles en que se desea clasificar un dato de entrada) y para los árboles de regresión (Árboles en que se predice un valor numérico a partir de la entrada). Para la construcción de árboles de clasificación, la métrica más utilizada es la *Impureza Gini* que mide la homogeneidad de la separación realizada por un nodo. Cabe notar que para el caso de una variable binaria solo se tiene una separación posible.
- Evaluar cuál variable presenta una mejor separación para el nodo mediante la métrica utilizada.
- El nodo de la raíz queda determinado a partir de la mejor separación obtenida entre todas las variables, y se realiza el mismo proceso en los nodos de las ramas siguientes.
- Se continúa recursivamente este proceso hasta cumplir con una regla de detención. Las reglas de detención se definen para evitar que el árbol continúe realizando separaciones hasta el punto de que haya una hoja para cada dato con el que se entrenó.

2.2.1. Impureza Gini

La impureza Gini [9] permite evaluar la homogeneidad de una separación para determinar la calidad de un nodo al computar la probabilidad de que un dato sea asignado a una clase de forma aleatoria en una determinada hoja, es decir, una impureza Gini menor implica que la separación considerada es más eficiente. Con esto, la impureza Gini en una hoja se define como:

$$G = 1 - \sum_{i}^{c}(p_i^2), \tag{3}$$

donde G corresponde a la impureza Gini de la hoja, c es el número de clases y p_i la probabilidad de que un dato clasificado aleatoriamente en dicha hoja pertenezca a la clase i. Esta probabilidad se determina considerando las instancias de cada clase que tiene la hoja al clasificar los datos de entrenamiento usando la separación de la rama anterior a la hoja.

Por otra parte, la impureza Gini de un nodo se determina ponderando las impurezas Gini de las ramas que las siguen, sean éstas hojas u otros nodos, por la cantidad de instancias que se tienen en cada una. La definición de la impureza Gini de un nodo se presenta en la siguiente ecuación:

$$G = \frac{n_1}{n}(G_1) + \frac{n_2}{n}(G_2), \tag{4}$$

donde n corresponde al número de instancias de datos totales presentes en el nodo, mientras que n_1 corresponde a las instancias de la rama izquierda del nodo y n_2 las de la rama derecha. Asimismo, G_1 corresponde a la impureza Gini de la rama izquierda mientras que G_2 corresponde a la de la rama derecha.

De esta forma, se evalúan las mejores separaciones siguiendo el algoritmo CART para construir el árbol de decisión hasta que se cumpla una regla de detención, las que suelen considerar criterios como un número mínimo de instancias que debe tener un nodo para contener una separación, definiéndose como una hoja si no cuenta con este mínimo. Estas reglas se definen para evitar que el árbol se ajuste para predecir de forma exacta los datos de entrenamiento, dado que esto reduce la eficacia del árbol para clasificar datos nuevos con los que no fue entrenado. Este problema se denomina sobreajuste u *overfitting*.

2.3. Random Forest

Una aplicación mas compleja de los árboles de decisión corresponde a los algoritmos *Random Forest*, que realizan clasificaciones o predicciones mediante el entrenamiento de múltiples árboles de decisión [10]. Cada árbol de decisión de entrena con un nuevo conjunto de datos denominado *bootstrap* que es construido al tomar datos aleatoriamente del conjunto original, con repetición. A su vez, para cada nodo de un árbol también se considera sólo un subconjunto de las variables de datos de entrada. Esto permite tener una alta variedad de árboles entrenados con el mismo objetivo, haciendo que el modelo general de *Random Forest* presente mayor robustez para clasificar datos desconocidos.

Para realizar una clasificación, el algoritmo de *Random Forest* obtiene la clasificación de cada árbol para el dato de entrada, y finalmente se determina qué clase recibió más "votos" de los árboles.

Este proceso se denomina *Bootstrap Agreggating* o *Bagging*.

2.3.1. Balanced Random Forest

Balanced Random Forest corresponde a una variación del algoritmo tradicional de *Random Forest* diseñada para enfrentar el problema de desbalance de clases existente en algunos conjuntos de datos [4]. Un conjunto de datos desbalanceado corresponde a un conjunto que contiene muchas instancias de una o más clases determinadas en comparación a otras. Esto usualmente ocasiona que modelos de aprendizaje de máquinas convencionales tiendan a estructurarse para predecir de manera precisa las clases mayoritarias pero no presente el mismo desempeño con clases minoritarias.

Con el fin de que los árboles de decisión en un modelo de *Random Forest* no presenten este sesgo a clases mayoritarias, el algoritmo de *Balanced Random Forest* construye los *bootstraps* para cada árbol de manera que el número de instancias de todas las clases sean las mismas. Para esto se añaden primeramente al *bootstrap* n instancias de datos aleatorias de la clase minoritaria, con repetición, donde n corresponde al número de instancias totales de esta clase. Posteriormente se añaden de la misma manera n instancias de cada clase mayoritaria, asegurando así que el *bootstrap* cuenta con la misma cantidad de instancias para cada clase y el árbol de decisión se entrena de manera balanceada.

Cabe destacar que al construir los *bootstraps* de esta manera, en cada árbol se pierde información de las clases mayoritarias, por lo que toma mayor importancia la necesidad de construir un alto número de árboles con el fin de contar con distintos *bootstraps* aleatorios de los datos de entrenamiento.

Este tipo de algoritmo es el utilizado por el *broker* ALeRCE en su clasificador de curvas de luz [2], para hacer frente al alto desbalance de clases presente en los datos de alertas astronómicas.

2.4. Gradient Boosting

La construcción de modelos de predicción es realmente un problema de estimación de función, en el que dado un espacio de n datos de dimensión k: $D = (x_i, y_i)(|D| = n, x_i \in \mathbb{R}^k, y_i \in \mathbb{R})$ y una función de pérdida $L(y, F(x))$, se define una función F^* tal que el valor de $L(y, F^*(x))$ sea mínimo [11], esta es:

$$F^*(x) = \arg\min_F E_{x,y} L(y, F(x)), \tag{5}$$

donde E corresponde al valor esperado.

La diferencia entre los diferentes modelos de predicción es la estructura que tiene la función F^* y cómo se define. Tanto para el caso de *Random Forest* como para *Gradient Boosting* esta función corresponde a la suma de distintas funciones correspondientes a los árboles de decisión del modelo, pero la principal diferencia entre ambos algoritmos es que *Random Forest* entrena árboles de forma paralela y con diferentes conjuntos de datos construidos a partir del original, mientras que *Gradient Boosting* entrena múltiples árboles de forma secuencial, considerando el rendimiento de los árboles previos al momento de construir un nuevo árbol.

En otras palabras, las funciones $F(x)$ que se pueden definir para un modelo de este tipo corresponden a combinaciones lineales, cuya forma general está dada por:

$$F(x) = -\sum_{m=0}^{M} \beta_m h(x, a_m), \tag{6}$$

donde M corresponde al número de árboles del modelo y $h(x, a_m)$ es la función que define un árbol individual con un vector de parámetros a_m. β_m corresponde a coeficientes del modelo denominados coeficientes de expansión que, al igual que los parámetros a_m, se definen durante el entrenamiento del modelo.

Existen distintas *loss functions* que pueden ser utilizadas para definir el objetivo (5), y la selección de ésta define la forma en que se realizará la clasificación por parte del modelo entrenado. En particular, para el caso de un problema de clasificación se utiliza generalmente la función de *log loss* o pérdida logarítmica. La función *log loss* se describe como sigue:

$$L_{log}(y_i) = -\sum_{j}^{c} y_{i,j} log(p_{i,j}), \tag{7}$$

donde c corresponde al número de clases diferentes, $y_{i,j} = 1$ si el dato i pertenece a la clase j o $y_{i,j} = 0$ si no pertenece, y $p_{i,j}$ corresponde a la probabilidad de que el dato i pertenezca a la clase j, que viene dada por la función F definida al momento de evaluar la función de pérdida.

Para entrenar un modelo de *Gradient Boosting*, el primer paso es realizar una predicción inicial, la que se determina minimizando el valor de la suma de las *loss functions*:

$$F_o(x) = \arg\min_{F} \sum_{i}^{N} L_{log}(y_i, F). \tag{8}$$

Habiendo obtenido una predicción inicial F_0 para cada clase, se pasa a construir los árboles del modelo. Esto se realiza mediante M iteraciones en las que en cada una se construye un árbol considerando los errores de predicción o residuos de la predicción anterior. Para este proceso, en cada iteración m se obtiene en primer lugar los "*pseudo-residuos*" $\tilde{y_i}$:

$$\tilde{y_i} = -\left[\frac{dL(y_i, F(X_i))}{dF(x_i)}\right]_{F(x)=F_{m-1}(x)}, \tag{9}$$

donde $F_m(x)$ es la notación para definir el árbol construido en la iteración m, siendo $F(x_i)$ el valor predicho por el árbol cuando toma como entrada a x_i. El caso específico de $F_0(x)$ corresponde a una predicción directa basada solo en la probabilidad de las clases. Cabe destacar que cuando se utiliza como *loss function* la función *log loss*, el valor de $\tilde{y_i}$ corresponde a $y_{i,j} - p_{i,j}$, es decir, será mayor mientras más lejana haya sido la predicción de probabilidad inicial al valor real de la clase del dato i. Habiendo obtenido un valor de *pseudo-residual* para cada dato, se procede a construir un árbol de regresión utilizando el algoritmo CART cuyo objetivo es predecir el valor de los *pseudo-residuos* calculados.

Teniendo un árbol construido para la iteración m, se debe actualizar la función $F_m(x)$. Esto se logra minimizando el valor de la *loss function* para cada hoja del árbol entrenado cuando se agregan valores para cada clase a $F_m(x)$:

$$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{l,m}} L_{log}(y_i, F_{m-1}(x) + \gamma), \tag{10}$$

donde $R_{l,m}$ corresponde a la hoja l del árbol m. Finalmente con los valores de γ obtenidos se procede a actualizar las predicciones que se dará a los valores de entrada:

$$F_m(x) = F_{m-1} + \eta \gamma_{l,m} I(x \in R_{l,m}), \tag{11}$$

donde η se denomina *learning rate* y corresponde a un valor definido al entrenar el modelo para definir qué tan abruptos son los cambios realizados al modelo en cada iteración. Un *learning rate* alto puede provocar que ocurra *overfitting* mientras que un *learning rate* bajo puede provocar que el algoritmo deba construir muchos árboles para obtener una buena predicción. Finalmente se procede a una nueva iteración hasta que se cumple un número M de iteraciones para obtener el valor final de la predicción del modelo $F_M(x)$.

2.4.1. XGBoost

Extreme Gradient Boosting o *XGBoost* corresponde a una implementación de *Gradient Boosting* diseñada para optimizar el tiempo de computación requerido y a su vez mejorar su desempeño al realizar predicciones y al clasificar [12]. Los modelos *XGBoost* han mostrado ser competentes en distintos ámbitos y han sido utilizado por ganadores de distintas competiciones de aprendizaje de máquinas *Kaggle* [13].

El mejor desempeño de *XGBoost* en comparación a modelos tradicionales de *Gradient Boosting* se debe a diversas optimizaciones implementadas en su algoritmo [5], entre las que se encuentran:

- *Weighted Quantile Sketch*: Corresponde a un método para definir las separaciones de los nodos de los árboles, donde se dividen los datos en distintos cuantiles considerando la *loss function* de cada instancia. Con esto, al evaluar las posibles separaciones se priorizan aquellas que consideren valores límites entre los cuantiles, resultando en que el modelo da un mayor énfasis a las áreas en las que existe una mayor dificultad para predecir.
- *Complexity Penalty Term*: Se agrega a la *loss function* una función reguladora que considera factores como la cantidad de hojas del árbol construido. Esto se utiliza para favorecer la construcción de árboles más simples teniendo así una menor probabilidad de presentar *overfitting*.
- *Sparsity-Aware Split Finding*: El algoritmo *XGBoost* posee una metodología para tratar con datos faltantes (*NaN*) o datos nulos (frecuentes valores seguidos iguales a 0). Esto se realiza al probar los datos de ambas direcciones al dato faltante (mayor o menor) para evaluar cuál da un mejor resultado al definir una separación del árbol. Esta dirección se guarda para ser utilizada también en árboles posteriores evitando así la necesidad de computar los mismos datos faltantes en múltiples iteraciones.
- Paralelización: Cuando se construyen los árboles se evalúan distintas separaciones y ramificaciones de manera paralela para posteriormente compararlas. Esta paralelización puede

disminuir considerablemente el tiempo de entrenamiento del modelo, dependiendo de la arquitectura de la unidad procesadora que se esté utilizando,

- Optimización del Hardware: Se utiliza memoria caché para optimizar el acceso a los datos de entrenamiento. Esto tiene mayor relevancia en casos en que se tiene un conjunto de datos de gran tamaño. Además, se realizan optimizaciones al manejo de datos provenientes de múltiples discos.

Con estas metodologías implementadas en el modelo tradicional de *Gradient Boosting*, el algoritmo *XGBoost* logra realizar predicciones acertadas y con un bajo costo computacional. Por este motivo, se busca evaluar el desempeño de dicho algoritmo con la base de datos utilizada por ALeRCE con el objetivo de maximizar su potencial de predicción. Sin embargo, como muchos otros algoritmos de aprendizaje de máquinas, los modelos *XGBoost* pueden resultar poco efectivo si existe un desbalance suficientemente grande entre las clases de los datos de entrenamiento. Debido a esto, ante tal situación es necesario implementar diferentes técnicas para enfrentar el desbalance de clases.

2.5. Métricas de desempeño

Dado que para el desarrollo del trabajo se requiere evaluar el desempeño de distintos modelos de aprendizaje de máquinas, resulta de mucha importancia definir bajo qué métricas se evalúan las predicciones realizadas por un modelo [14]. Entre las métricas más utilizadas para un modelo de clasificación se encuentran:

- *Precision*: Para una clase j, la métrica *precision* responde a la pregunta "¿Qué proporción de las instancias clasificadas como clase j fueron clasificadas correctamente?", y se escribe según la ecuación siguiente:

$$Precision_j = \frac{TP_j}{TP_j + FP_j}, \tag{12}$$

  donde TP_j corresponde a los verdaderos positivos de la clase j (Instancias pertenecientes a la clase j que fueron clasificadas como clase j) y FP_j a los falsos positivos de la clase j (Instancias pertenecientes a otra clase que fueron clasificadas como clase j).

- *Recall*: Para una clase j, esta métrica responde a la pregunta "¿Qué proporción de las instancias de la clase j fueron clasificadas correctamente?", y se escribe según la ecuación:

$$Recall_j = \frac{TP_j}{TP_j + FN_j}, \tag{13}$$

  donde FN_j corresponde a los falsos negativos de la clase j (Instancias pertenecientes a la clase j que fueron clasificadas como otra clase).

- *F1-Score*: Se define como el *F1-Score* a la media armónica entre las métricas de *precision* y *recall*. Este se define como sigue:

$$F1\text{-}Score_j = 2\frac{Precision_j \cdot Recall_j}{Precision_j + Recall_j} \tag{14}$$

Las métricas definidas en las ecuaciones (12), (13) y (14) consideran a una clase en particular. Para evaluar el desempeño del modelo en general se pueden considerar las métricas *micro*, que se calculan utilizando el número de verdaderos positivos, falsos positivos y falsos negativos de todos los datos sin considerar la clase de cada dato, o las métricas *macro*, correspondientes al promedio de las métricas obtenidas individualmente para cada clase. Debido a que se desea analizar el efecto del desbalance de clases en la clasificación y el desempeño de los modelos al clasificar las clases minoritarias, se utilizan las métricas *macro*, ya que al promediar el valor de las métricas para cada clase, permiten visualizar de mejor manera el desempeño de todas las clases por igual.

Dado que las diferentes métricas de desempeño presentes en la literatura califican la predicción realizada por un clasificador bajo distintos estándares, para evaluar el desempeño de un modelo es relevante analizar los resultados de diferentes métricas, y tener en consideración cuál es el objetivo prioritario del problema de clasificación que se está realizando. Lo anterior puesto que incluso métricas utilizadas comúnmente como una medición generalizada de desempeño, como el área bajo la curva ROC en problemas de clasificación binaria, pueden llevar a interpretaciones no representativas [15].

Cabe destacar que en la literatura existen dos diferentes maneras de calcular el promedio *macro* de la métrica *F1-Score* [16], las que corresponden al promedio aritmético de los *F1-Score* obtenidos para cada clase, y a la media aritmética de los promedios *macro* de las métricas *precision* y *recall*.

La primera versión se obtiene mediante el cálculo de *F1-Score* para cada clase, es decir, la media aritmética entre *precision* y *recall* obtenidos para cada clase específica. Finalmente, estos valores de *F1-Score* se promedian mediante la media aritmética:

$$F1\text{-}Score = \frac{1}{n}\sum_{j}^{n} F1_j = \frac{1}{n}\sum_{j}^{n} 2\frac{Precision_j \cdot Recall_j}{Precision_j + Recall_j}, \tag{15}$$

donde n corresponde al número total de clases.

Por otro lado, la segunda forma de calcular el promedio *macro* de esta métrica corresponde a calcular primeramente los promedios *macros* de las métricas de *precision* y *recall*, para luego obtener la media aritmética de los valores obtenidos:

$$F1\text{-}Score = 2\frac{Precision_{macro} \cdot Recall_{macro}}{Precision_{macro} + Recall_{macro}} = 2\frac{(\frac{1}{n}\sum_{j}^{n} P_j) \cdot (\frac{1}{n}\sum_{j}^{n} R_j)}{(\frac{1}{n}\sum_{j}^{n} P_j) + (\frac{1}{n}\sum_{j}^{n} R_j)}, \tag{16}$$

Estas dos formas de calcular el *F1-Score* pueden conllevar resultados diferentes, por lo que si se desea estudiar dicha métrica es necesario definir que definición se utilizará. La definición descrita en (16) presenta un sesgo cuando se tiene un conjunto de datos desbalanceado en un problema de clasificación de múltiples clases, otorgando un valor alto cuando el clasificador tiene un porcentaje de aciertos alto en las clases mayoritarias y bajo en las clases minoritarias [16], por lo que analizar el *F1-Score* utilizando dicha definición sería engañoso considerando los objetivos del presente trabajo. Por este motivo, para el cálculo de la métrica *F1-Score* la definición descrita en (15).

Por otro lado, dado que se está trabajando con un conjunto de datos desbalanceado, es de interés saber si a pesar de que se tenga un promedio alto de las métricas para un determinado modelo,

existe alguna clase en particular que esté siendo predicha de forma ineficiente por el modelo. Para esto se estudiará también en cada modelo el valor del *recall* mínimo entre todas las clases, para determinar si existe alguna clase específica que tenga un rendimiento particularmente bajo.

Por el mismo motivo, se propone una modificación de la métrica *recall* que de mayor peso al desempeño del modelo al predecir las clases minoritarias. Esto se realiza al obtener el producto punto entre el vector de los valores de *recall* para cada clase con un vector de pesos normalizado. Para establecer un vector de pesos estándar que priorice las clases minoritarias, se define el peso de cada clase inversamente proporcional al número de instancias respectivas en el conjunto de datos, es decir, para una clase con un número i de instancias en el conjunto de datos, su peso corresponde al valor $1/i$. Como se mencionó anteriormente, este vector se normaliza con el fin de que el resultado de la multiplicación con el vector de los *recall* de cada clase esté entre 0 y 1.

Con esto, la métrica adicional propuesta para el análisis de desempeño de los clasificadores estudiados se define a continuación:

$$Recall_{mod} = \frac{1}{n} \sum_{j}^{n} Recall_j \cdot w_j \tag{17}$$

$$w_j = \frac{1}{N_j} \cdot \frac{1}{\sum_j^n \frac{1}{N_j}}, \tag{18}$$

donde N_j corresponde al número de instancias de la clase j presentes en la base de datos, y n es el número de clases. Los pesos w_j se definen de tal forma que la suma de sus valores sea igual a 1, definiendo así límites para el valor de $recall_{mod}$ entre 0 y 1, alcanzándose estos valores cuando el *recall* de cada clase es igual a 0 y cuando este valor es 1 para cada clase, respectivamente.

2.6. Estrategias de balance a nivel de datos

Como se mencionó anteriormente, la mayoría de los algoritmos de aprendizaje de máquinas convencionales presenta dificultad para predecir correctamente si fue entrenado con datos altamente desbalanceados. En el caso del conjunto de datos a utilizar, correspondiente a distintos tipos de alertas astronómicas, se tiene que los datos conocidos de supernovas (de la clase Transiente) presentan un número de instancias considerablemente menor a las otras clases, y dentro de cada clase existe también un alto desbalance entre el número de instancias de sus subclases (Ver Figura 4).

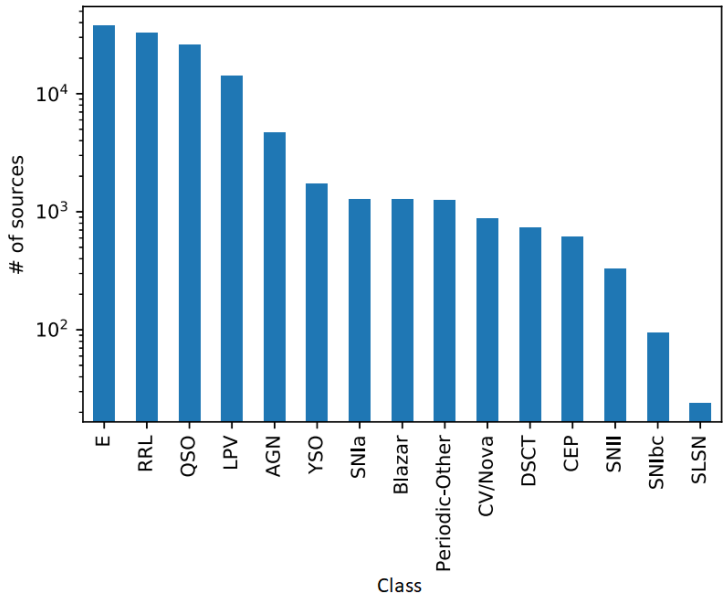


Figura 4: Número de instancias de cada clase en el conjunto de datos. La cantidad de datos va desde 37901 (Binarias Eclipsantes) hasta 24 (Supernovas Superluminosas)

Existen distintas metodologías presentes en la literatura para enfrentar la problemática del desbalance de clases, categorizándose principalmente en estrategias a nivel de datos, en las que se modifica el conjunto de datos con el que se entrena el modelo, y estrategias a nivel de algoritmo, en las que se modifica el algoritmo de entrenamiento para considerar el desbalance de clases.

Las estrategias de balance a nivel de datos corresponden a métodos en los que se busca que el modelo se entrene con un conjunto de datos con un desbalance de datos reducido en comparación al conjunto de datos original. Se categorizan en estrategias de *undersampling*, *oversampling* y estrategias mixtas [17].

- *UnderSampling*: Corresponde a remover del conjunto de datos instancias pertenecientes a las clases mayoritarias para disminuir el desbalance en el conjunto.
- *Oversampling*: Corresponde a aumentar el número de instancias minoritarias en el conjunto de datos. Esto puede lograrse al repetir datos de instancias minoritarias o crear datos sintéticos a partir de los datos existentes.
- Estrategias mixtas: Corresponden a estrategias que combinan el uso de *undersampling* y *oversampling*.

2.6.1. Undersampling

Existen diferentes metodologías para determinar qué instancias de las clases mayoritarias se eliminan para balancear el conjunto de datos. Entre ellas se encuentran:

- *Random Undersampling*: Como su nombre lo indica, al realizar *random undersampling* se seleccionan instancias aleatorias de la clase mayoritaria para no ser consideradas en el proceso de entrenamiento del modelo.

- *Bootstrapping*: Corresponde a un submuestreo del conjunto de datos en el que se construye un subconjunto utilizando instancias seleccionadas de forma aleatoria y con repetición. La variación de este método denominada *Bootstrap balanceado* corresponde a una condicionalidad adicional que se impone cuando se construye el subconjunto submuestrado, definiendo que el número de instancias de cada clase presente en el subconjunto sea el mismo. Este variación del algoritmo de submuestreo diferencia un modelo *Balanced Random Forest* de un modelo *Random Forest* tradicional [4].
- *Near-Miss*: Estrategia que utiliza un algoritmo *kNN* (*k-Nearest Neighbors*) capaz de identificar instancias que se encuentran cercanas a otras en el espacio de sus variables [18]. Con esta información, se determina qué datos serán eliminados del conjunto de datos. Existen también otras metodologías que utilizan *kNN* para determinar datos a ser eliminados del conjunto a utilizar, como *Condensed nearest-neighbors* [19] o *All-KNN* [20].

Las estrategias de *undersampling* cuentan con la ventaja de que se obtienen conjuntos de datos de menor tamaño lo que facilita el tiempo de computación para entrenar los modelos, además de permitir un mejor desempeño al balancear las clases en dicho conjunto. Sin embargo, al eliminar datos se tiene la desventaja de que existe pérdida de información, por lo que en conjuntos de datos pequeños o en los que el número de instancias minoritarias sea significativamente menor al de las mayoritarias, puede darse que el modelo no cuente con la información suficiente para aprender a clasificar los datos de forma eficaz.

Esta problemática se da al aplicar técnicas de *undersampling* en el conjunto de datos utilizados para el presente trabajo, debido a que existen clases como la de SuperNova Super-Luminosa (*SLSN*) que cuenta con solo 24 instancias. Por esto, si se desea reducir en gran medida el desbalance de datos existe pérdida de información, por lo que al utilizar submuestreo es necesario estudiar múltiples iteraciones de la estrategia con el fin de tener un análisis representativo del conjunto de datos.

2.6.2. Oversampling

Contrarias a las estrategias de *undersampling*, las estrategias de *oversampling* consisten en aumentar el número de instancias minoritarias en la base de datos, ya sea por medio de repetición o por generación de datos nuevos, denominados *datos sintéticos*.

- *Random Oversampling*: Análogo al *random undersampling*, se seleccionan instancias aleatorias de la clase minoritaria para repetirse en el conjunto de datos y aumentar así artificialmente el número de instancias de las clases minoritarias.
- SMOTE: *Synthetic Minority Oversampling Technique* [21] corresponde a la creación de datos sintéticos de clases minoritarias a partir de la información que se tiene de los datos existentes. Esto se realiza mediante la interpolación de las variables de los datos minoritarios, permitiendo así la creación de datos "intermedios" entre una instancia y sus vecinos. A partir de SMOTE se han generado distintas variaciones que buscan generar datos sintéticos siguiendo diferentes metodologías [22], entre las que se encuentran *Borderline-SMOTE* [23], SVM-SMOTE [24], ADASYN [25], entre otras. Cabe considerar que debido a que al utilizar SMOTE se están creando datos sintéticos a partir de variables de datos existentes, la representatividad de los datos creados depende de las características de los datos reales, puesto que para datos complejos es posible que una interpolación entre dos instancias de una clase no sea una muestra representativa de una instancia nueva de dicha clase.

- *Variational Auto-Encoders*: Un *Auto-Encoder* corresponde a un algoritmo que utiliza estructuras de redes neuronales para transformar datos de entrada a un espacio con dimensión diferente (generalmente menor) denominado *espacio latente*, y también para decodificar datos transformados desde este espacio. Se basa en un modelo probabilístico que busca maximizar la similitud de un conjunto de datos con una distribución de probabilidad parametrizada por el modelo [26]. Obteniendo dicha distribución, es posible generar datos sintéticos ajustándolos a los parámetros determinados.

2.6.2.1. Generación de curvas sintéticas basada en modelo paramétrico

Para el caso específico de los objetos de clase transiente o supernova, existe un modelo denominado *Supernova Parametric Model* (SPM) que describe el comportamiento de su curva de luz a partir de un conjunto de parámetros [7]. La siguiente función describe el modelo que representa una curva de luz en función de dichos parámetro:

$$f_{sne}(t) = f_{early}(t) \cdot (1 - g(t)) + f_{late}(t) \cdot g(t) \tag{19}$$

$$g(t) = \sigma(s \cdot (t - (\gamma + t_0))) \tag{20}$$

$$f_{early}(t) = \frac{A \cdot (1 - \beta'(\frac{t-t_0}{\gamma})}{1 + exp(\frac{-(t-t_0)}{\tau_{rise}})} \tag{21}$$

$$f_{late}(t) = \frac{A \cdot (1 - \beta') \cdot exp(\frac{-(t-(\gamma+t_0))}{\tau_{fall}})}{1 + exp(\frac{-(t-t_0)}{\tau_{rise}})}, \tag{22}$$

donde A, t_0, γ, β', τ_{rise} y τ_{fall} corresponden a los parámetros SPM mencionados anteriormente, σ es la función sigmoide y s corresponde a un parámetro para controlar la transición entre $f_{early}(t)$ y $f_{late}(t)$, definido en $s = 0{,}2$. La curva de luz de una supernova tiene la forma general de un aumento en magnitud constante o exponencial que comienza decaer luego de llegar a un pico de luminosidad, forma que se tiene debido a la naturaleza del evento en cuestión. Con esto, los distintos parámetros SPM determinan características de dicha forma en la curva de luz, como el momento en el tiempo en que se llega al pico o el valor de la magnitud de éste.

Para generar una curva sintética utilizando esta estrategia, se realiza primeramente una estimación de distribución de parámetros SPM óptimos a partir de una curva de luz determinada. Posteriormente se utiliza una ventana de muestreo temporal para evaluar la función f_{sne} con un conjunto de parámetros SPM obtenidos anteriormente en dicha ventana. De esta forma, al utilizar diferentes parámetros y ventanas de muestreo se obtienen diferentes observaciones sintéticas (Ver Figura 5).

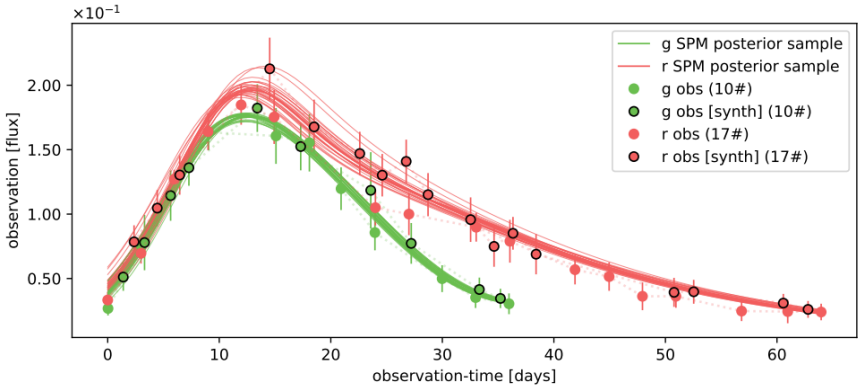


Figura 5: Ejemplo de generación de una curva de luz sintética. Se grafican curvas en ambas bandas generadas con una distribución de parámetros SPM y de estas se toman muestras aleatorias para construir una curva sintética (puntos con borde negro). *(Fuente:* [7]*)*

Posteriormente se utilizan diferentes distribuciones gaussianas para definir un error sintético a partir de una observación empírica determinada. De esta forma se da a las curvas generadas sintéticamente un grado de error factible para obtener una mejor representatividad. Con esto, es posible realizar un balance en el número de instancias para cada clase al utilizar un mayor número de curvas sintéticas correspondientes a las clases minoritarias, específicamente a curvas de eventos tipo SNIc y SLSN.

En [7] se observó que el uso de curvas sintéticas generadas con esta metodología en conjunto con un modelo basado en atención presentó mejoras significativas en la predicción al compararse con un modelo *Balanced Random Forest* como el utilizado en ALeRCE, por lo que es de interés analizar el impacto que tiene el uso de esta metodología de aumento de datos cuando es utilizada en conjunto con un modelo *XGBoost*.

De las diferentes metodologías descritas en [7] para la generación de las curvas sintéticas, se utilizará la propuesta en dicho trabajo, correspondiente a "*mcmc-estw*". Esta metodología comprende el uso de un método de *Markov Chain Monte Carlo (MCMC)* para la estimación de la distribución de parámetros óptimos, en conjunto con una ventana de tiempo de muestreo extendida, que tiene la capacidad de muestrear observaciones sintéticas previas a los datos iniciales de la curva base.

2.6.3. Estrategias Mixtas

Con el fin de balancear las ventajas y desventajas de las técnicas de *oversampling* y *undersampling* se han desarrollado estrategias mixtas, que realizan ambas labores de manera híbrida.

- SPIDER: Algoritmo enfocado en problemas de clasificación de múltiples clases, que incorpora técnicas de *undersampling* y *oversampling* a nivel de vecindarios de instancias, eliminando instancias de las clases mayoritarias que se consideren conflictivas y aplicando un cierto grado de *random oversampling*. Incorpora también una matriz de costos que define el peso que tiene clasificar de forma incorrecta una clase determinada, matriz que se define por parte del usuario dependiendo de las necesidades que se tengan para la clasificación [27].

- SOUP: La estrategia de *Similarity Oversampling and Undersampling Preprocessing* considera factores denominados *difficulty factors*, que definen la seguridad que ofrece una clase para la clasificación dentro de un determinando vecindario. Para obtener los *difficulty factors* se calculan factores como la sobreposición de clases dentro de un mismo vecindario y el desbalance de éstas. Con esto, el algoritmo realiza primeramente un proceso de *undersampling* para todas las clases, eliminando las instancias menos seguras, y finalmente realiza *oversampling* duplicando a nivel de vecindario las instancias que se determinaron más seguras [28].

  Al aplicar una metodología híbrida es posible controlar que el proceso de *oversampling* y *undersampling* se ejecuten de forma complementaria para asegurar una máxima eficiencia, sin embargo, se deben ajustar los parámetros del algoritmo y condiciones del conjunto de datos para evitar así los problemas que presentan ambos métodos.

2.7. Estrategias de balance a nivel de algoritmo

2.7.1. Cost Sensitive Learning

La estrategia de *Cost-Sensitive Learning* [29] consiste en asignar a cada clase un costo determinado que se le da al algoritmo por clasificar erróneamente dicha clase. Con esto, al entrenarse un algoritmo de aprendizaje de máquinas se priorizará predecir las clases que tengan un mayor costo.

Generalmente el costo de cada clase es inversamente proporcional al número de instancias de dicha clase presentes en el conjunto de datos, con el fin de que las clases con menor presencia sean mayormente priorizadas por el algoritmo. Por otro lado, también se pueden definir manualmente los costos para especificar qué clases deben ser priorizadas por sobre otras.

Existe también la posibilidad de definir una matriz de costos donde cada valor $c_{i,j}$ corresponde al costo asociado a clasificar erróneamente una instancia de la clase i a la clase j. Esto resulta de utilidad si se observa que el algoritmo presenta confusión con clases en específico del conjunto de datos.

Una desventaja que presenta esta estrategia es que si un modelo cuenta con poca información de la clase minoritaria, no podrá realizar una predicción adecuada incluso si se prioriza mediante costos, por lo que la mejora observada por utilizar *cost-sensitive learning* depende de las características del conjunto de datos.

2.7.2. Focal Loss

También se puede abordar el problema a nivel de algoritmo al definir una *loss function* diferente. Una variación de la función *log loss* corresponde a la función *Focal Loss* [30], diseñada para modelos de clasificación binaria en la que se busca disminuir la pérdida asociada a la *loss function* en las instancias fáciles de clasificar.

La función de *Focal Loss* se define como sigue:

$$FL = -\frac{1}{N}\sum_{i}^{N}(1-p_{ti})^{\gamma} log(p_{ti}), \tag{23}$$

donde p_{ti} corresponde a la probabilidad de que la instancia i sea clasificada correctamente. Con esto, con un $\gamma > 0$, la función de pérdida descrita en (23) aumentará mientras mayor sea la probabilidad p_{ti} se tiene una menor función de pérdida, provocando así que el modelo priorice las instancias difíciles de clasificar.

2.8. Herramientas de análisis estadístico

Los resultados obtenidos en problemas de aprendizaje de máquinas están generalmente dados por valores promedio con sus respectivas desviaciones estándar, dado que las métricas y valores asociados a la medición del desempeño de un modelo se obtienen mediante estrategias de validación cruzada que consideran la evaluación del modelo al ser sujeto a distintos conjuntos de evaluación. Debido a que en este proceso se obtienen múltiples resultados para alguna métrica de comparación definida, es relevante contar con herramientas de análisis estadístico para determinar si se tendrá cierto porcentaje de certeza sobre la validez de las interpretaciones que se realicen de dichos resultados.

Una de estas herramientas corresponde al test de Shapiro-Wilk [31], en el que se plantea la hipótesis nula de que una determinada muestra de valores está asociada a una distribución aleatoria normal. Luego, al estimar una probabilidad asociada a dicha hipótesis se puede contar un grado de certeza sobre si la distribución de los valores testeados se asemeja a una normal.

Para estimar la probabilidad de que se cumpla la hipótesis nula para un conjunto de valores y se calcula la estadística de prueba W, según se describe a continuación:

$$W = \frac{\sum_i (a_i \cdot y_i)^2}{\sum_i (y_i - \overline{y})^2}, \tag{24}$$

donde a_i corresponde a un factor asociado a los valores esperados de una distribución normal y su matriz de covarianza, e $\overline{y}$ corresponde al valor promedio del conjunto de valores y_i. Posteriormente se procede a estimar el valor de *p-value*, o probabilidad de que se cumpla la hipótesis nula, a partir del valor de la estadística de prueba. Este cálculo se hace mediante distintas aproximaciones dependiendo del tamaño del conjunto al que se le realiza el test. Finalmente, se compara el *p-value* obtenido con un umbral definido para determinar si se acepta o rechaza la hipótesis nula. Este umbral puede variar según las necesidades del trabajo que se esté realizando, sin embargo, un valor comúnmente utilizado corresponde a 0,05.

Otra herramienta estadística que se debe tener en consideración si se desea comparar resultados es el *t-test* o test de *t-student* [32], el cual corresponde a un test estadístico que plantea la hipótesis nula de que dos conjuntos de valores se asocian a distribuciones con igual varianza y media.

El valor de la estadística de prueba para el test *t-student* utilizado para evaluar dos muestras y_1 e y_2 se detalla a continuación:

$$t = \frac{\overline{y_1} - \overline{y_2}}{\sqrt{s_{(1+2)} \frac{2}{n}}} \tag{25}$$

$$s_{(1+2)} = \sqrt{\frac{s_1 + s_2}{2}} \tag{26}$$

$$s_i = \frac{\sum_j (y_j - \overline{y})^2}{n-1}, \tag{27}$$

donde n corresponde al tamaño del conjunto de valores, por lo que las ecuaciones descritas consideran dos conjuntos de igual número de valores. Cabe destacar también que para comparar dos conjuntos mediante el test *t-student* se asume que ambos tienen varianza semejante. Por esto, resulta de utilidad utilizar el anteriormente mencionado test de *Shapiro-Wilk* con el fin de determinar si la distribución de y_1 e y_2 tiene semejanza con una distribución normal, ya que el aceptar la hipótesis nula de dicho test conlleva a que ambos conjuntos son comparables mediante el test *t-student*.

3. Metodología

Para desarrollar el trabajo se utilizan los datos de ALeRCE descritos en [2], correspondientes a alertas astronómicas de las cuales se han extraído características y se han determinado sus respectivas clases. Este conjunto de datos cuenta con 123,496 objetos para los cuales se tiene un total de 152 características para cada uno.

El análisis de los modelos y técnicas de balance de datos se elaboran utilizando el lenguaje de programación *Python* debido a la amplia variedad de bibliotecas relacionadas al aprendizaje de máquinas disponibles en dicho lenguaje, y se utilizan las librerías *imbalanced-learn* y *xgboost* para implementar los modelos de *Balanced Random Forest* y *XGBoost*, respectivamente.

Siguiendo la metodología descrita en [2], se consideran 4 clasificadores correspondientes a un clasificador jerárquico que predice la clase de cada dato específico (Periódica, Estocástica o Transiente) y un clasificador para cada clase que determina la sub-clase a la que corresponde cada dato, los que se entrenan de manera independiente entre si.

Dado que se tiene como objetivo evaluar el desempeño de los modelos entrenados en el contexto del balance de clases, es relevante considerar la cantidad de instancias específicas de cada clase presentes en el conjunto de datos utilizado:

- Objetos periódicos: 37.901 *eclipsing binary* (E), 32.482 *RR Lyrae* (RRL), 14.076 *Long-Period Variable* (LPV), 1.256 *Periodic-Other*, 732 δ *Scuti* (DSCT), y 618 *Cepheid* (CEP), sumando un total de 87065.
- Objetos estocásticos: 26.168 Quasar tipo 1 (QSO), 4.667 *Active galactic nuclei* (AGN), 1.740 *young stellar object* (YSO), 1.267 *Blazar*, y 871 *cataclysmic variable/nova* (CV/Nova), sumando un total de 34713.
- Objetos transientes: 1.271 Supernovas tipo Ia (SNIa), 328 Supernovas tipo II (SNII), 94 Supernovas tipo Ibc (SNIbc) y 24 Supernovas superluminosas (SLSN), sumando un total de 1718.

Para evaluar el desempeño de cada modelo es relevante considerar la naturaleza estocástica de los algoritmos de entrenamiento, y que los datos seleccionados para entrenar y para validar influyen en el desempeño observado de un modelo. Ante esto, al buscar la combinación óptima de hiperparámetros para un modelo se utiliza *Nested Cross Validation* o validación cruzada anidada [33]. En esta, se realiza una validación por dos etapas: Primeramente se separa una parte del conjunto de datos para ser utilizados como datos de prueba, una parte para validación y el resto se ocupa como datos de entrenamiento. Posteriormente para cada conjunto de entrenamiento se utilizan distintas combinaciones de hiperparámetros y se evalúa su desempeño según los datos de validación para determinar la combinación de parámetros óptima según una de las métricas definidas en el capítulo 2.5, para posteriormente evaluar dicha combinación de parámetros con los datos que se definieron originalmente como datos de prueba (Ver Figura 6).

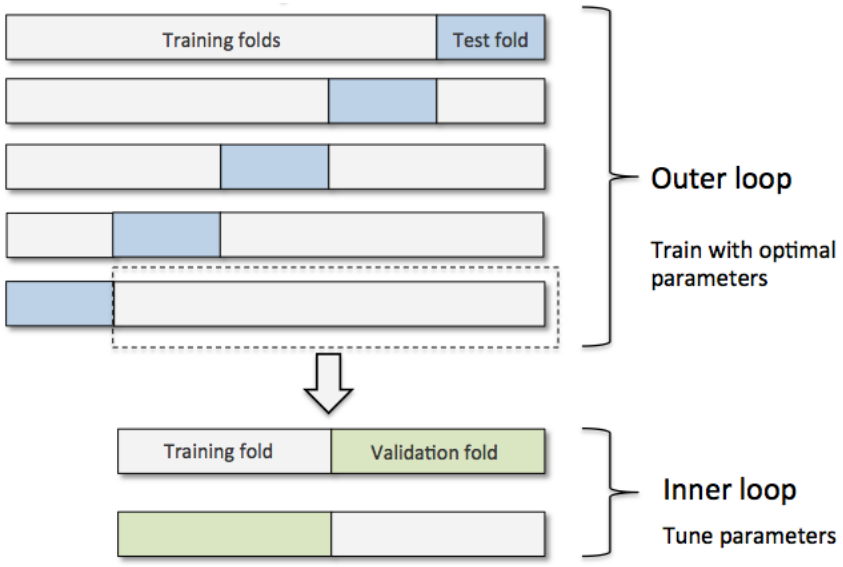


Figura 6: Diagrama representativo del proceso de validación cruzada anidada. *(Fuente: Kaggle)*

La separación de los conjuntos de entrenamiento y prueba se realiza tomando un 33 % del conjunto total para usarse para prueba. Esto con el fin de tener un conjunto de prueba suficientemente representativo sin eliminar gran cantidad de información del conjunto de entrenamiento, especialmente en las clases con menor representación en el conjunto. Particularmente, se tiene que de esta forma al evaluar el desempeño de un clasificador para predecir curvas transientes, se tiene solamente 8 instancias de la clase SLSN en el conjunto de prueba, lo que conlleva que el desempeño de la predicción sea propenso a variaciones notorias dependiendo de las instancias específicas que se consideren. Sin embargo, utilizar un conjunto de prueba mas grande significaría reducir el tamaño del conjunto que se utilizará para entrenar los modelos, lo que puede conllevar que el modelo no se entrene de forma deseada. Para el caso específico mencionado, un modelo que predice curvas transientes dispone de solo 16 ejemplos para "aprender" a distinguir una curva de clase SLSN, cantidad que en un contexto de aprendizaje de máquinas está muy por debajo del volumen que se espera tener para entrenar un modelo de predicción.

Debido a la naturaleza desbalanceada de los datos del conjunto, es especialmente importante considerar que la separación entre los conjuntos de prueba y entrenamiento, así como también del conjunto de validación que será utilizado para la prueba de hiperparámetros, sea realizada mediante *Stratified K-Fold*, separación que divide el conjunto de datos de forma tal que la proporción de datos por clase en ambos subconjuntos resultantes sea similar. De esta forma se asegura que el conjunto de prueba sea representativo de los datos que fueron utilizados para entrenar al modelo.

Al utilizar la mencionada estrategia de validación cruzada anidada, es posible evaluar el desempeño del modelo cuando se utilizan los datos conocidos para estudiar las mejores combinaciones de hiperparámetros. Esto resulta relevante particularmente al utilizar un modelo basado en *XGBoost* debido a que distintas combinaciones de sus hiperparámetros conllevan a que el modelo pueda presentar patrones de predicción notablemente diversos, principalmente con respecto al sesgo que

presenta la clasificación hacia determinadas clases.

Debido a esto es que toma mayor importancia la métrica que se utilice para determinar la mejor combinación de hiperparámetros, puesto que distintas configuraciones del modelo pueden maximizar distintas métricas de clasificación, y como se mencionó anteriormente, algunas métricas tales como *Accuracy* no son representativas de la calidad de la clasificación cuando se tiene un conjunto de datos altamente desbalanceado. Por esto, se busca evaluar diferentes métricas y seleccionar una que proporcione un valor acorde al comportamiento que se espera para el modelo.

Es intuitivo utilizar en este caso la métrica *macro recall*, también denominada *Balanced Accuracy*, que al ser una métrica *macro*, promedia con igual peso el desempeño del modelo para cada clase, haciendo que un bajo porcentaje de predicciones correctas en clases minoritarias se vea representado al evaluar dicha métrica. Sin embargo, dado que existen diversos casos en que se tiene mayor interés por predecir correctamente las clases minoritarias, se evaluará también los resultados utilizando la métrica $Recall_{mod}$ propuesta en 2.5. Por este mismo motivo se estudiará el valor mínimo del *recall* entre todas las clases de un problema determinado, con el fin de visualizar el caso más bajo entre los desempeños de predicción de clases individuales y tener así una noción del volumen del sesgo en la predicción. Este valor de *recall* mínimo será presentado en los resultados bajo el nombre *min-recall*.

Con esto, para los modelos entrenados en el presente trabajo, como caso general se utilizó la métrica *macro-recall* para el proceso de validación, debido a que provee una noción del desempeño medio del modelo para todas las clases, y a que se espera que sea más representativa como caso general que métricas como el $recall_{mod}$ o el min_recall.

A pesar de que, como se mencionó, métricas como *macro-precision* o *macro-f1-score* pueden no ser representativas del desempeño de clasificación en un conjunto de datos altamente desbalanceado, se estudiarán igualmente con el fin de comparar los modelos que presenten resultados altos en los distintos tipos de métricas.

3.1. Balanced Random Forest

Para evaluar el desempeño del clasificador *Balanced Random Forest* se utiliza la configuración definida en la versión 1.1.1 del modelo de ALeRCE, que se constituye de los siguientes parámetros:

- $n\_estimators = 500$
- $max\_features =$ "sqrt" para los clasificadores jerárquico, periódico, y transiente, y 0.2 para el clasificador estocástico
- $max\_depth =$ None
- $criterion =$ "entropy"
- $min\_samples\_split = 2$
- $min\_samples\_leaf = 1$

Con esto se busca reproducir los resultados descritos en [2] para realizar una comparación significativa con los resultados obtenidos en el presente trabajo.

3.2. XGBoost con Cost-Sensitive Learning

En [5], se compara el desempeño de distintas estrategias de balance de datos del estado del arte descritas en el capítulo 2 aplicadas a un modelo *XGBoost*. Se obtuvo que para todos los clasificadores el uso de *Cost-sensitive learning* produjo los mejores resultados o resultados que mostraban mayor potencial. Por este motivo, se utiliza esta técnica como base para evaluar el desempeño del clasificador *XGBoost*.

Los pesos definidos para cada clase son inversamente proporcionales al porcentaje de instancias de dicha clase en el conjunto de datos, con el fin de que las clases menos representadas posean un peso mayor y el algoritmo priorice su correcta clasificación.

Con el fin de optimizar el desempeño del modelo utilizando esta estrategia, se realiza una búsqueda de hiperparámetros en cada conjunto de entrenamiento resultante de la validación cruzada anidada, separando una parte de dicho conjunto para validar el resultado obtenido con una combinación de parámetros determinada. Así, se busca evaluar los parámetros que afectan en mayor medida la clasificación y estudiar la mejor forma de disminuir el sesgo hacia la clase mayoritaria.

Los hiperparámetros sobre los que se realiza la búsqueda corresponden a los siguientes:

- *learning_rate*: Coeficiente que pondera a cada estimador que se agrega al modelo de forma secuencial, con el fin de ajustar la predicción de manera gradual y evitar un sobreajuste.
- *gamma*: Valor mínimo de reducción de pérdida que debe tener una hoja determinada en un estimador para realizar una partición. Regulariza el proceso de construcción de los árboles al evitar que hojas que aportan poca información en la predicción sean divididas.
- *max_depth*: Profundidad máxima que puede tener un árbol en el modelo. Se define este límite para evitar que los árboles se ajusten de forma indefinida a los datos de entrenamiento y se produzca un sobreajuste.
- *min_child_weight*: Valor mínimo del peso de instancias que debe tener una hoja para dividirse. Si en el proceso de entrenamiento una hoja contiene un número bajo de instancias del conjunto de entrenamiento, se detendrá la partición.
- *subsample*: Porcentaje de los datos que se utilizará como muestreo para cada iteración del algoritmo, con el fin de que los árboles construidos por el modelo presenten mayor variabilidad.
- *colsample_bytree*: Porcentaje de columnas (variables) que se consideran al momento de construir cada árbol.

La función de pérdida utilizada por los modelos corresponde a *cross-entropy* o *log-loss* multiclase, descrita en (7).

3.3. Gradient Boosting utilizando Bootstrap Balanceado

Al identificar que el algoritmo de *Balanced Random Forest* presenta una mejora considerable en comparación al algoritmo de *Random Forest* original con datos desbalanceados debido a su estrategia de *bootstrapping*, se consideró la posibilidad de adaptar un modelo *XGBoost* para incluir un proceso de *bootstrapping* balanceado similar al de *BRF* al momento de construir sus árboles. Debido a la complejidad del algoritmo de *XGBoost* derivada de sus modificaciones al algoritmo tradicional de *Gradient Boosting*, se optó por estudiar el efecto de implementar *bootstrapping* balanceado en el algoritmo de la clase *GradientBoostingClassifier* de la librería *sklearn*, para analizar el desempeño de un algoritmo de *Boosting* más simple que *XGBoost* al modificarse para incluir la técnica de *bootstrapping* mencionada en el proceso de construcción de sus árboles.

Para esto se modificó el algoritmo de construcción de los árboles en el modelo, específicamente el método fit_stage de la clase *GradientBoostingClassifier*, método que realiza la función de construir un árbol de regresión para cada clase y ajustarlo para predecir los errores de clasificación del árbol anterior. Con esto, el árbol se ajusta para disminuir el error secuencialmente para cada clase.

Para la construcción del árbol de cada clase, el método fit_stage modifica los datos para trabajar con una metodología *one vs rest* (*ovr*), es decir, se considera un caso binario en que la clase con la que se está trabajando corresponde al caso positivo y el resto de las clases corresponde al caso negativo. Con esto, se modifica el método para que previo al cálculo de los errores del árbol anterior y de la transformación de los datos a un caso binario, se realice el proceso de *bootstrapping*. De esta forma, el árbol de regresión que se construye para cada clase se entrena utilizando un conjunto de datos balanceado, en el que cada clase está representada con el mismo número de instancias. Cabe notar que los datos completos antes del submuestreo son guardados para así ser utilizados nuevamente en la siguiente iteración de fit_stage sin tener pérdida secuencial de información (Ver Figura 7).

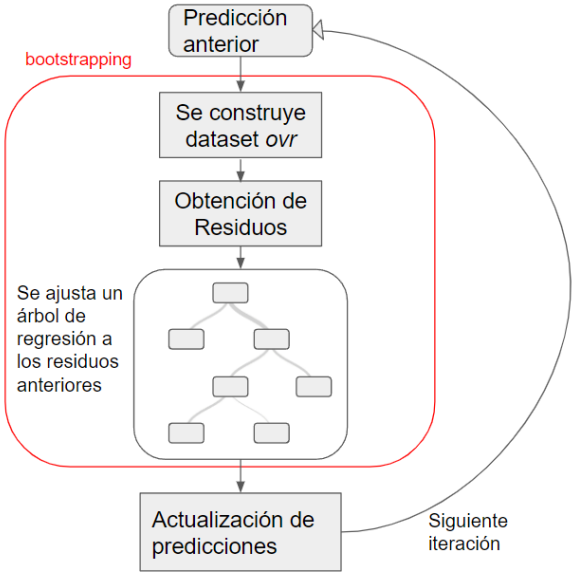


Figura 7: Diagrama que representa de manera simplificada el algoritmo utilizado en el método fit_stage. La parte dentro del contorno rojo corresponde a las acciones que se realizan con el subconjunto generado con *bootstrapping* al modificarse el algoritmo. En un problema con múltiples clases, *ovr* corresponde a tratar el conjunto de datos como un caso binario en que la clase con la que se está trabajando corresponde al caso positivo y el resto de las clases corresponde al caso negativo.

Es necesario considerar que al utilizar esta modificación, cada árbol que se construye en el modelo está siendo entrenado con un porcentaje del conjunto de datos real, por lo que para asegurar el mayor uso de información posible es importante que el modelo se construya con un número de árboles mayor relativo al al usado por el método de *Gradient Boosting* original.

3.4. Generación de curvas sintéticas usando parámetros SPM

Utilizando el método propuesto en [7] se pueden generar diferentes curvas sintéticas de objetos transientes a partir de cada curva presente en el conjunto de entrenamiento. Se utiliza la metodología descrita para generar 32 curvas sintéticas por cada observación disponible.

Con esto, se crea un nuevo conjunto de datos generados sintéticamente, los que se pueden utilizar para realizar un balance en el volumen de instancias de sus clases sin comprometer la representatividad del conjunto, como puede ocurrir al realizar *Random Undersampling* cuando la clase minoritaria está representada en volúmenes muy bajos. Cabe mencionar que dichas curvas generadas de forma sintética solo serán utilizadas para entrenar el modelo y no como parte de los conjuntos de prueba, con el fin de analizar la capacidad del modelo para predecir correctamente datos empíricos.

4. Resultados y Análisis

En este capítulo se presentan los resultados experimentales obtenidos para las distintas metodologías de clasificación utilizadas y el análisis del desempeño de los distintos modelos.

4.1. Clasificador de curvas de luz Balanced Random Forest

Primeramente se obtienen los resultados al evaluar el desempeño del clasificador *Balanced Random Forest* de ALeRCE con el fin de establecer un punto de comparación experimental en relación a las distintas metodologías estudiadas. Según la metodología descrita, se realiza una validación cruzada de 10 particiones y se obtiene tanto el valor promedio como la desviación estándar para las diferentes métricas estudiadas, así como también los valores promedio de las matrices de confusión del modelo.

Se observa en las matrices de confusión de la Figura 8 que el modelo tiene un buen desempeño en diferenciar las clases, sin presentar un sesgo alto hacia las clases mayoritarias. No se observa una correlación directa entre el número de instancias de cada clase descrito en el capítulo 3 con el desempeño de clasificación del modelo.

Al analizar las métricas descritas en la Tabla 1, se puede confirmar que los resultados obtenidos son similares a los reportados en [2]. Con esto, se obtiene que los resultados presentados para el clasificador *BRF* sirven de un punto eficaz de comparación para estudiar el desempeño de los otros modelos entrenados en el presente trabajo.

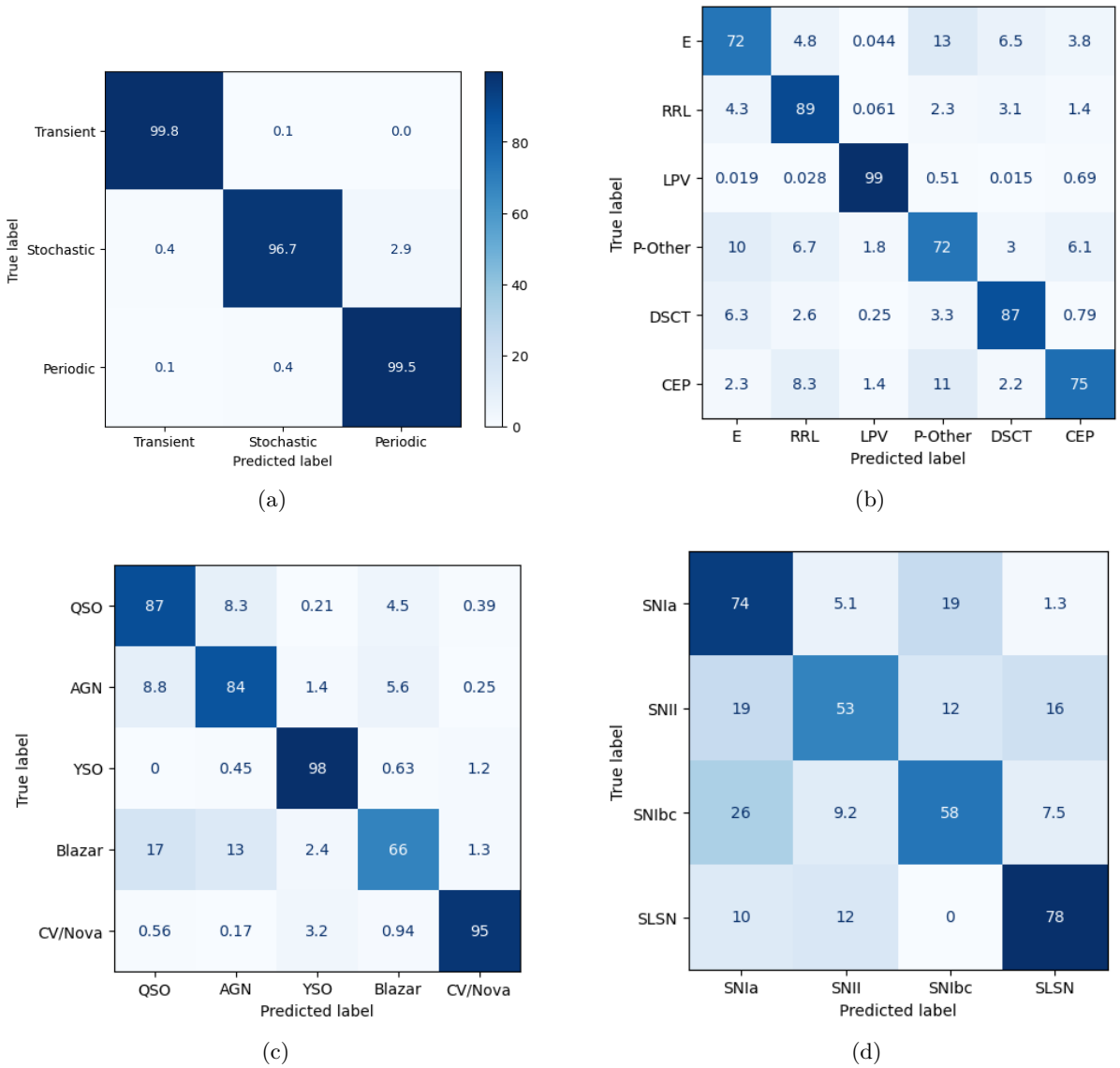
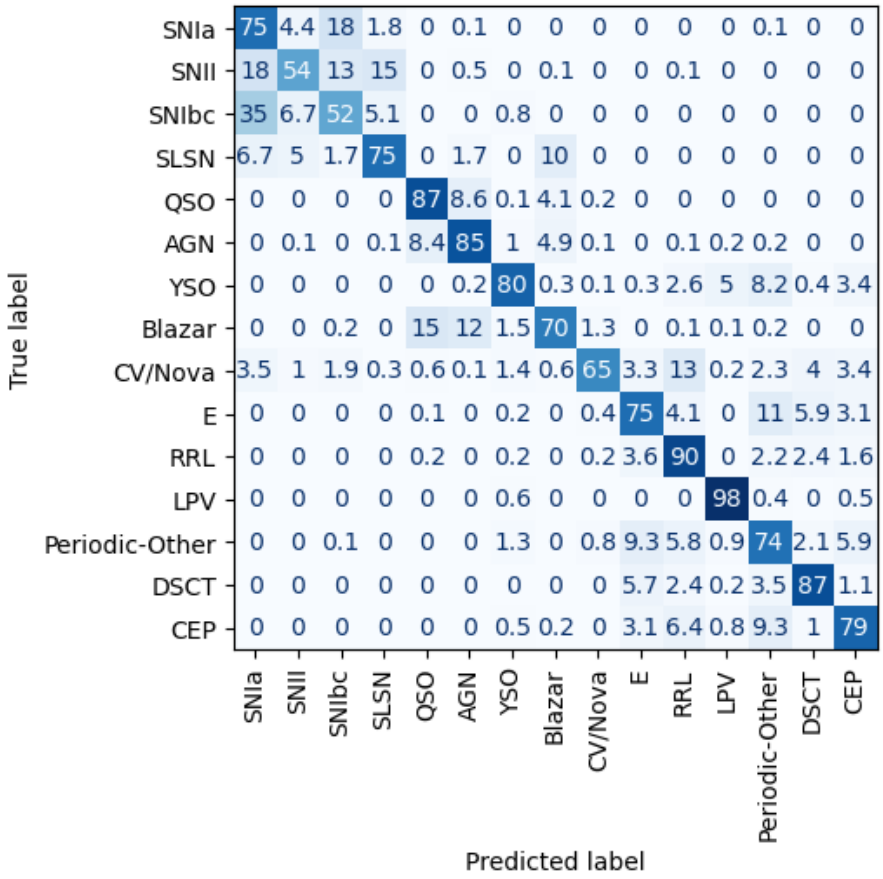


Figura 8: Matrices de confusión de clasificador Balanced Random Forest. Los valores se presentan en porcentajes. (a) Clasificador Jerárquico, (b) Clasificador Periódico, (c) Clasificador Estocástico, (d) Clasificador Transiente.

Observando las métricas de desempeño, es posible notar que se obtiene generalmente un alto valor de la métrica propuesta $recall_{mod}$, superando incluso a la métrica *recall* original en el caso del clasificador periódico y en el clasificador transiente. Esto sirve de representación para analizar que, en dichos casos, se tiene una clasificación mejor en las clases minoritarias del conjunto, mostrando la capacidad que tiene *Balanced Random Forest* para evitar el sesgo esperado hacia las clases mayoritarias y proveer un buen desempeño distribuido en las distintas clases del conjunto de datos, lo que se puede ver en la Figura 9.



(a) Clasificador Compuesto

Figura 9: Matriz de confusión de clasificador Balanced Random Forest al considerar las 15 clases. Los valores se presentan en porcentajes.

Tabla 1: Desempeño en métricas de clasificación obtenidas para los distintos clasificadores basados en Balanced Random Forest. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Clasificador	*Precision*	*Recall*	$F1 - Score$	Min_recall	$Recall\_mod$
Periódico	$0{,}56 \pm 0{,}002$	$0{,}82 \pm 0{,}006$	$0{,}58 \pm 0{,}04$	$0{,}71 \pm 0{,}008$	$0{,}86 \pm 0{,}004$
Estocástico	$0{,}74 \pm 0{,}006$	$0{,}86 \pm 0{,}004$	$0{,}79 \pm 0{,}02$	$0{,}66 \pm 0{,}02$	$0{,}86 \pm 0{,}03$
Transiente	$0{,}5 \pm 0{,}02$	$0{,}66 \pm 0{,}04$	$0{,}5 \pm 0{,}02$	$0{,}5 \pm 0{,}05$	$0{,}73 \pm 0{,}1$
Jerárquico	$0{,}96 \pm 0{,}006$	$0{,}99 \pm 0{,}00$	$0{,}97 \pm 0{,}003$	$0{,}97 \pm 0{,}002$	$0{,}99 \pm 0{,}002$
Compuesto	$0{,}58 \pm 0{,}005$	$0{,}76 \pm 0{,}01$	$0{,}61 \pm 0{,}01$	$0{,}48 \pm 0{,}05$	$0{,}70 \pm 0{,}08$

A pesar de los buenos resultados observables en las métricas de *recall* y $recall_{mod}$, se puede observar que estos conllevan un desempeño menor en las métricas de *precision* y *f1-score*, especialmente en el clasificador periódico y transiente. Esto debido a que como se mencionó anteriormente, la métrica *precision* tiende a favorecer a la correcta clasificación de la clase mayoritaria, aspecto que no se prioriza al usar un modelo de *Balanced Random Forest*.

4.2. Clasificación usando XGBoost con Cost-Sensitive Learning

Al evaluar el desempeño de un clasificador basado en *XGBoost* usando *cost-sensitive learning* se realiza una búsqueda de hiperparámetros en el conjunto de entrenamiento determinado y se evalúa su desempeño según un conjunto definido para validación. Al realizar este experimento es posible identificar que el parámetro que genera el mayor cambio en el comportamiento del modelo para realizar predicciones corresponde a *max_depth* o profundidad máxima, parámetro que define el número de ramificaciones que puede tener como máximo cada árbol construido por el algoritmo.

Al construir árboles de decisión se define un valor máximo de profundidad para evitar que el modelo se sobreajuste a los datos del conjunto de entrenamiento, debido a que un árbol que se ramifique de manera ilimitada llegaría a predecir de forma perfecta cada dato de dicho conjunto, lo que conlleva una alta probabilidad de que el desempeño de clasificación disminuya al predecir datos nuevos. Existen otros parámetros utilizados para evitar el sobreajuste en *XGBoost*, como *min_child_weight* que define un umbral mínimo de valores que deben existir en cada hoja de los árboles del modelo, sin embargo, como se mencionó anteriormente, la modificación de la profundidad máxima demuestra tener una influencia mayormente directa en las métricas de desempeño del modelo.

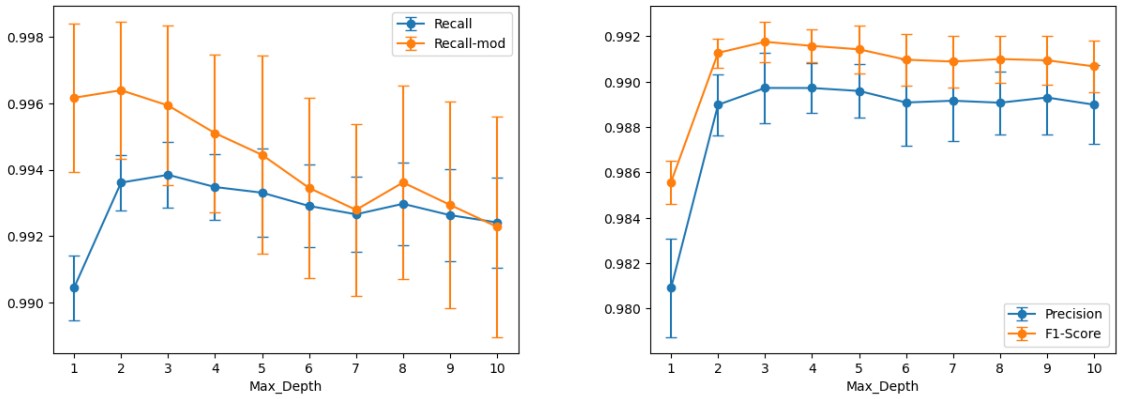
Por esto, se determinan parámetros óptimos mediante la metodología descrita para distintos valores de profundidad máxima de los árboles, con el fin de evaluar las variaciones de las métricas a estudiar en función de dicho parámetro.

Cabe mencionar que todos los modelos *XGBoost* entrenados se construyen con un máximo de 500 árboles, pero en el proceso de construcción se evalúa la función de pérdida con cada árbol que se agrega al modelo, y si en algún punto del entrenamiento se determina que la función de pérdida no ha disminuido en los últimos 15 árboles agregados, se define el mejor modelo con el árbol que presentó el mejor resultado.

Este número de árboles se seleccionó al realizar pruebas experimentales con modelos *XGBoost* en las que se observó que para el conjunto de datos utilizado, 500 árboles aseguraba llegar a un punto en que la disminución de la función de pérdida al agregar un nuevo árbol al modelo sea mínima.

4.2.1. Clasificador Jerárquico

En la Figura 10 se muestran los valores promedios y desviaciones estándares de distintas métricas seleccionadas al realizar la validación cruzada anidada de 10 particiones, para distintos valores de profundidad máxima definida para los estimadores del modelo.



(a) (b)

Figura 10: Métricas de clasificación del clasificador jerárquico *XGBoost* en función del parámetro de profundidad máxima. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

En la Figura 10 se observa que para una profundidad máxima de 2 o más las métricas del modelo presentan poca variación en sus valores promedios, con la excepción del $recall_{mod}$ que presenta una pendiente negativa en función de la profundidad máxima. En esta métrica se observan mayores valores de desviación estándar, lo que puede interpretarse como que dicha métrica depende en mayor medida de la clasificación de las clases minoritarias, clases que presentan menor volumen en el conjunto de prueba por lo que cambios en la selección de instancias para ser evaluadas pueden reflejarse en mayores variaciones en el desempeño de clasificación de dicha clase.

Dicha estabilidad del clasificador jerárquico en función de la profundidad máxima se ejemplifica en los valores promedios de sus matrices de confusión, las que, según se puede ver en la Figura 11, resultan similares a pesar de la diferencia de profundidad máxima de los árboles del modelo.

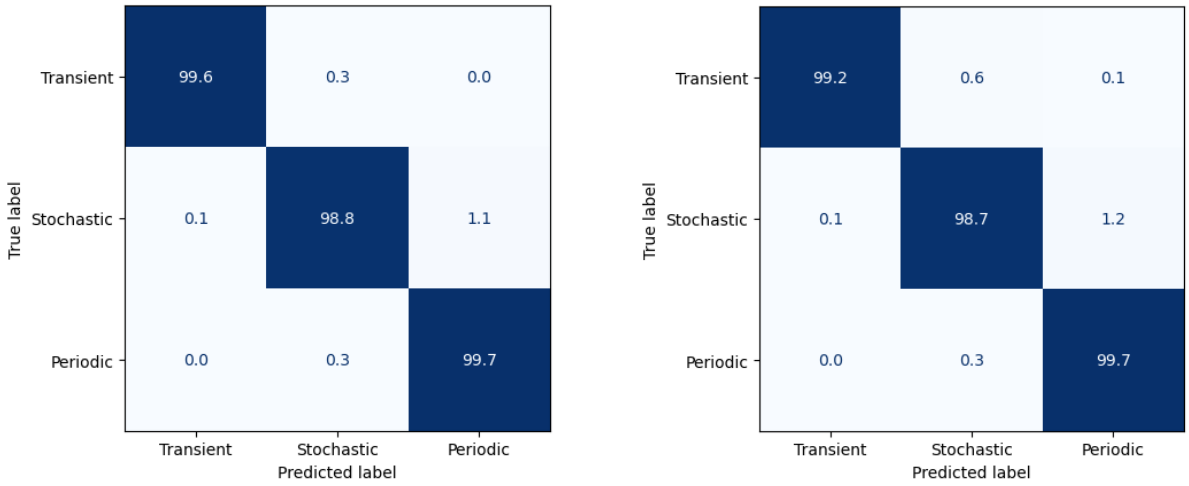


Figura 11: Matrices de confusión de clasificador Jerárquico *XGBoost* según la profundidad máxima. (a) Max_depth : 3, (b) Max_depth : 10.

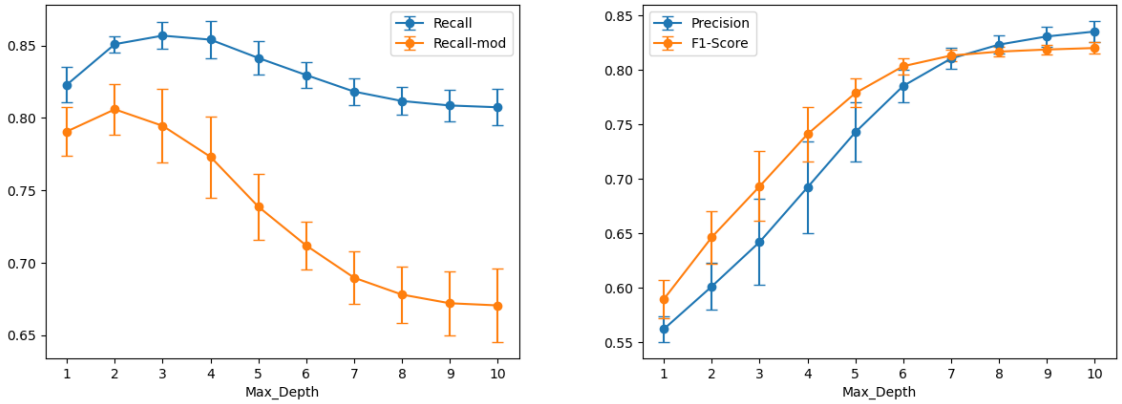
Como se puede observar en la Tabla 2, se tienen resultados muy altos para todas las métricas estudiadas, que resultan incluso mayores que los obtenidos con *BRF*. Estos resultados se ven acompañados también de desviaciones estándar relativamente bajas, lo que muestra que el uso de *XGBoost* con *Cost-Sensitive Learning* resulta una opción prometedora para el nivel superior del modelo encargado de predecir la clase jerárquica de las curvas.

Tabla 2: Desempeño en métricas de clasificación obtenidas para el clasificador Jerárquico *XGBoost*. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Métrica	$m\_d : 3$	$m\_d : 10$
$Precision$	$0{,}99 \pm 0{,}002$	$0{,}99 \pm 0{,}002$
$Recall$	$0{,}99 \pm 0{,}001$	$0{,}99 \pm 0{,}001$
$F1 - Score$	$0{,}99 \pm 0{,}001$	$0{,}99 \pm 0{,}001$
Min_recall	$0{,}99 \pm 0{,}001$	$0{,}99 \pm 0{,}001$
$Recall\_mod$	$1{,}00 \pm 0{,}002$	$0{,}99 \pm 0{,}003$

4.2.2. Clasificador Periódico

Siguiendo la metodología propuesta, para cada división de la validación cruzada se entrena un clasificador de curvas periódicas buscando la combinación óptima de hiperparámetros validados según su desempeño en obtener un *recall* alto para el modelo. Al observar las métricas de clasificación en función de la profundidad máxima en la Figura 12, se puede notar que a diferencia del clasificador jerárquico, el valor de la profundidad máxima de los árboles del modelo tiene un efecto significativo en el desempeño de éste.



(a) (b)

Figura 12: Métricas de clasificación del clasificador periódico *XGBoost* en función del parámetro de profundidad máxima. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

Resulta interesante analizar el hecho de que existe un "*trade-off*" o compromiso entre las métricas de *recall* y *precision*, disminuyendo en cierta medida la primera cuando aumenta la profundidad máxima mientras que la segunda aumenta. La métrica de $recall_{mod}$ presenta un comportamiento similar al del *recall* pero de forma más pronunciada. Esto cumple las expectativas de esta métrica dado que fue propuesta como una alternativa para visualizar de forma más notoria los efectos del desbalance de datos.

Existen distintos puntos de interés en los gráficos de la Figura 12. En la Figura 13 se muestran las matrices de confusión para los puntos con profundidad máxima 2, 3, 4 y 7 con el fin de estudiar la clasificación en el máximo del *recall*, del $recall_{mod}$ y cuando ya tiene un valor alto de *precision* y *f1-score* a costo de las métricas de *recall*.

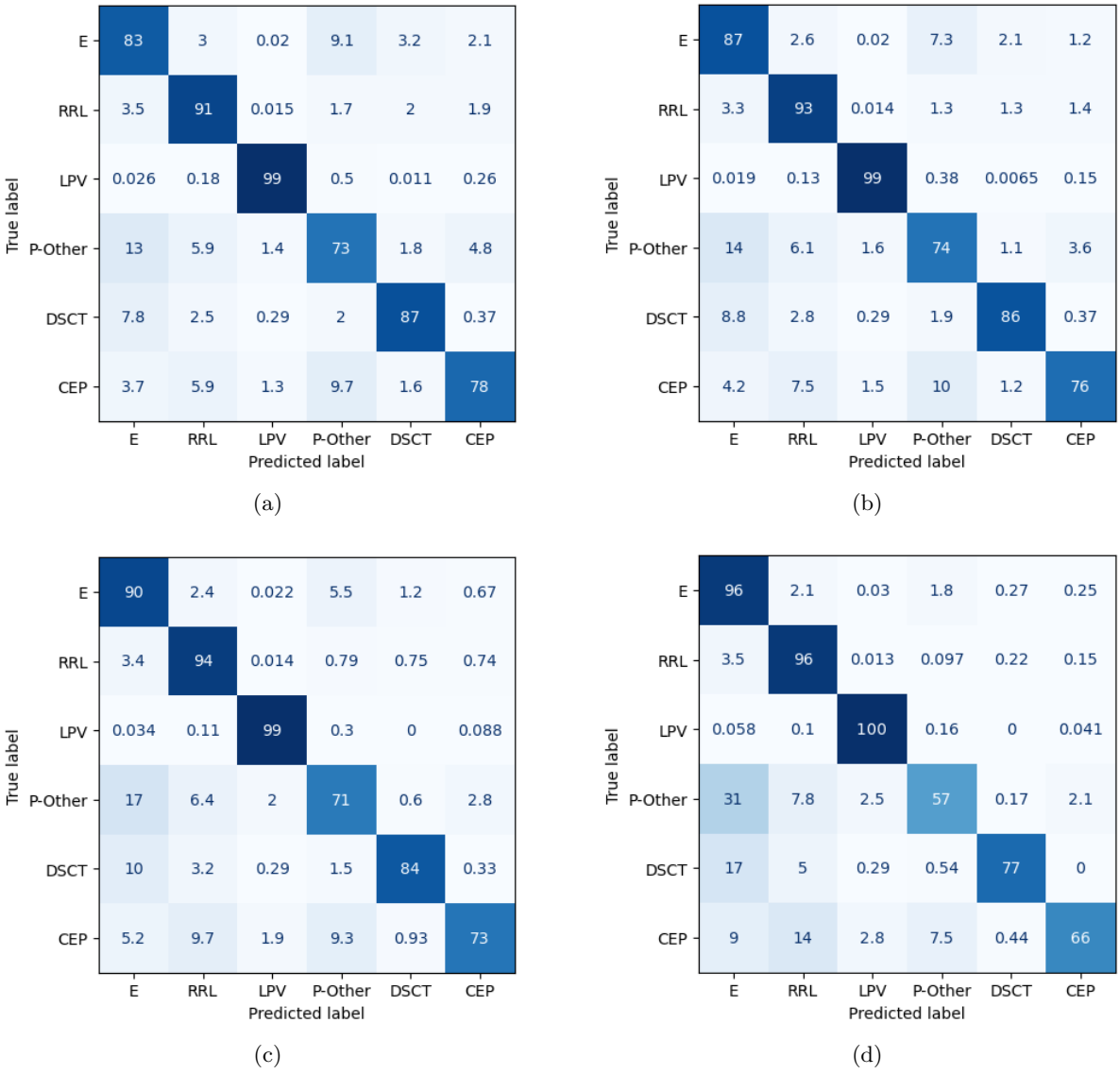


Figura 13: Matrices de confusión de clasificador *XGBoost* de curvas periódicas utilizando Cost-Sensitive Learning según la profundidad máxima. (a) Max_depth : 2, (b) Max_depth : 3, (c) Max_depth : 4, (d) Max_depth : 7.

Al analizar las matrices de confusión presentadas en la Figura 13, se puede notar que a medida que aumenta la profundidad máxima de los árboles, el porcentaje de clasificación promedio aumenta para algunas clases mientras que disminuye en otras. En particular las clases mayoritarias E, RRL y LPV presentan un aumento en el porcentaje promedio de predicciones correctas, mientras que por otro lado, las 3 clases con menor volumen en el conjunto de datos (*Periodic-Other*, DSCT y CEP) presentan un peor desempeño en su clasificación cuando se tiene una profundidad máxima mayor.

Tabla 3: Desempeño en métricas de clasificación obtenidas para el clasificador *XGBoost* de curvas periódicas usando distintas profundidades máximas (m_d). *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

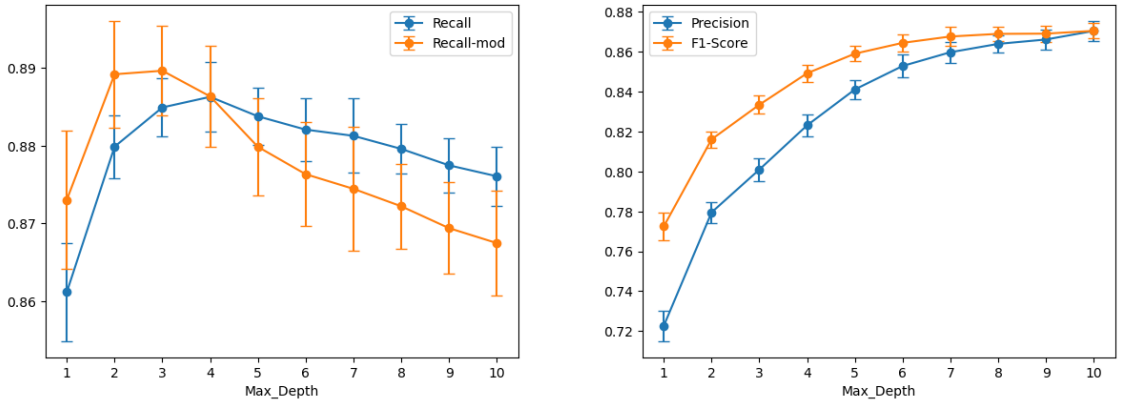
Métrica	$m\_d : 2$	$m\_d : 3$	$m\_d : 4$	$m\_d : 7$
$Precision$	$0{,}60 \pm 0{,}022$	$0{,}64 \pm 0{,}040$	$0{,}69 \pm 0{,}042$	$0{,}81 \pm 0{,}009$
$Recall$	$0{,}85 \pm 0{,}006$	$0{,}86 \pm 0{,}009$	$0{,}85 \pm 0{,}013$	$0{,}82 \pm 0{,}009$
$F1 - Score$	$0{,}65 \pm 0{,}024$	$0{,}69 \pm 0{,}032$	$0{,}74 \pm 0{,}025$	$0{,}81 \pm 0{,}005$
Min_recall	$0{,}73 \pm 0{,}014$	$0{,}73 \pm 0{,}032$	$0{,}70 \pm 0{,}052$	$0{,}57 \pm 0{,}031$
$Recall\_mod$	$0{,}81 \pm 0{,}018$	$0{,}79 \pm 0{,}026$	$0{,}77 \pm 0{,}028$	$0{,}69 \pm 0{,}018$

Analizando los valores promedios para las distintas métricas presentados en la Tabla 3, se observa que para el caso de profundidad máxima 7, se obtienen los mayores valores promedio de *precision* y *recall*, con el *trade-off* de presentar así también un valor de *recall* promedio ligeramente menor y valores de min_recall y $recall_{mod}$ notablemente menores. Este efecto, en conjunto con las matrices observadas en la Figura 13, ejemplifican la importancia de elegir una métrica de acuerdo al contexto y necesidades de un trabajo específico, pues seleccionar un modelo por su alto nivel de *f1-score* puede significar comprometerse a una clasificación de peor desempeño para las clases minoritarias del conjunto.

Es relevante también notar que si el objetivo es seleccionar un modelo que presente un buen desempeño en clasificar las clases minoritarias, estudiar solamente la métrica de *macro-recall* puede no ser representativo, puesto que en la Figura 13 se observa una disminución considerable del porcentaje de predicciones correctas en clases como *Periodic-Other* al pasar de utilizar una profundidad máxima de 4 a una de 7, pero el valor promedio del *recall* solo disminuye en 0.3, lo que puede verse como una disminución leve. En este contexto son útiles las métricas min_recall y $recall_{mod}$, la primera muestra la existencia de alguna clase que esté siendo clasificada incorrectamente (como en este caso, *Periodic-Other* con un *recall* de clase promedio de 0.57) y la segunda cuantiza con mayor notoriedad el efecto del sesgo hacia las clases mayoritarias, mostrando en el presente caso una disminución de 0.8 en su valor promedio al cambiar la profundidad máxima de los árboles del modelo de 4 a 7.

4.2.3. Clasificador Estocástico

El análisis para el clasificador de curvas estocásticas se realiza mediante la misma metodología que la utilizada en el clasificador de curvas periódicas, mostrándose en la Figura 14 gráficos de los valores promedios de las distintas métricas, junto a sus desviaciones estándar respectivas, en función de la profundidad máxima definida para los árboles del modelo.



(a) (b)

Figura 14: Métricas de clasificación del clasificador estocástico *XGBoost* en función del parámetro de profundidad máxima. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

Nuevamente se observa que la *precision* y el *f1-score* aumentan cuando el valor del parámetro *max_depth* aumenta, mientras que el aumento de este parámetro hace que, tras cierto punto, tanto el *recall* como el $recall_{mod}$ disminuyan.

Se puede observar que el valor promedio del *recall* y del $recall_{mod}$ alcanzan su punto máximo para distintos valores de profundidad máxima. Esta diferencia se debe al peso que asigna el $recall_{mod}$ a las clases minoritarias, lo que hace que esta métrica alcance valores altos aunque las clases mayoritarias estén siendo clasificadas de peor manera. Cabe mencionar que este efecto se da cuando la profundidad máxima es menor a 4, al punto de que se están clasificando de mejor manera las clases minoritarias que las mayoritarias, efecto que se deduce al observar que para estos niveles de *max_depth* el valor promedio de $recall_{mod}$ es mayor al valor promedio de *recall*. A pesar de esto, es relevante considerar el valor de la desviación estándar, que nuevamente es mayor para el caso de $recall_{mod}$ debido a las razones anteriormente descritas.

Con el fin de ejemplificar la clasificación en los casos en que distintas métricas alcanzan un valor alto, en la Figura 15 se muestran las matrices de confusión del clasificador de curvas estocásticas para valores de profundidad máxima de 2, 3, 4 y 10.

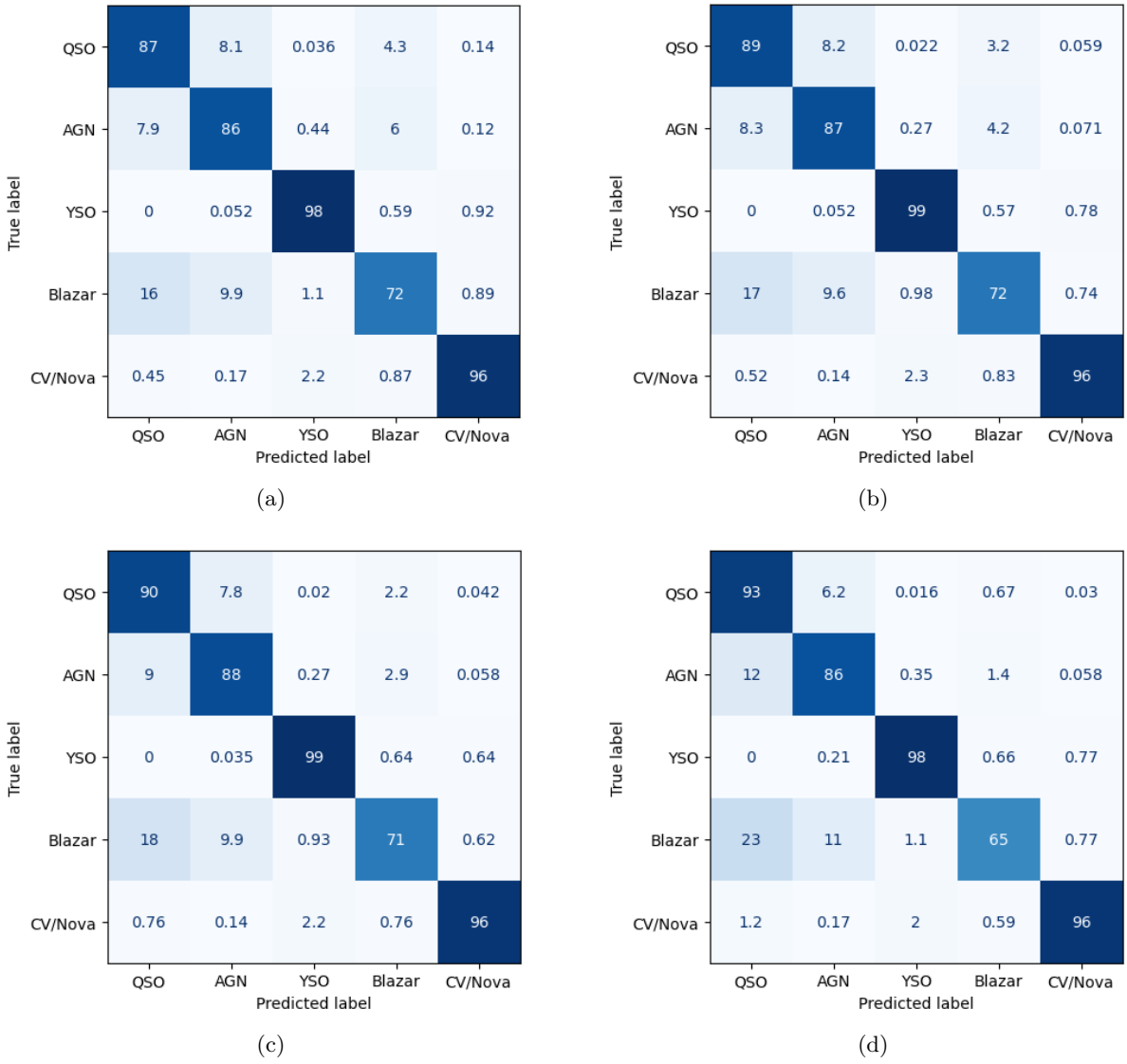


Figura 15: Matrices de confusión de clasificador *XGBoost* de curvas estocásticas utilizando Cost-Sensitive Learning según la profundidad máxima. (a) Max_depth : 2, (b) Max_depth : 3, (c) Max_depth : 4, (d) Max_depth : 10.

Al estudiar las matrices de la Figura 15 es interesante identificar que en este caso no existe una relación directa entre representación de la clase en el conjunto de datos y la variación que tiene ésta al alterar la profundidad máxima. Esto esta ejemplificado principalmente en el hecho de que la clase CV/Nova, a pesar de ser la clase minoritaria del conjunto, no varía su porcentaje promedio de clasificación en función de la profundidad máxima de los árboles del modelo, de forma similar a la clase YSO. Sin embargo, la clase en la que si observa esta variación es la clase *Blazar*, en la que se observa una disminución en el valor promedio del desempeño del modelo cuando se aumenta la profundidad de 4 a 10.

Tabla 4: Desempeño en métricas de clasificación obtenidas para el clasificador *XGBoost* de curvas estocásticas usando distintas profundidades máximas (m_d). *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Métrica	$m\_d : 2$	$m\_d : 3$	$m\_d : 4$	$m\_d : 10$
$Precision$	$0{,}78 \pm 0{,}01$	$0{,}80 \pm 0{,}01$	$0{,}82 \pm 0{,}01$	$0{,}87 \pm 0{,}01$
$Recall$	$0{,}88 \pm 0{,}00$	$0{,}88 \pm 0{,}00$	$0{,}89 \pm 0{,}00$	$0{,}88 \pm 0{,}00$
$F1 - Score$	$0{,}82 \pm 0{,}00$	$0{,}83 \pm 0{,}00$	$0{,}85 \pm 0{,}00$	$0{,}87 \pm 0{,}00$
Min_recall	$0{,}72 \pm 0{,}02$	$0{,}72 \pm 0{,}02$	$0{,}71 \pm 0{,}03$	$0{,}65 \pm 0{,}02$
$Recall\_mod$	$0{,}89 \pm 0{,}01$	$0{,}89 \pm 0{,}01$	$0{,}89 \pm 0{,}01$	$0{,}87 \pm 0{,}01$

Analizando la Tabla 4 se observa nuevamente que los valores de *precision* y *f1-score* llegan a su valor promedio máximo cuando se tiene un valor alto para la profundidad máxima. Por otro lado, las métricas de min_recall y $recall_{mod}$ muestran un leve cambio en el balance en la predicción de una forma más notoria que la métrica *recall*.

Periodic

Clasificador	Precision	Recall	F1-Score
XGBoost Original	$0{,}87 \pm 0{,}01$	$0{,}75 \pm 0{,}007$	$0{,}8 \pm 0{,}007$
Cost-Sensitive Learning	$0{,}6 \pm 0{,}022$	$0{,}85 \pm 0{,}006$	$0{,}65 \pm 0{,}024$
Balanced Bootstrap	$0{,}64 \pm 0{,}01$	$0{,}85 \pm 0{,}01$	$0{,}69 \pm 0{,}013$

Estocastic

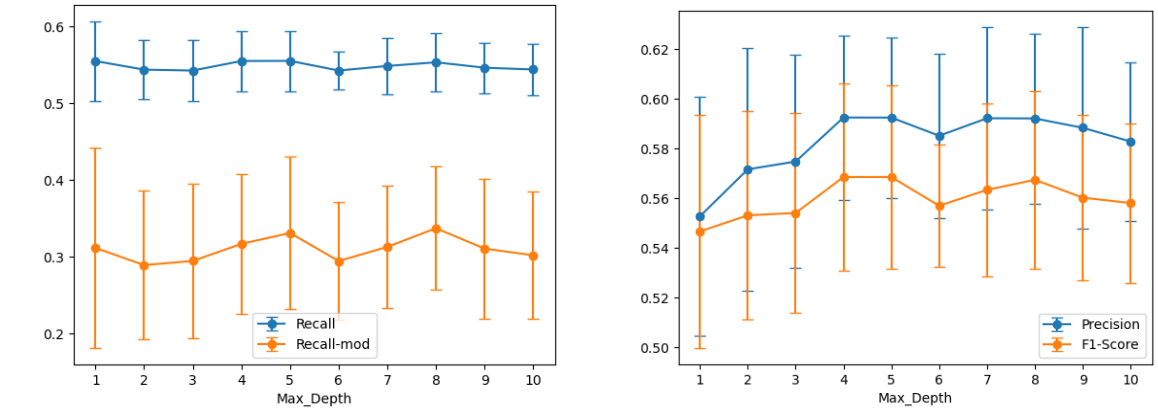
Clasificador	Precision	Recall	F1-Score
XGBoost Original	$0{,}91 \pm 0{,}004$	$0{,}85 \pm 0{,}005$	$0{,}88 \pm 0{,}004$
Cost-Sensitive Learning	$0{,}78 \pm 0{,}01$	$0{,}88 \pm 0{,}00$	$0{,}82 \pm 0{,}00$
Balanced Bootstrap	$0{,}79 \pm 0{,}01$	$0{,}87 \pm 0{,}01$	$0{,}83 \pm 0{,}01$

Transient

Clasificador	Precision	Recall	F1-Score
XGBoost Original	$0{,}68 \pm 0{,}14$	$0{,}48 \pm 0{,}032$	$0{,}48 \pm 0{,}048$
Cost-Sensitive Learning	$0{,}59 \pm 0{,}03$	$0{,}56 \pm 0{,}04$	$0{,}57 \pm 0{,}04$
Balanced Bootstrap	$0{,}53 \pm 0{,}025$	$0{,}63 \pm 0{,}047$	$0{,}56 \pm 0{,}03$

4.2.4. Clasificador Transiente

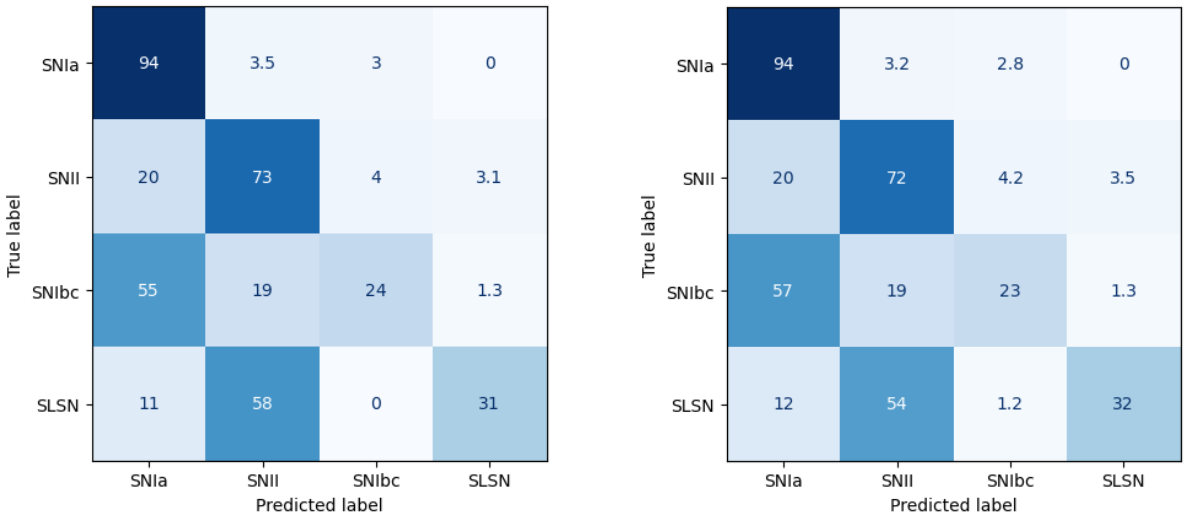
Los gráficos de los valores promedios y desviación estándar de las distintas métricas en función de la profundidad máxima para el clasificador de curvas transientes se muestra en la figura 16.



(a) (b)

Figura 16: Métricas de clasificación del clasificador transiente *XGBoost* en función del parámetro de profundidad máxima. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

A diferencia del caso del clasificador periódico o estocástico, se puede observar que las métricas para el clasificador transiente no presentan una correlación definida con el valor de la profundidad máxima de los árboles del modelo. Por este motivo se muestran en la Figura 17 dos casos para ejemplificar el efecto de la profundidad en la clasificación.



(a) (b)

Figura 17: Matrices de confusión de clasificador *XGBoost* de curvas transientes utilizando Cost-Sensitive Learning según la profundidad máxima. (a) Max_depth : 5, (b) Max_depth : 8.

Al analizar las matrices de confusión de la Figura 17 se observa que no existe diferencia signifi-

cativa entre ambas, lo que se relaciona directamente con las curvas de la Figura 16. Se puede ver en ambas matrices que el modelo realiza la predicción con un alto sesgo hacia las clases mayoritarias SNIa y SNII, lo que puede deberse principalmente a que las curvas de clase transiente son las menos representadas en el conjunto de datos, por lo que las sub-clases minoritarias dentro de este subconjunto cuentan con un volumen de instancias que no son suficientes para que el modelo aprenda a distinguirlas, incluso al aplicar *Cost-Sensitive Learning* en su algoritmo.

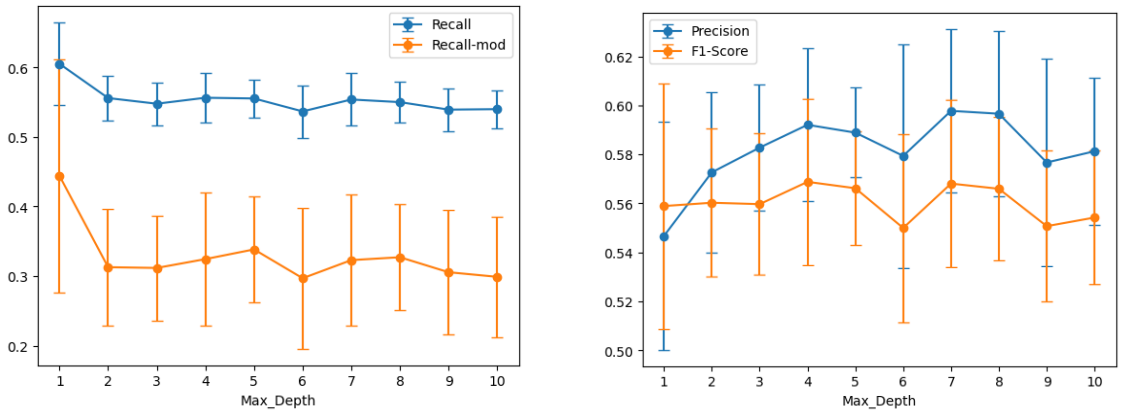
Tabla 5: Desempeño en métricas de clasificación obtenidas para el clasificador *XGBoost* de curvas transientes usando distintas profundidades máximas (m_d). *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Métrica	$m\_d : 5$	$m\_d : 8$
$Precision$	$0{,}59 \pm 0{,}03$	$0{,}59 \pm 0{,}03$
$Recall$	$0{,}56 \pm 0{,}04$	$0{,}55 \pm 0{,}04$
$F1 - Score$	$0{,}57 \pm 0{,}04$	$0{,}57 \pm 0{,}04$
Min_recall	$0{,}21 \pm 0{,}07$	$0{,}22 \pm 0{,}08$
$Recall\_mod$	$0{,}33 \pm 0{,}10$	$0{,}34 \pm 0{,}08$

Algo relevante de los resultados presentados en la Tabla 5, es el alto valor de la desviación estándar de los resultados en comparación con resultados anteriores. Estos valores elevados denotan una alta variabilidad en la clasificación de curvas transientes, la que se debe principalmente a la poca representación que tienen dentro del conjunto de datos.

4.2.5. Clasificador Transiente con validación usando recall mínimo

Con el objetivo de disminuir el sesgo elevado que presenta *XGBoost* a las clases mayoritarias al clasificar curvas de clase transiente, se evaluó también el desempeño de dicho clasificador al entrenarse usando como métrica de validación el *min-recall*. Ésta se utiliza teniendo en consideración que las clases minoritarias presentaban un porcentaje de clasificación excepcionalmente bajo, por lo que se busca priorizar que el modelo tenga el mejor desempeño posible para predecir la clase que resulte peor clasificada.

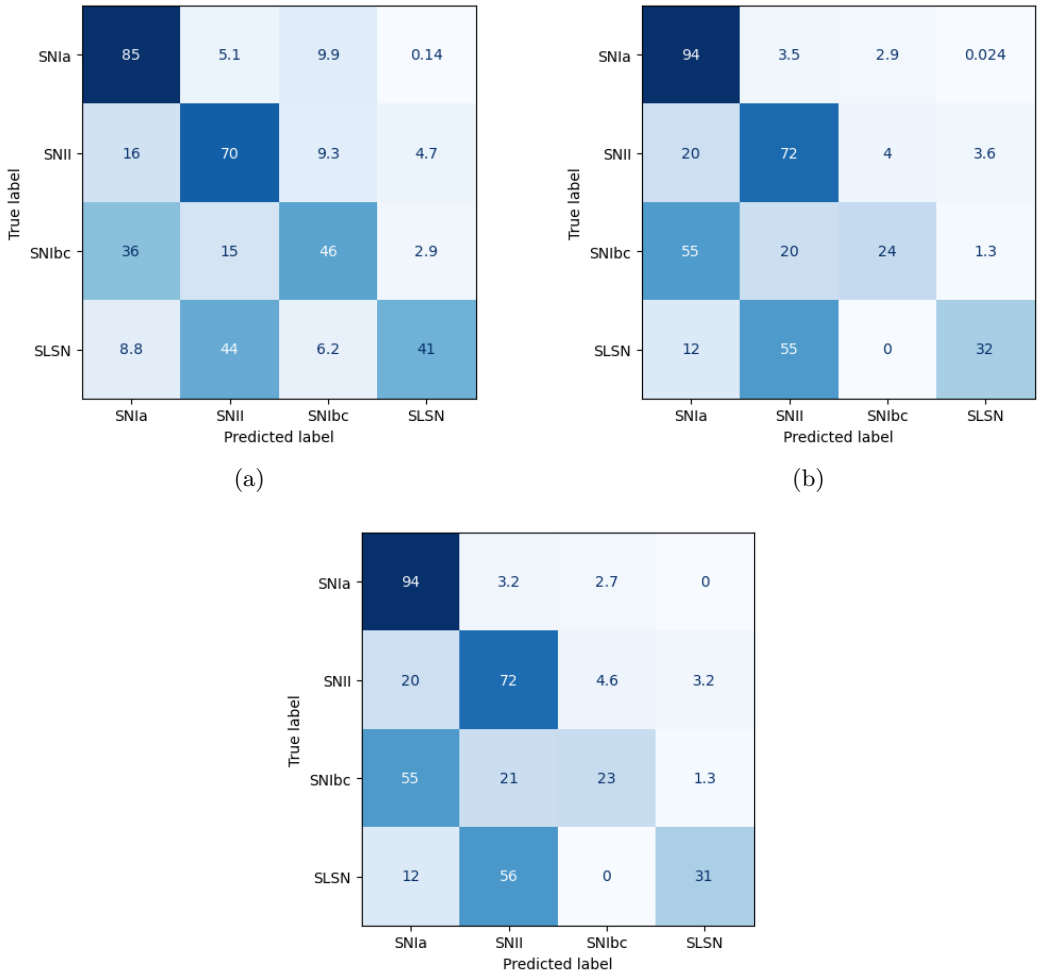


(a) (b)

Figura 18: Métricas de clasificación del clasificador transiente *XGBoost* con validación usando *recall mínimo* en función del parámetro de profundidad máxima. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

En la Figura 18 se observa que no hay una correlación notable entre las métricas estudiadas y el valor de profundidad máxima, sin embargo, para el caso de profundidad máxima 1, es decir, cuando el modelo se construye únicamente con árboles que separan el conjunto de datos en 2, se obtiene un valor promedio de *recall* y $recall_{mod}$ levemente mayor.

Al observar las matrices de confusión en la Figura 19, se puede notar que el aumento en el valor promedio de las métricas mencionadas se traduce en un ligero aumento en el porcentaje promedio de predicción correcta para las curvas de clases minoritarias SNIbc y SLSN. A pesar de esto, no es correcto afirmar que un clasificador transiente de profundidad máxima 1 tendrá mejor desempeño que uno con un valor de profundidad mayor, dado que hay que tomar en cuenta la varianza. Además, la mejora en la clasificación de las clases SNIbc y SLSN conlleva una disminución notable en el desempeño promedio de clasificación para la clase SNIa y una leve disminución en el desempeño promedio de la clase SNII.



(a) (b)

(c)

Figura 19: Matrices de confusión de clasificador *XGBoost* de curvas transientes utilizando Cost-Sensitive Learning con *recall mínimo* como métrica de validación según la profundidad máxima. (a) Max_depth : 1, (b) Max_depth : 5, (c) Max_depth : 8.

Tabla 6: Desempeño en métricas de clasificación obtenidas para el clasificador *XGBoost* de curvas transientes con *recall mínimo* como métrica de validación usando distintas profundidades máximas (m_d). *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Métrica	$m\_d : 1$	$m\_d : 5$	$m\_d : 8$
$Precision$	$0{,}55 \pm 0{,}05$	$0{,}59 \pm 0{,}02$	$0{,}60 \pm 0{,}03$
$Recall$	$0{,}61 \pm 0{,}06$	$0{,}56 \pm 0{,}03$	$0{,}55 \pm 0{,}03$
$F1 - Score$	$0{,}56 \pm 0{,}05$	$0{,}57 \pm 0{,}02$	$0{,}57 \pm 0{,}03$
Min_recall	$0{,}36 \pm 0{,}18$	$0{,}22 \pm 0{,}06$	$0{,}22 \pm 0{,}06$
$Recall\_mod$	$0{,}44 \pm 0{,}17$	$0{,}34 \pm 0{,}08$	$0{,}33 \pm 0{,}08$

4.2.6. Clasificador Completo

Como se mencionó en el capítulo 3, para realizar la predicción sobre una curva de luz cuya clase es desconocida se utiliza una combinación de los cuatro clasificadores mencionados, ponderando la probabilidad de cada una de las tres clases dadas por el clasificador jerárquico (*Periodic*, *Stochastic* o *Transient*) por el vector de probabilidades dado por cada uno de los clasificadores de nivel inferior.

Para seleccionar los parámetros de cada clasificador se seleccionó un valor de max_depth según lo observado en los resultados obtenidos y se utilizaron los hiperparámetros obtenidos en la optimización realizada con dicho valor de max_depth. Cabe destacar que para el caso del clasificador transiente se utilizó la configuración obtenida al utilizar min_recall como métrica de validación.

Los valores de profundidad máxima utilizados para entrenar el clasificador compuesto son:

- Clasificador Jerárquico: 4
- Clasificador Periódico: 3
- Clasificador Estocástico: 3
- Clasificador Transiente: 1

Con esta configuración se obtuvo la matriz de confusión presentada en la Figura 20.

Los resultados de la Tabla 7 muestran una mejoría en el valor promedio de las métricas de *precision* y *f1-score* en comparación con *BRF* sin variar el valor de *recall*. Sin embargo, al observar el valor promedio de min_recall, queda en evidencia que existen clases con un porcentaje de clasificación correcta sumamente bajo, correspondientes principalmente a las clases SNIbc y SLSN, según se puede ver en la Figura 20. Al ser estas clases minoritarias muy poco representadas en el conjunto de datos, son consideradas con un alto peso al calcular $recall_{mod}$, lo que explica el bajo valor promedio de esta métrica.

Estos resultados ejemplifican la problemática presente en distintos casos de los estudiados en el presente trabajo, que corresponde al *trade-off* existente entre el aumento del valor de una métrica de desempeño a costa de la disminución del valor de otra. Esta dificultad para interpretar la calidad de la clasificación por parte de un modelo afecta principalmente a las clases minoritarias, que en muchos contextos corresponden a las clases que tienen una importancia mayor.

Tabla 7: Desempeño en macro-métricas de clasificación obtenidas para el clasificador *XGBoost* completo.

Clasificador	$Precision$	$Recall$	$F1-Score$	Min_recall	$Recall\_mod$
XGB Completo	$0{,}72 \pm 0{,}003$	$0{,}76 \pm 0{,}01$	$0{,}76 \pm 0{,}02$	$0{,}34 \pm 0{,}1$	$0{,}48 \pm 0{,}1$

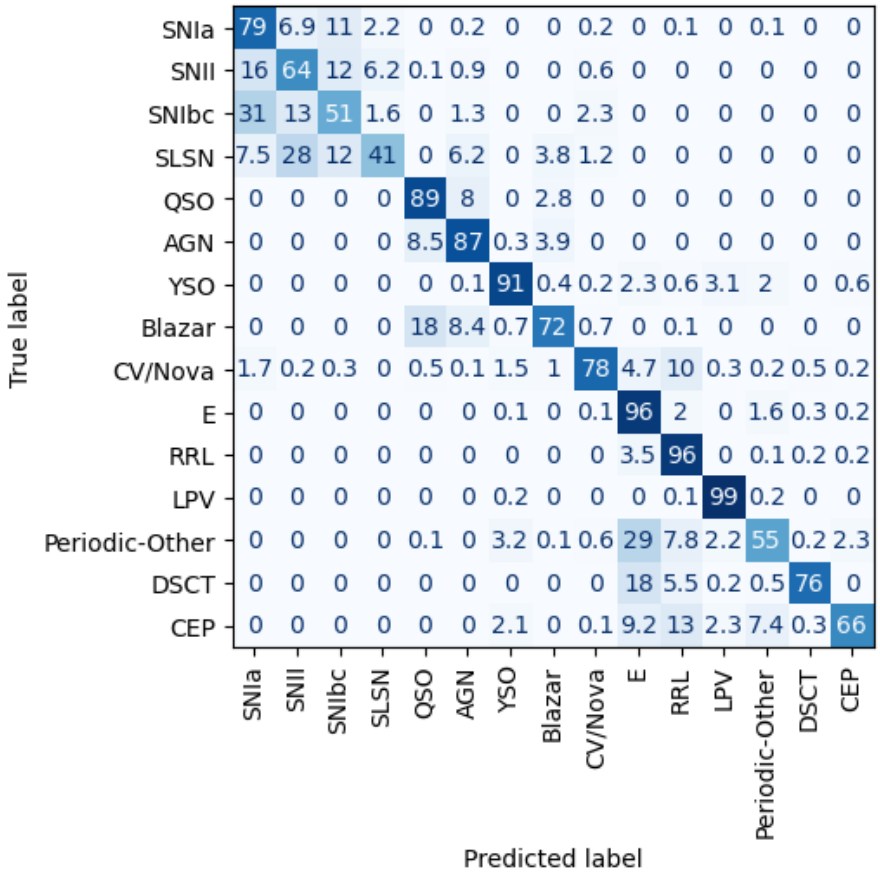


Figura 20: Matriz de confusión de clasificador *XGBoost* completo con *cost-sensitive learning*

4.3. Gradient Boosting con Balanced Bootstrapping

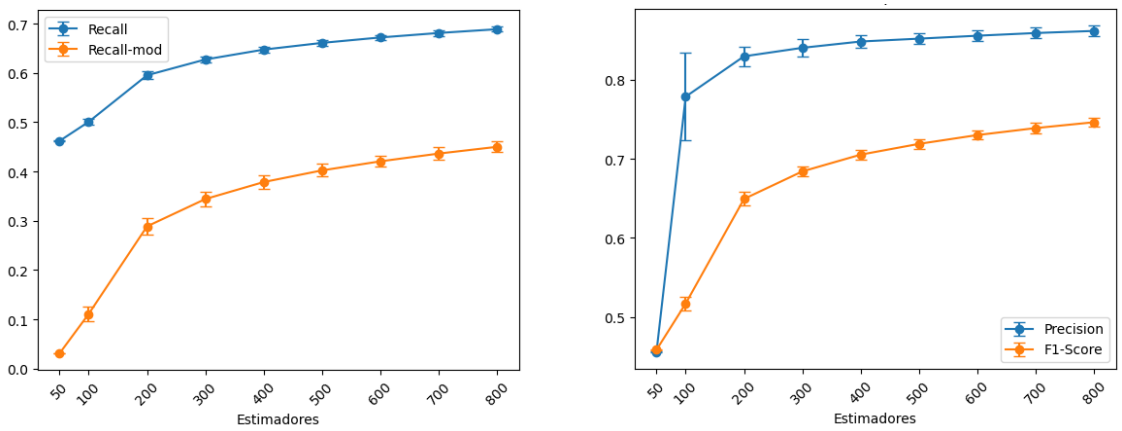
4.3.1. Gradient Boosting sin modificar

Para estudiar los efectos de utilizar un algoritmo de *bootstrapping* balanceado en un modelo de *Gradient Boosting*, se entrenan primeramente distintos clasificadores basados en *Gradient Boosting* sin modificar su algoritmo, y se analiza su desempeño de clasificación.

Al agregar una fase de *bootstrapping* de los datos al algoritmo de *Gradient Boosting*, se requerirá construir un mayor número de árboles de decisión para el modelo debido a que cada árbol se construye con una cantidad menor de datos, y en caso de alto desbalance de clase, un *bootstrapping* balanceado como el que se implementó en el presente trabajo conlleva una disminución del volumen de datos utilizado para construir cada árbol. Por esto, es relevante estudiar el desempeño de los modelos en función de la cantidad de estimadores con los que se entrenan. De esta forma se espera que al aumentar el número de estimadores en el modelo, aumente la representatividad de los datos y asimismo su desempeño. También se puede analizar cómo se refleja el efecto esperado de balance en el desempeño de clasificación en base a la cantidad de iteraciones en las que se utiliza el *bootstrapping*

balanceado implementado.

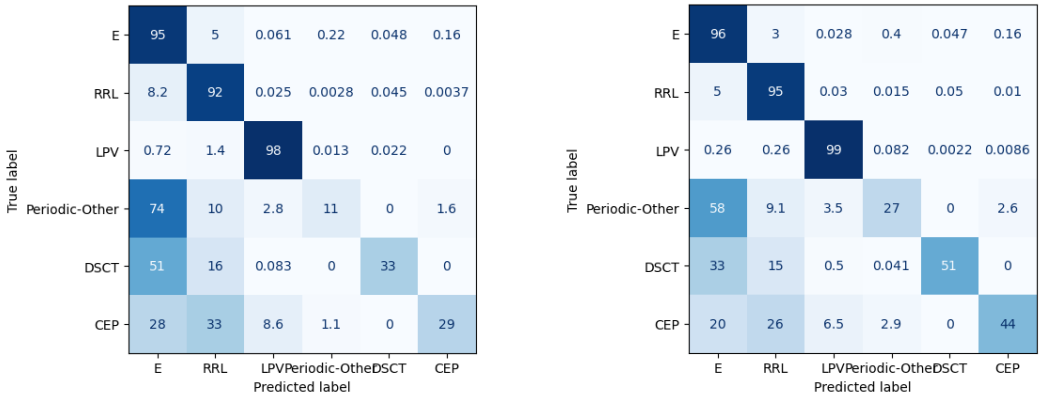
Por este motivo, la obtención de resultados de *Gradient Boosting* sin modificar se realiza igualmente en función del número de estimadores del modelo, con el fin de analizar si la diferencia en la clasificación al implementar *bootstrapping* se debe en principal medida a esta modificación o al cambio en el número de estimadores.



(a) (b)

Figura 21: Métricas de clasificación de curvas periódicas para el del clasificador *Gradient Boosting* original en función del número de estimadores. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

En la Figura 21 se observa que el desempeño en todas las métricas del modelo aumenta al agregar estimadores al sistema, con una pendiente decreciente. Este es el comportamiento esperado de un algoritmo de *machine learning* como *Gradient Boosting* ya que muestra el aumento en su desempeño que se atenúa a lo largo del entrenamiento debido a que se ajusta poco a poco al conjunto de entrenamiento.



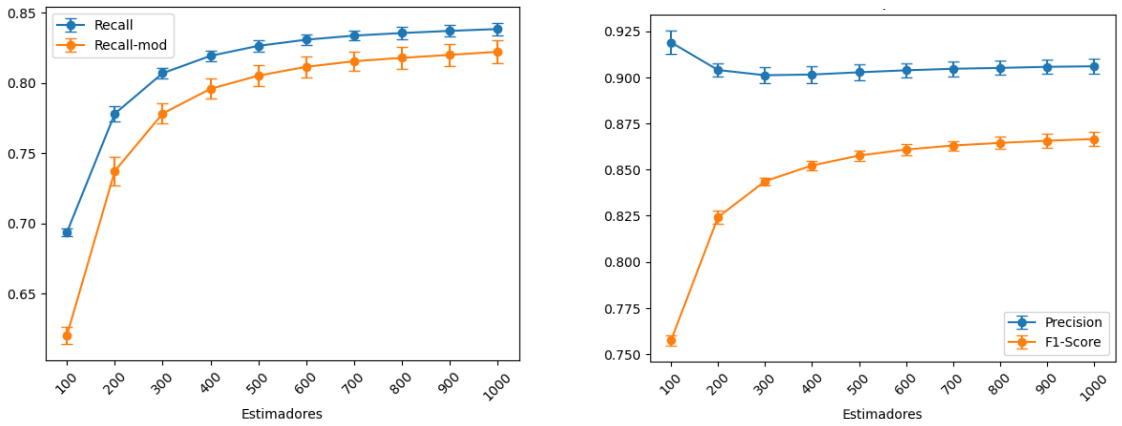
(a) (b)

Figura 22: Matrices de confusión de clasificador *Gradient Boosting* original para curvas periódicas. (a) 200 estimadores, (b) 800 estimadores.

En la Figura 22 se puede observar el sesgo presente en la clasificación del modelo, expresado en la cantidad de instancias de las clases minoritarias que son incorrectamente predichas como pertenecientes a las clases mayoritarias E y RRL. Este sesgo disminuye al aumentar el número de árboles en el modelo, sin embargo, cabe notar que en la Tabla 8 la métrica de *precision* tiene valores altos incluso en los puntos en donde se puede observar visualmente que existe un gran desbalance en los porcentajes de predicciones correctas hechas por el modelo.

Tabla 8: Desempeño en métricas de clasificación obtenidas para el clasificador *Gradient Boosting* original para curvas periódicas usando distintos números de estimadores. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Métrica	*Estimadores: 200*	*Estimadores: 800*
$Precision$	$0{,}83 \pm 0{,}01$	$0{,}86 \pm 0{,}01$
$Recall$	$0{,}60 \pm 0{,}01$	$0{,}69 \pm 0{,}01$
$F1 - Score$	$0{,}65 \pm 0{,}01$	$0{,}75 \pm 0{,}01$
Min_recall	$0{,}11 \pm 0{,}01$	$0{,}27 \pm 0{,}02$
$Recall\_mod$	$0{,}29 \pm 0{,}02$	$0{,}45 \pm 0{,}01$



(a) (b)

Figura 23: Métricas de clasificación de curvas estocásticas para el clasificador *Gradient Boosting* original en función del número de estimadores. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

Para las curvas estocásticas se tiene un caso similar al observado en el desempeño del clasificador periódico. En la Figura 23 se observa que cuando se tienen pocos árboles en el modelo, se tiene un alto valor promedio de la métrica *precision*, sin embargo, al observar la matriz de confusión para 100 estimadores en la Figura 24, se tiene que hay un alto porcentaje de clasificaciones incorrectas por parte del modelo, debido principalmente a la clasificación objetos de la clase Blazar como de la clase QSO.

A pesar de que las clases minoritarias, en particular la clase Blazar, presentan un porcentaje de clasificación correcta sumamente bajo, un número alto de instancias de conjunto de datos pertenecientes a la clase QSO están siendo clasificadas correctamente, por lo que se tiene un valor promedio de *precision* elevado. Esto sirve para ejemplificar nuevamente la importancia de evaluar distintas métricas de desempeño y seleccionar la que se considere conveniente, ya que si se tuviera como prioridad clasificar correctamente instancias de la clase QSO sin darle mayor relevancia a otras clases, la métrica de *precision* puede ser un buen indicativo sobre el desempeño del modelo bajo los estándares que se esperan en dicho contexto.

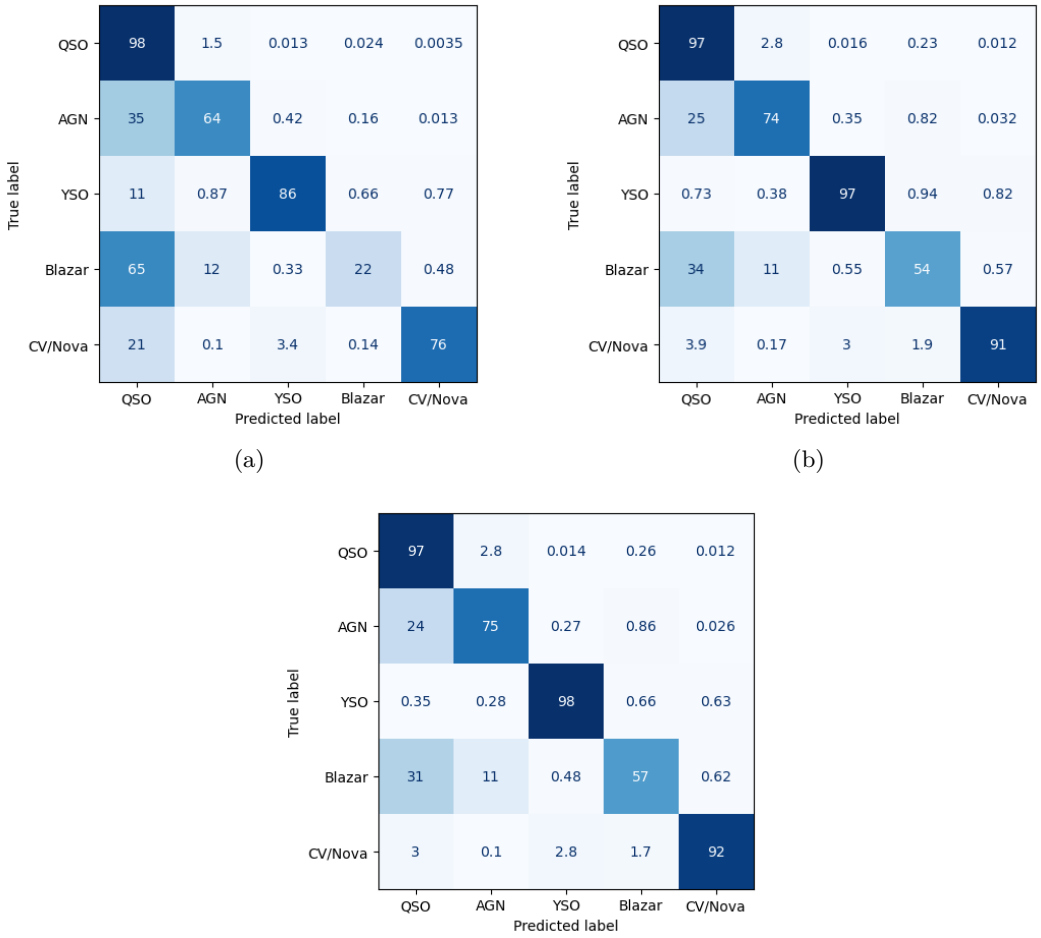


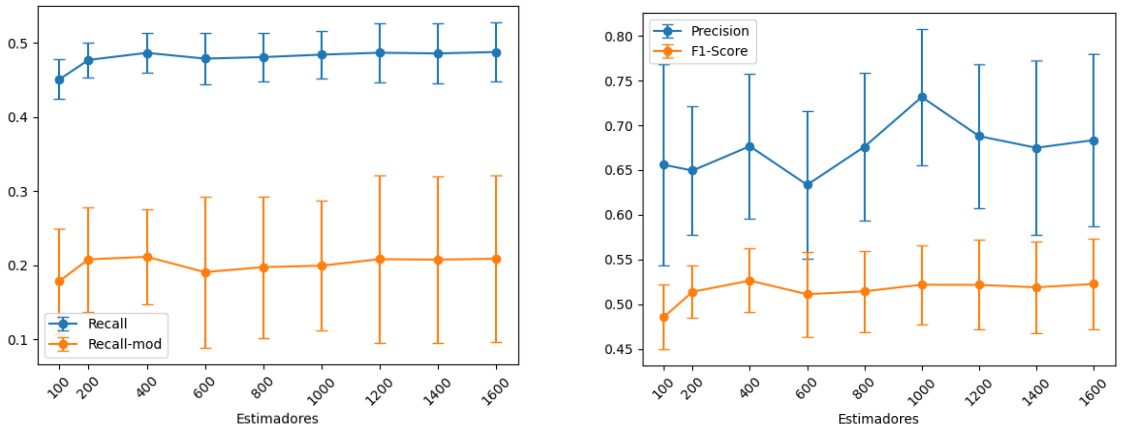
Figura 24: Matrices de confusión de clasificador *Gradient Boosting* original para curvas estocásticas. (a) 100 estimadores, (b) 500 estimadores, (c) 1000 estimadores.

Tabla 9: Desempeño en métricas de clasificación obtenidas para el clasificador *Gradient Boosting* original para curvas estocásticas usando distintos números de estimadores. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Métrica	$Estimadores$: 100	$Estimadores$: 500	$Estimadores$: 1000
$Precision$	$0{,}92 \pm 0{,}01$	$0{,}90 \pm 0{,}002$	$0{,}91 \pm 0{,}001$
$Recall$	$0{,}69 \pm 0{,}001$	$0{,}83 \pm 0{,}001$	$0{,}84 \pm 0{,}001$
$F1 - Score$	$0{,}76 \pm 0{,}001$	$0{,}86 \pm 0{,}002$	$0{,}87 \pm 0{,}002$
Min_recall	$0{,}22 \pm 0{,}02$	$0{,}54 \pm 0{,}02$	$0{,}57 \pm 0{,}02$
$Recall\_mod$	$0{,}62 \pm 0{,}01$	$0{,}81 \pm 0{,}01$	$0{,}82 \pm 0{,}01$

Una característica del clasificador *Gradient Boosting* que se deduce al analizar los resultados de

la Tabla 9 es su consistencia, la que se ve reflejada en los bajos valores de desviación estándar de los resultados obtenidos. También cabe destacar que el clasificador con 1000 estimadores supera al clasificador con 500 estimadores en todas las métricas, evidenciando el crecimiento del modelo.



(a) (b)

Figura 25: Métricas de clasificación de curvas transientes para el clasificador *Gradient Boosting* original en función del número de estimadores. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

Similarmente al caso estudiado con *XGboost*, se observa en la Figura 25 que el clasificador transiente presenta un desempeño diferente al de los clasificadores periódico y estocástico. En este caso, se observa que las métricas de desempeño no denotan una correlación con el número de árboles en el modelo.

(a) (b)

Figura 26: Matrices de confusión de clasificador *Gradient Boosting* original para curvas transientes. (a) 100 estimadores, (b) 1000 estimadores.

En las matrices de confusión de la Ffigura 26 es posible notar que se tiene un comportamiento en la clasificación similar al caso de *XGBoost* con *Cost-Sensitive Learning*, en donde el modelo predice erróneamente una alta cantidad de instancias de la clase SNIbc como SNIa, y de la clase SLSN como SNII. Sin embargo, los resultados presentados en la tabla 10 reflejan que en el caso de un clasificador basado en *Gradient Boosting* sin modificar el sesgo a dichas clases es aún mayor, lo que se debe principalmente al balance realizado por la estrategia de *Cost-Sensitive Learning* en *XGBoost*.

Tabla 10: Desempeño en métricas de clasificación obtenidas para el clasificador *Gradient Boosting* para curvas transientes usando distintos números de estimadores. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Métrica	$Estimadores$: 400	$Estimadores$: 1000
$Precision$	$0{,}68 \pm 0{,}08$	$0{,}73 \pm 0{,}08$
$Recall$	$0{,}49 \pm 0{,}03$	$0{,}48 \pm 0{,}03$
$F1 - Score$	$0{,}53 \pm 0{,}04$	$0{,}52 \pm 0{,}04$
Min_recall	$0{,}11 \pm 0{,}04$	$0{,}09 \pm 0{,}03$
$Recall\_mod$	$0{,}21 \pm 0{,}06$	$0{,}20 \pm 0{,}09$

4.3.2. Resultados con Gradient Boosting modificado

Al modificar el algoritmo de *Gradient Boosting* con la metodología descrita en el capítulo 3, se puede observar un cambio en el desempeño del clasificador en relación con el número de árboles en el modelo. La Figura 27 muestra que a pesar de que las métricas de *recall* mantienen un aumento en su valor promedio al aumentarse el número de estimadores, la métrica *precision* disminuye al superar un determinado umbral, ocasionando de la misma forma una disminución en el *f1-score*.

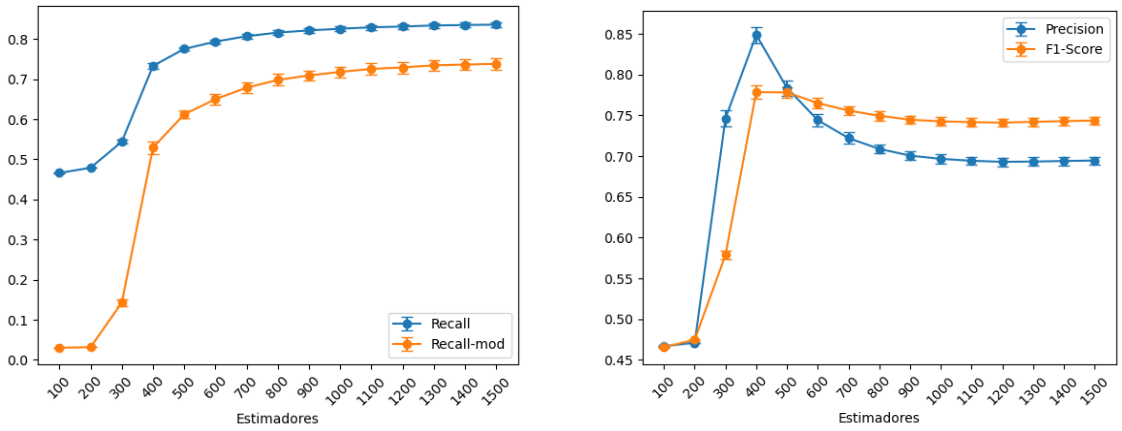


Figura 27: Métricas de clasificación de curvas periódicas para el clasificador *Gradient Boosting* modificado en función del número de estimadores. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

Para analizar este efecto se estudian las matrices de confusión en distintos puntos de interés

en el gráfico, las que se muestran en la Figura 28. En estas matrices se puede observar el efecto de este cambio de comportamiento de las métricas en la clasificación, notándose que al aumentar el número de árboles o estimadores en el clasificador, se obtiene un mejor desempeño promedio al predecir instancias de las clases minoritarias *Periodic-Other*, DSCT y CEP, a cambio de una disminución en el desempeño de predicción de la clase mayoritaria E. Estos resultados observables denotan que la modificación del algoritmo para incluir *bootstrap* balanceado en la construcción de cada árbol permite afrontar el problema de desbalance de datos, haciéndose notorio en mayor medida al aumentar el número de árboles en el algoritmo.

Sin embargo, se debe tener en consideración el hecho de que un alto número de árboles en el modelo conlleva una disminución en el desempeño promedio de clasificación de las clases mayoritarias, por lo que para un problema de clasificación específico se debe determinar cuánto se desea balancear el desempeño del clasificador antes de que dicha disminución de desempeño sea relevante.

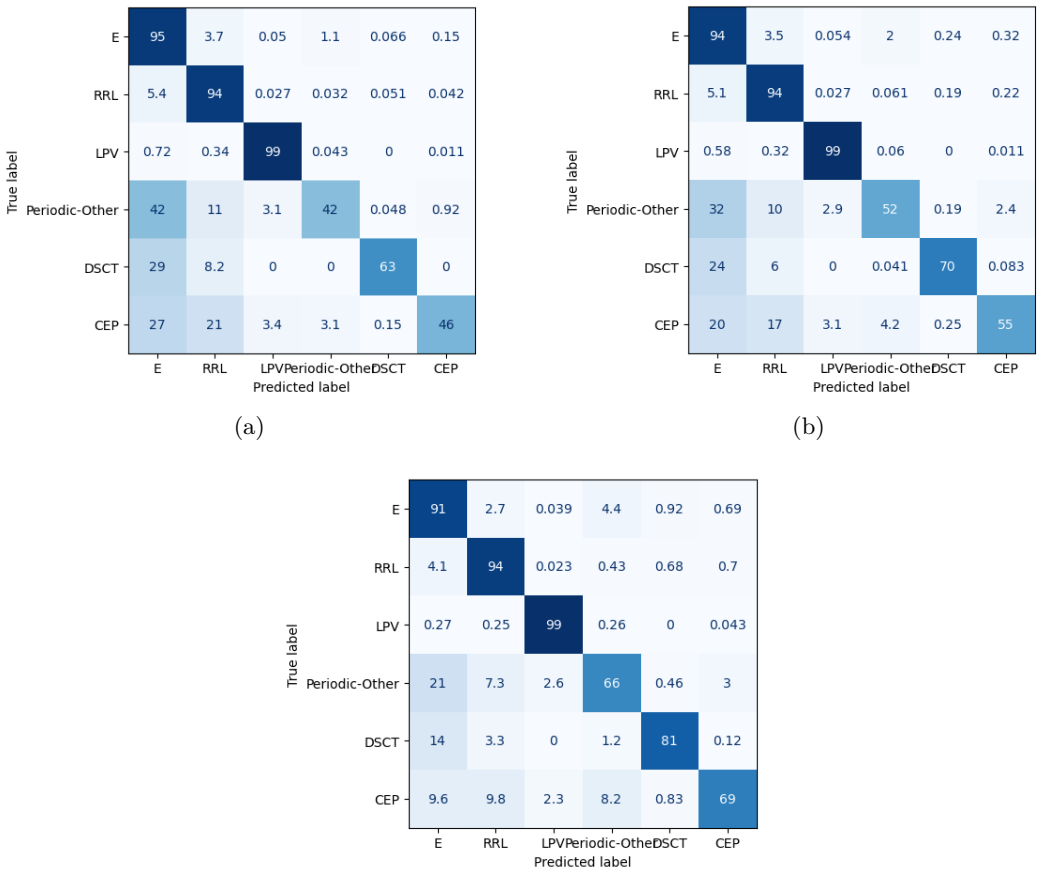


Figura 28: Matrices de confusión de clasificador *Gradient Boosting* modificado para curvas periódicas. (a) 400 estimadores, (b) 500 estimadores, (c) 1400 estimadores.

Los resultados presentados en la Tabla 11 confirman lo observado en las matrices de confusión y

en los gráficos, teniéndose para el caso de 1400 estimadores una mejora significativa en las métricas de *recall*, $recall\_min$ y $recall_{mod}$ en comparación al modelo de *Gradient Boosting* sin modificar, particularmente en las últimas, en donde existe una mejora considerable debido al hecho ya mencionado de que éstas permiten una interpretación con mayor notoriedad del desempeño del clasificador en clases minoritarias.

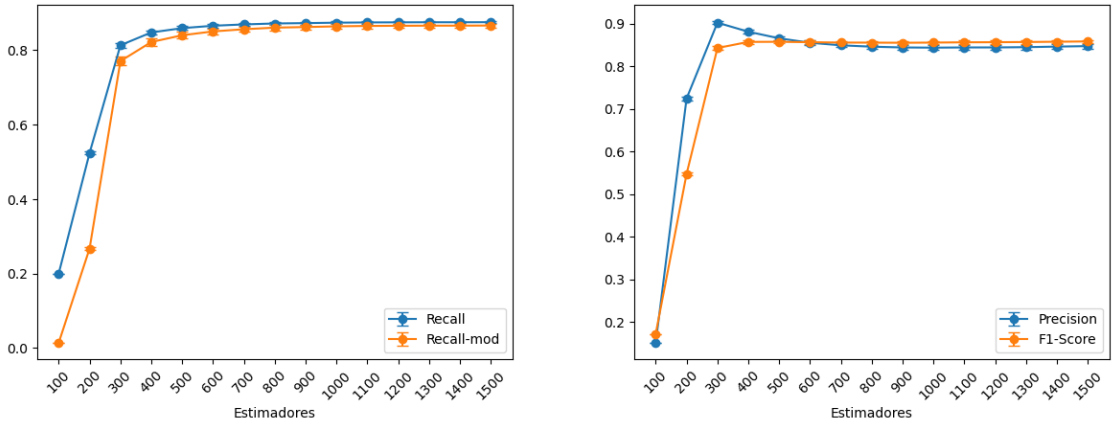
Por otro lado, es relevante considerar que la métrica *precision* presenta un valor promedio menor al observado cuando se utilizó *Gradient Boosting* sin modificar, lo que se debe al peor desempeño que se tiene para las clases mayoritarias por efecto del balance de datos. Por este motivo es de interés analizar el desempeño del modelo con un número distinto de estimadores o árboles, dado que observando las métricas promedio de *precision* y *recall* del modelo con 500 árboles, se observa que se tienen valores intermedios entre los obtenidos utilizando *Gradient Boosting* modificado y sin modificar.

Tabla 11: Desempeño en métricas de clasificación obtenidas para el clasificador *Gradient Boosting* modificado para curvas periódicas. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Métrica	$Estimadores$: 400	$Estimadores$: 500	$Estimadores$: 1400
$Precision$	$0{,}85 \pm 0{,}01$	$0{,}78 \pm 0{,}01$	$0{,}69 \pm 0{,}01$
$Recall$	$0{,}73 \pm 0{,}01$	$0{,}78 \pm 0{,}00$	$0{,}83 \pm 0{,}01$
$F1 - Score$	$0{,}78 \pm 0{,}01$	$0{,}78 \pm 0{,}01$	$0{,}74 \pm 0{,}01$
Min_recall	$0{,}42 \pm 0{,}02$	$0{,}52 \pm 0{,}01$	$0{,}66 \pm 0{,}02$
$Recall\_mod$	$0{,}53 \pm 0{,}02$	$0{,}61 \pm 0{,}01$	$0{,}74 \pm 0{,}01$

En la Figura 29 se puede notar que la relación entre las métricas del modelo y el número de estimadores tiene menor intensidad, puesto que se tiene que los valores promedios de las métricas convergen a un valor determinado tras un cierto número de estimadores.

Sin embargo, nuevamente se observa que tras cierto punto, la *precision* muestra una pendiente negativa, mientras que el *recall* aumenta al aumentar los árboles en el modelo.



(a) (b)

Figura 29: Métricas de clasificación de curvas estocásticas para el clasificador *Gradient Boosting* modificado en función del número de estimadores. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

Al analizar las matrices de confusión de distintos puntos, presentadas en la Figura 30, se observa el efecto de aumentar el número de estimadores del modelo, que resulta en una mejora en el desempeño de predicción de las clases minoritarias a costo de valores promedio menores para el caso de las clases mayoritarias, particularmente la clase QSO.

Sin embargo, en dichas matrices de confusión se observa que este efecto se presenta en menor medida para todas las clases del modelo a excepción de la clase Blazar, que muestra un porcentaje de clasificación correcta bajo al utilizar un modelo *Gradient Boosting* modificado con pocos estimadores. El hecho de que el efecto del *bootstrapping* balanceado sea notorio en una clase en particular muestra la baja variabilidad que se observa en las métricas en función del número de estimadores en la Figura 29, debido a que estas corresponden a *macro-métricas* por lo que promedian el valor de las métricas de desempeño para todas las clases, con lo que variaciones de baja intensidad en una clase en particular no conlleva un cambio significativo en el valor promedio de las métricas

En la Tabla 12, se observa una relación similar a los resultados obtenidos al entrenar un modelo *Gradient Boosting* sin modificar para el caso del clasificador periódico (Ver Tabla 11), teniéndose en el clasificador con 1500 estimadores un valor promedio de *precision* menor y *recall* mayor, mientras que el valor de *f1-score* resulta similar en ambos casos. La mayor diferencia se observa en el valor promedio de *min_recall*, que denota una mejor representación de la clase Blazar, la cual el modelo es difícil de clasificar.

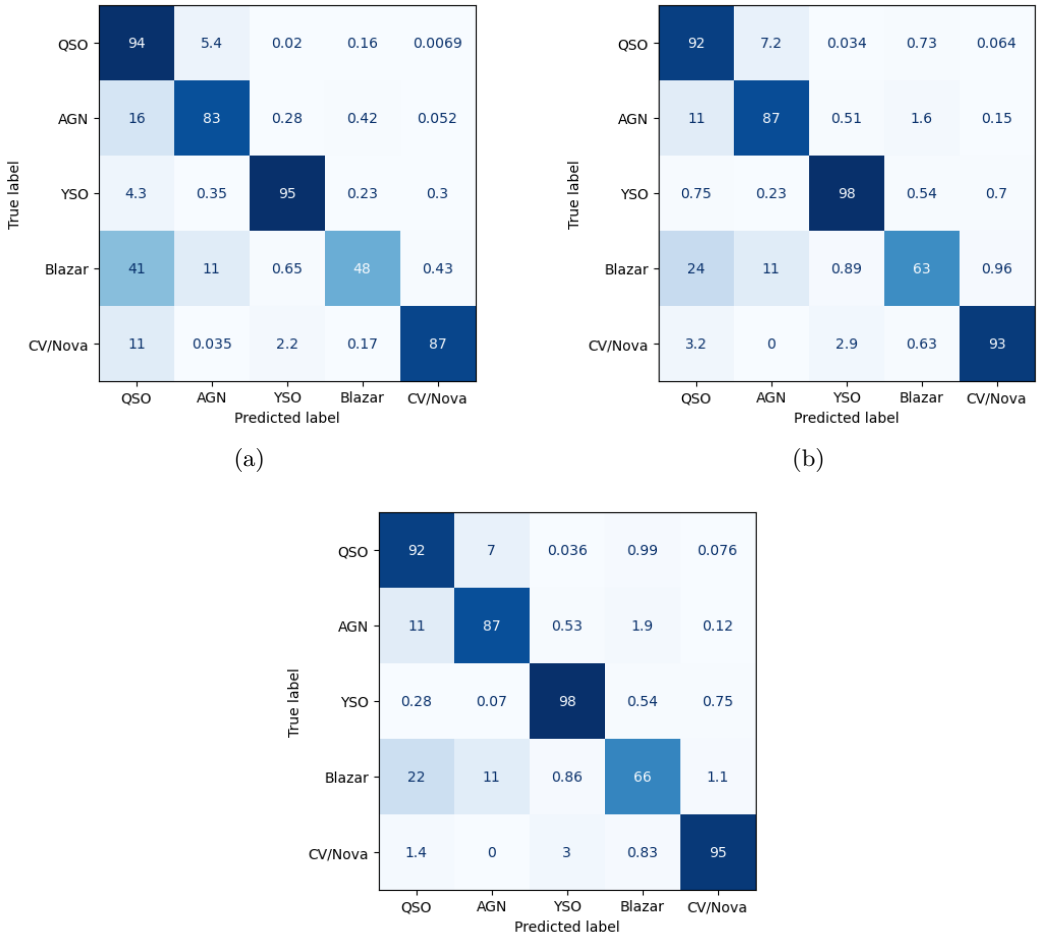
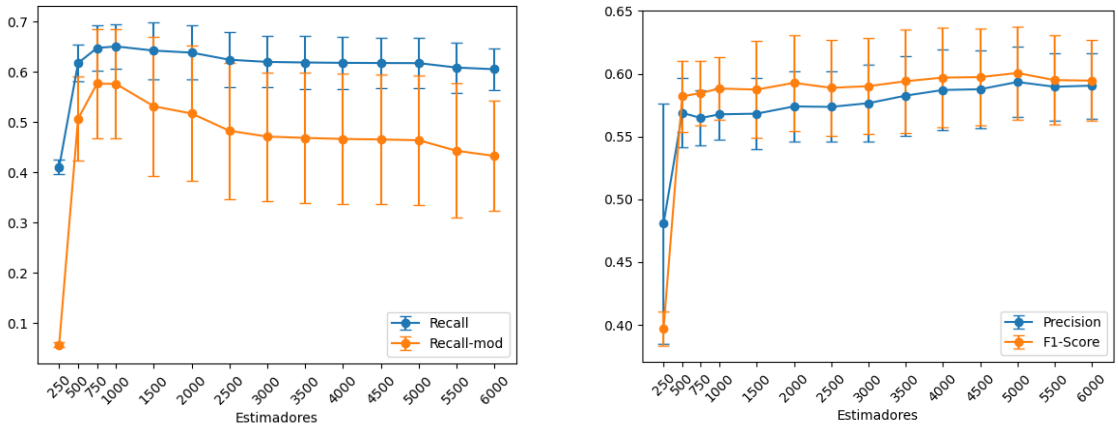


Figura 30: Matrices de confusión de clasificador *Gradient Boosting* modificado para curvas estocásticas. (a) 300 estimadores, (b) 600 estimadores, (c) 1500 estimadores.

Tabla 12: Desempeño en métricas de clasificación obtenidas para el clasificador *Gradient Boosting* modificado para curvas estocásticas. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Métrica	$Estimadores$: 300	$Estimadores$: 600	$Estimadores$: 1500
$Precision$	$0{,}90 \pm 0{,}00$	$0{,}86 \pm 0{,}00$	$0{,}85 \pm 0{,}01$
$Recall$	$0{,}81 \pm 0{,}01$	$0{,}87 \pm 0{,}00$	$0{,}88 \pm 0{,}00$
$F1 - Score$	$0{,}84 \pm 0{,}00$	$0{,}86 \pm 0{,}00$	$0{,}86 \pm 0{,}00$
Min_recall	$0{,}48 \pm 0{,}02$	$0{,}63 \pm 0{,}03$	$0{,}66 \pm 0{,}02$
$Recall\_mod$	$0{,}77 \pm 0{,}01$	$0{,}85 \pm 0{,}01$	$0{,}87 \pm 0{,}00$



(a) (b)

Figura 31: Métricas de clasificación de curvas transientes para el clasificador *Gradient Boosting* modificado en función del número de estimadores. (a) *Recall* (*macro*) y *Recall-mod*, (b) *Precision* y *F1-Score*.

Debido al bajo volumen de instancias (24) de la clase minoritaria SLSN dentro del grupo de curvas de objetos transientes, el subconjunto muestreado utilizado para entrenar cada árbol del modelo al usar *bootstrap* balanceado resulta muy pequeño en comparación a los casos anteriores, por lo que es necesario utilizar un número mayor de árboles en el modelo con el fin de asegurar una menor pérdida de información asociada al balance de datos realizado.

En la Figura 31 se observa un comportamiento diferente al de los clasificadores periódico y estocástico, dado que un aumento en el número de estimadores conlleva una disminución en el valor promedio de las métricas de *recall*, principalmente en el $recall_{mod}$. Por otro lado, un aumento en el número de árboles en el modelo muestra también un aumento en el valor promedio de la *precision* y del *f1-score*.

Cabe notar que, al igual que en otros casos en que se ha estudiado clasificadores de curvas transientes en el presente trabajo, se tienen valores relativamente altos de desviación estándar, especialmente en la métrica $recall_{mod}$, la que que según se ha observado presenta una volatilidad con mayor sensibilidad al desbalance de clases.

Este efecto se ejemplifica en las matrices de confusión presentadas en la Figura 32, en las que se observa que para el modelo con 1000 árboles se tiene una mejor clasificación de las clases SNIbc y SLSN en comparación con el modelo que se construye con 5000 estimadores, aunque se debe considerar que este último presenta un porcentaje de clasificación de curvas SNIa y SNII mayor, debido al sesgo de predicción que se observa en su desempeño.

Los resultados promedio presentados en la Tabla 13 evidencian que estos efectos de balance de desempeño de clasificación se interpretan con mayor notoriedad en las métricas *min_recall* y $recall_{mod}$, puesto que las bajas variaciones del promedio para las métricas *precision*, *recall* y *f1-score*, las cuales no superan un cambio de 0.03 en ningún caso, podrían llevar a interpretar de los resultados que ambos modelos presentan un comportamiento de predicción similar, caso que es

evidente descartar al observar las matrices de confusión en la Figura 32.

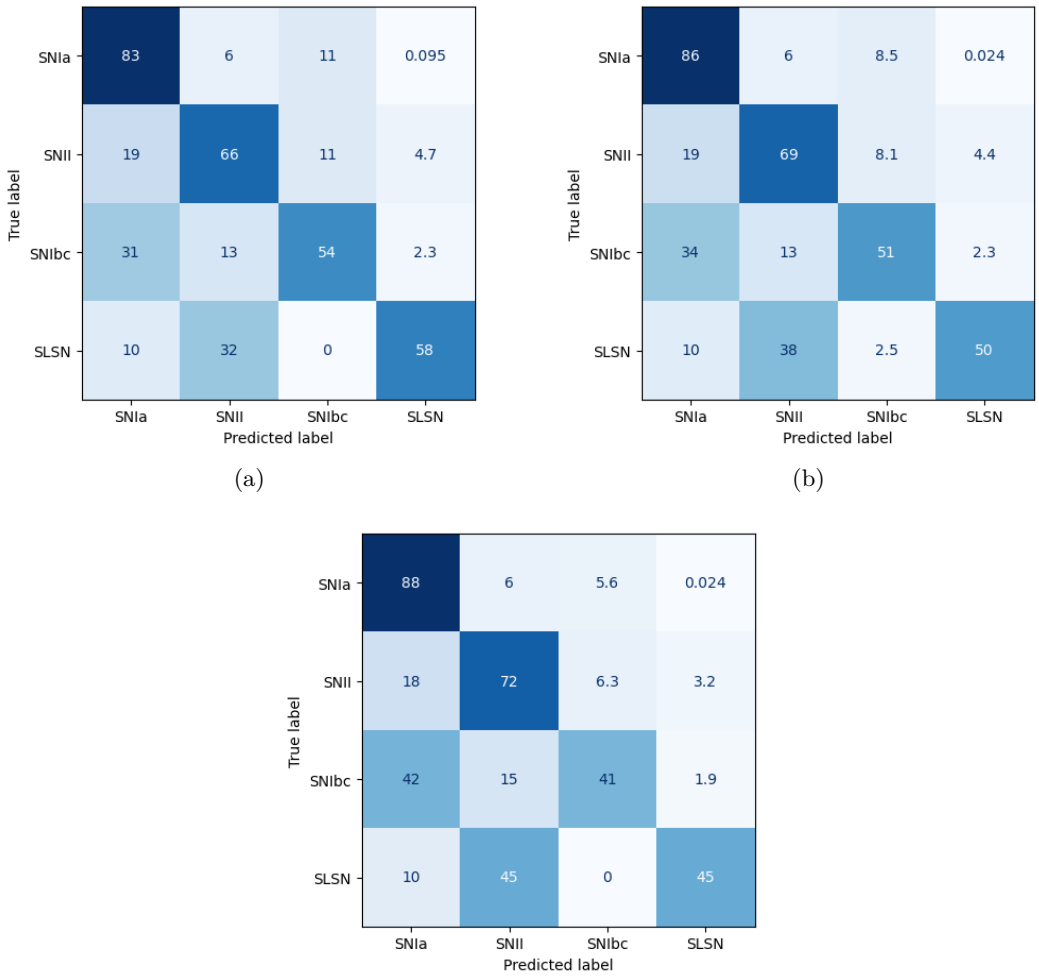


Figura 32: Matrices de confusión de clasificador *Gradient Boosting* modificado para curvas transientes. (a) 1000 estimadores, (b) 2000 estimadores, (c) 5000 estimadores.

Tabla 13: Desempeño en métricas de clasificación obtenidas para el clasificador *Gradient Boosting* modificado para curvas transientes. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Métrica	*Estimadores* : 1000	*Estimadores* : 2000	*Estimadores* : 5000
$Precision$	$0{,}57 \pm 0{,}02$	$0{,}57 \pm 0{,}03$	$0{,}59 \pm 0{,}03$
$Recall$	$0{,}65 \pm 0{,}04$	$0{,}64 \pm 0{,}05$	$0{,}62 \pm 0{,}05$
$F1 - Score$	$0{,}59 \pm 0{,}03$	$0{,}59 \pm 0{,}04$	$0{,}60 \pm 0{,}04$
Min_recall	$0{,}50 \pm 0{,}08$	$0{,}43 \pm 0{,}11$	$0{,}37 \pm 0{,}07$
$Recall\_mod$	$0{,}58 \pm 0{,}11$	$0{,}52 \pm 0{,}13$	$0{,}46 \pm 0{,}13$

4.4. Generación de curvas sintéticas usando modelo SPM

Generar un conjunto de curvas sintéticas a partir de las curvas del conjunto de entrenamiento permite realizar un balance a nivel de datos, buscando que el algoritmo se entrene con un número igual de instancias de cada clase con el fin de reducir el sesgo en la predicción. Debido al bajo número de datos que se tienen originalmente, realizar un sobremuestreo aleatorio no agregaría una cantidad relevante de información al modelo y un submuestreo aleatorio conllevaría una posible pérdida de información al no considerar un gran número de instancias del conjunto de entrenamiento.

Por otro lado, al balancear un conjunto de datos donde las clases minoritarias tienen un mayor volumen se esperaría que estas problemáticas disminuyesen, por lo que se entrenó un clasificador *XGBoost* de curvas transientes al incluir de distintas formas las curvas generadas sintéticamente en el conjunto de entrenamiento. Cabe destacar que se utiliza exclusivamente un clasificador *XGBoost* de curvas transientes debido a que la generación de curvas sintéticas a partir del modelo SPM solo se puede utilizar para generar curvas de objetos transientes.

Con el fin de tener un punto de comparación, en la Figura 33 se presentan los valores promedio de la matriz de confusión de un clasificador *XGBoost* de curvas transientes al realizar *Random Undersampling* sin repetición en el conjunto de entrenamiento, y los valores promedio para las métricas de clasificación. Cabe destacar que para las tablas presentadas se utilizará la notación de "Datos Originales" (D.O.) para referirse a las curvas presentes en las particiones de entrenamiento determinadas, y "Curvas Sintéticas" (C.S.) para referirse a las curvas sintéticas generadas a partir de dichos datos.

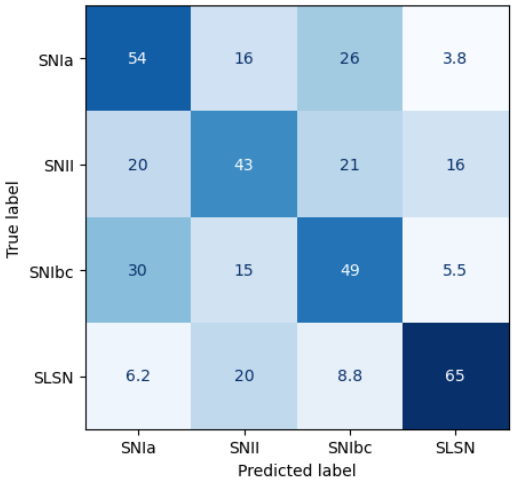


Figura 33: Matriz de confusión de clasificador *XGBoost* para curvas transientes al utilizar *Random UnderSampling*

Tabla 14: Desempeño en métricas de clasificación obtenidas para el clasificador *XGBoost* para curvas transientes al utilizar *Random UnderSampling*. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Clasificador	*Precision*	*Recall*	$F1 - Score$	Min_recall	$Recall\_mod$
D.O (RUS)	$0{,}38 \pm 0{,}03$	$0{,}53 \pm 0{,}07$	$0{,}37 \pm 0{,}046$	$0{,}34 \pm 0{,}055$	$0{,}61 \pm 0{,}15$

En la matriz de confusión de la Figura 33 que no se observa un sesgo hacia la clase SNIa como en otros casos, debido a que ya tiene el efecto de ser la clase mayoritaria del conjunto de datos. Sin embargo, la mencionada pérdida de información que se tiene al realizar un submuestreo con un volumen bajo de clases minoritarias se traduce en valores promedios bajos para la mayoría de las métricas presentadas en la Tabla 14, con excepción del $recall_{mod}$ que presenta un valor relativamente alto debido a que el clasificador tiene un alto porcentaje de predicción correcto para la clase minoritaria SLSN, y esta métrica es menos sensible a la mala clasificación que se observa en las clases mayoritarias del clasificador.

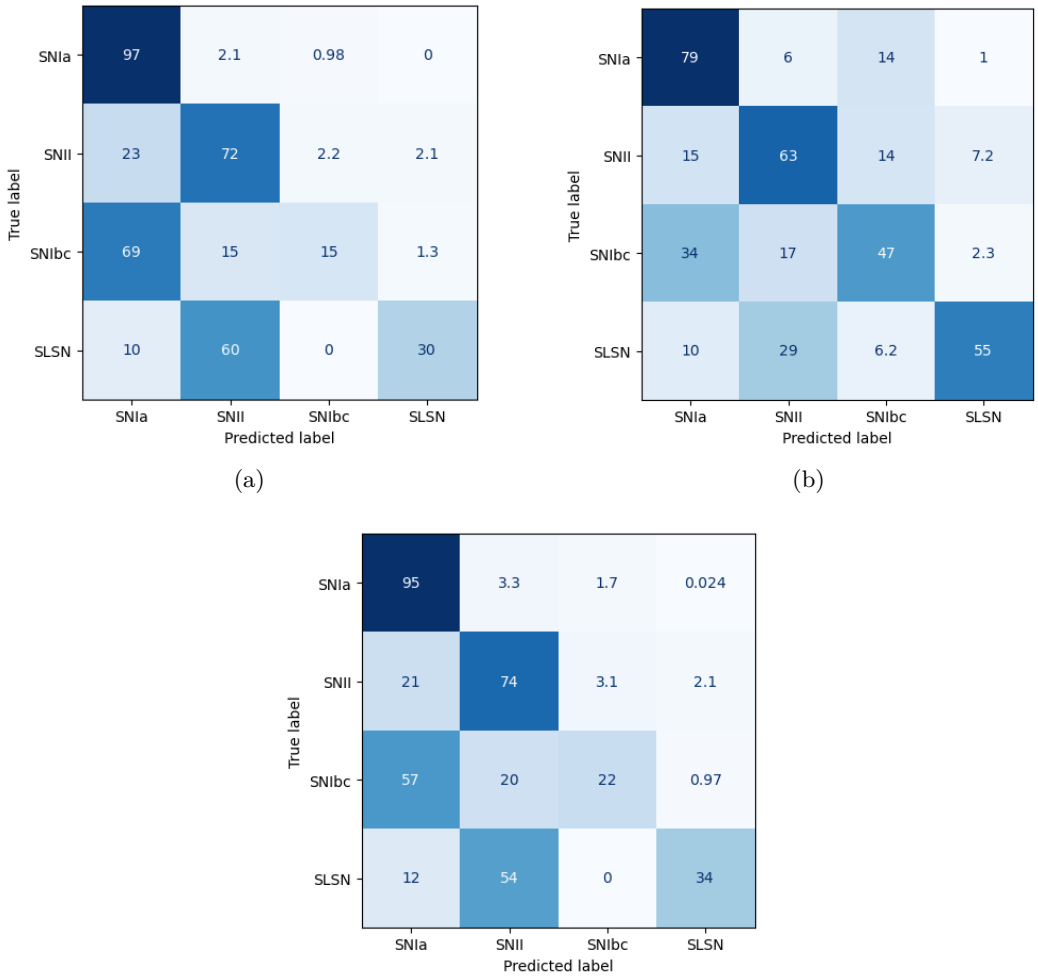
En vista de estos resultados, se busca entrenar el modelo agregando las curvas sintéticas al conjunto de entrenamiento, y para eso se formulan diferentes estrategias de balance:

- (a) D.O. + C.S. balanceadas: Se mantienen los datos originales del conjunto de entrenamiento y se agrega el mismo número de curvas sintéticas para cada clase, es decir, de todas las curvas sintéticas generadas se realiza un submuestreo para obtener un conjunto balanceado para agregarse al conjunto de entrenamiento. Con esto no se tiene un conjunto total balanceado debido a que el conjunto original no se modifica, y se evalúa el desempeño del clasificador con el fin de analizar si se obtiene una mejora significativa al contar con un mayor número de datos.
- (b) D.O. (RUS) + C.S. balanceadas: Se agregan las curvas sintéticas de forma similar al caso descrito anteriormente pero se realiza también un submuestreo a los datos de entrenamiento. Con esto se obtiene un conjunto de entrenamiento balanceado.
- (c) D.O. balanceados con C.S.: se busca balancear el conjunto de entrenamiento manteniendo la mayor cantidad de curvas empíricas posible. Esto se realiza al juntar todas las curvas originales y sintéticas de la clase menos representada, e igualar el número de instancias totales agregando curvas sintéticas de otras clases al conjunto de entrenamiento hasta que dicha clase esté representada por el mismo número de instancias que la clase minoritaria. Si alguna clase del conjunto presenta un mayor número de instancias que la suma de las que se tienen disponibles de la clase minoritaria al sumar las curvas originales con las sintéticas, se toma un número aleatorio de instancias de dicha clase del conjunto original.

Las valores promedio de las matrices de confusión para los tres casos descritos se presentan en la Figura 34. Se observa que solo al utilizar una suma de datos empíricos balanceados como de datos sintéticos balanceados (caso b) se logra evitar el sesgo hacia las clases mayoritaria.

Para el caso (a) de los datos originales en conjunto con curvas sintéticas balanceadas se tiene que el conjunto de entrenamiento utilizado resulta, aunque en menor medida, desbalanceado. Por otro lado, es relevante analizar que el caso (c) en que los datos originales se balancean haciendo

uso de curvas sintéticas mantiene el sesgo hacia las clases mayoritarias a pesar de que el modelo se entrena con un número igual de instancias para cada clase. Este sesgo puede atribuírsele al hecho de que, a pesar de que el conjunto de datos está balanceado, las clases SNIa y SNII están mayormente representadas por curvas empíricas, las que otorgan una mayor cantidad de información al clasificador durante el proceso de entrenamiento, mientras que para la clase SLSN, se tiene diversas instancias que fueron generadas a partir de datos presentes en el mismo conjunto de datos, por lo que la cantidad de información que aportan al clasificador es menor (Ver Figura 5).



(a) (b)

(c)

Figura 34: Matrices de confusión del clasificador *XGBoost* para curvas transientes al agregar curvas sintéticas y realizar tres formas de balance. (a) Datos originales + Curvas sintéticas balanceadas, (b) Datos originales balanceados + Curvas sintéticas balanceadas, (c) Datos originales balanceados con curvas sintéticas.

En la Tabla 15 se identifica que el caso "D.O (RUS) + C.S balanceadas" presenta un valor menor de *precision* y *f1-score* en comparación a los otros casos. Sin embargo, en la Tabla 16 las

métricas *min_recall* y $recall_{mod}$ presentan mayores valores promedios. Esto representa el mejor trabajo que hace dicha estrategia de balance al disminuir el sesgo y realizar una clasificación con mayor balance.

Cabe notar que incluso con el balance de datos hecho posible por la generación de curvas sintéticas, el clasificador *XGboost* de curvas transientes mantiene resultados promedio menores a *Balanced Random Forest* en métricas como *recall* o $recall_{mod}$ (Ver Tabla 1), lo que se debe al sesgo natural hacia las clases mayoritarias que resulta del algoritmo de entrenamiento del modelo *XGboost*.

Tabla 15: Desempeño en macro-métricas *Precision, Recall* y *F1 - Score* para el clasificador *XGBoost* para curvas transientes en conjunto con curvas sintéticas.

Caso	$Precision$	$Recall$	$F1 - Score$
D.O + C.S balanceadas	$0{,}66 \pm 0{,}047$	$0{,}54 \pm 0{,}044$	$0{,}57 \pm 0{,}044$
D.O (RUS) + C.S balanceadas	$0{,}51 \pm 0{,}025$	$0{,}61 \pm 0{,}05$	$0{,}52 \pm 0{,}03$
D.O balanceados con C.S	$0{,}65 \pm 0{,}055$	$0{,}56 \pm 0{,}04$	$0{,}59 \pm 0{,}043$

Tabla 16: Desempeño en métricas *min_recall* y $recall_{mod}$ obtenidas para el clasificador *XGBoost* para curvas transientes en conjunto con curvas sintéticas.

Caso	Min_Recall	$Recall\_Mod$
D.O + C.S balanceadas	$0{,}14 \pm 0{,}08$	$0{,}3 \pm 12$
D.O (RUS) + C.S balanceadas	$0{,}41 \pm 0{,}11$	$0{,}54 \pm 0{,}13$
D.O balanceados con C.S	$0{,}22 \pm 0{,}07$	$0{,}35 \pm 0{,}1$

4.5. Análisis estadístico de los resultados

Dado que el desempeño de los modelos de aprendizaje de máquinas depende de diversos factores aleatorios como la partición de entrenamiento y prueba o inicialización de parámetros en los procesos de entrenamiento, no se puede considerar un resultado obtenido en un contexto determinado como absoluto, motivo por el que se realiza la metodología de validación cruzada y se consideran tanto valores promedios como desviación estándar de los resultados obtenidos.

Sin embargo, para comparar el desempeño de diferentes modelos sometidos a distintos conjuntos de prueba mediante validación cruzada no es directo determinar el valor promedio mayor de alguna métrica, por lo que es de utilidad someter dichos resultados a un test estadístico que permita determinar con cierta certeza el comportamiento de dos poblaciones diferentes de valores.

Para esto se utiliza el test de *t-student*, que dado dos poblaciones de resultados, en este caso, los valores de las métricas obtenidas para cada partición de la validación cruzada, permite obtener la probabilidad de observar dichas poblaciones bajo una hipótesis nula, que en este caso corresponde a la hipótesis de que los valores de ambas poblaciones fueron obtenidas a partir de la misma distribución aleatoria.

En otras palabras, si se obtiene un valor *p-value* del test menor a un umbral α determinado (en este caso, $\alpha = 0{,}05$), se rechaza la hipótesis nula y se determina que ambas poblaciones son estadísticamente diferentes.

Sin embargo, para utilizar dicho test en un par de poblaciones de datos, es necesario determinar que los valores de dichas poblaciones se distribuyen de una forma que se asemeje en cierto grado a una distribución normal, o, en otras palabras, que exista una probabilidad suficiente de que los datos fueron obtenidos aleatoriamente de una distribución normal.

Con el objetivo de determinar si cierta población cumple esta condición se puede hacer uso del test de *Shapiro-Wilk*, que similarmente al test *t-student* mencionado, permite rechazar o aceptar una hipótesis nula según un valor *p-value* determinado. En este caso, la hipótesis corresponde a asumir que la distribución de los datos no tiene diferencia estadística con una distribución normal. Por esto, si se tiene un *p-value* menos a un umbral $\alpha = 0{,}05$ esta hipótesis se rechaza y se concluye que los datos presentan un grado de normalidad suficiente para ser evaluados según el test de *t-student*.

En la Tabla 17 se muestran los valores de *p-value* para los valores promedios de las distintas métricas estudiadas y su distribución en las 10 divisiones realizadas en la validación cruzada. Se obtienen los valores *p-values* para los distintos clasificadores de los modelos basados en *Balanced Random Forest*, *XGBoost* y *Gradient Boosting* modificado. Para los dos últimos se seleccionaron versiones de los modelos entrenado según lo observado en los resultados presentados:

- XGB:
  - Periódico: $max\_depth = 3$.
  - Estocástico: $max\_depth = 3$.
  - Transiente: $max\_depth = 1$, con métrica de validación min_recall.
- GB Mod:
  - Periódico: 1400 Estimadores.
  - Estocástico: 1400 Estimadores.
  - Transiente: 1000 Estimadores.

Tabla 17: Resultados p-value de tests de normalidad Shapiro-Wilk.

Clasificador	Modelo	$Precision$	$Recall$	$F1score$	Min_recall	$Recall_{mod}$
Periódico	XGB	**0.014**	0.19	**0.048**	0.46	0.06
	GB Mod	0.79	0.39	0.88	0.28	0.28
	BRF	0.75	0.87	0.22	0.2	0.052
Estocástico	XGB	0.27	0.46	0.58	0.45	0.16
	GB Mod	0.99	0.97	0.21	0.09	0.43
	BRF	0.26	0.72	0.44	0.28	0.72
Transiente	XGB	0.86	0.27	0.61	0.31	0.17
	GB Mod	0.74	0.58	0.98	0.36	0.36
	BRF	0.89	0.79	0.085	0.073	0.3
Completo	XGB	0.22	0.91	0.58	0.1	0.31
	BRF	0.3	0.12	0.44	0.24	0.78

Se puede observar que para todos los casos evaluados, se obtiene un *p-value* superior al umbral 0,05 para todas las métricas, siendo la única excepción el clasificador *XGBoost* para curvas periódicas. Para todos los demás casos no se tiene certeza suficiente para rechazar la hipótesis nula y se concluye que poseen una distribución cercana a una distribución Gaussiana y pueden ser sometidos al test *t-student*.

Por otro lado, no se puede asumir normalidad en la distribución de resultados del clasificador periódico con profundidad máxima 3. Sin embargo, los resultados de este test cambian dependiendo de la profundidad máxima que se utilice para obtener los datos de la validación. En la Tabla 18 se muestran los *p-values* obtenidos al realizar el test de normalidad a los valores promedio de las métricas obtenidas cuando se entrenaron clasificadores con valor de $max\_depth = 7$. Se puede notar que los valores aumentan de forma significativa, evidenciando que en este caso si se tiene una distribución cercana a la distribución normal que puede ser usada en un test *t-student*. De esta observación se puede inferir que se puede esperar que un clasificador *XGBoost* de curvas periódicas con un bajo valor de max_depth tenga un desempeño variable y posiblemente inválido para análisis estadísticos.

Tabla 18: *p-values* de tests de normalidad Shapiro-Wilk para clasificador periódico XGB con $max\_depth = 7$.

Clasificador	Modelo	$Precision$	$Recall$	$F1score$	Min_recall	$Recall_{mod}$
Periódico	XGB	0.12	0.19	0.75	0.09	0.58

Utilizando esta versión para el clasificador *XGBoost* periódico se obtienen los resultados del test *t-student* al comparar el clasificador *XGBoost* con *Balanced Random Forest*, presentados en la Tabla 19.

Tabla 19: p-values de tests de *t-student* entre resultados obtenidos con *XGBoost* y Balanced Random Forest.

Clasificador	$Precision$	$Recall$	$F1score$	Min_recall	$Recall_{mod}$
Periódico	8.17e-13	2.6e-10	4.48e-9	5.26e-7	2.99e-16
Estocástico	2.3e-19	4.34e-9	2.7e-18	**0.53**	0.007
Transiente	8.95e-22	2.5e-12	1.32e-20	3.74e-8	6.3e-4
Completo	3.7e-15	3.4e-18	6.3e-27	7.3e-9	2.6e-5

Se puede observar que para la mayoría de las métricas se obtienen valores menores al umbral definido de 0,05 por lo que se determina que los resultados son estadísticamente distintos, y por ende, comparables. Sin embargo, se observa que para el caso específico del clasificador estocástico se tiene un valor alto de la métrica min_recall, lo que indica que ambos clasificadores tienen una distribución similar de esta métrica y no se puede determinar una diferencia estadística entre ellos usando el valor de min_recall promedio. Con esto y dado la variabilidad observada de dicha métrica a lo largo del presente trabajo, se concluye que resulta de mayor utilidad al ser usada con fines interpretativos y no para comparar el desempeño de dos modelos diferentes.

Para el caso de *Gradient Boosting* modificado y *Balanced Random Forest* los resultados del test *t-student* se presentan en la Tabla 20, en la que se observan múltiples métricas con valores altos en sus *p-values*, es decir, que presentan similitud en la distribución de sus valores. Particularmente se observa que para los tres clasificadores evaluados la métrica min_recall presenta valores altos y superiores al umbral definido, lo que refuerza la noción de que dicha métrica no corresponde a una métrica útil para comparar modelos diferentes.

Tabla 20: p-values de tests de *t-student* entre resultados obtenidos con Gradient Boosting modificado y Balanced Random Forest.

Clasificador	$Precision$	$Recall$	$F1score$	Min_recall	$Recall_{mod}$
Periódico	5.99e-13	1.99e-9	3.22e-12	**0.57**	5.9e-16
Estocástico	2.3e-9	3.0e-8	3.25e-18	**0.84**	**0.56**
Transiente	8.95e-22	**0.71**	2.24e-7	**0.97**	6.7e-3

5. Resultados con datos actualizados de ALeRCE

Con el fin de evaluar el desempeño del modelo con datos a los que no se ha tenido acceso durante el desarrollo del presente trabajo, se consideran datos actualizados de la base de datos de ALeRCE hasta Diciembre de 2022, verificando que en dichos datos actualizados no se encuentren datos utilizados anteriormente durante el presente trabajo.

Este nuevo conjunto de datos de prueba consta con los siguientes volúmenes de instancias por clase:

- Periódicas:
  - LPV: 35145
  - E: 32345
  - RRL: 12591
  - Periodic-Other: 784
  - DSCT: 679
  - CEP: 393
- Estocásticas:
  - QSO: 37329
  - YSO: 6176
  - AGN: 3733
  - Blazar: 701
  - CV/Nova: 465
- Transientes:
  - SNIa: 1490
  - SNII: 287
  - SNIbc: 81
  - SLSN: 33

Según los resultados observados durante el desarrollo del presente trabajo, se realiza la predicción de los datos actualizados utilizando *XGboost* con *cost-sensitive learning* para los clasificadores jerárquico, periódico y estocástico, utilizando profundidades máximas de 4, 3, y 3, respectivamente. Para el caso del clasificador transiente, se utiliza un modelo *XGBoost* con generación de curvas sintéticas y datos balanceados según lo descrito en el capítulo 4.4.

Para cada clasificador se utiliza una combinación de hiperparámetros que durante el desarrollo del trabajo resultó en un alto desempeño del clasificador. Estas combinaciones se muestran en la Tabla 21.

Tabla 21: Parámetros seleccionados para cada clasificador *XGBoost* para prueba con datos nuevos de ALeRCE.

Parámetro	Jerárquico	Periódico	Estocástico	Transiente
learning_rate	0,125	0,025	0,025	0,1
gamma	0,2	1,4	1,5	1,3
min_child_weight	7	1	6	4
subsample	0,85	0,9	1	0,6
colsample_bytree	0,8	0,8	0,95	0,9

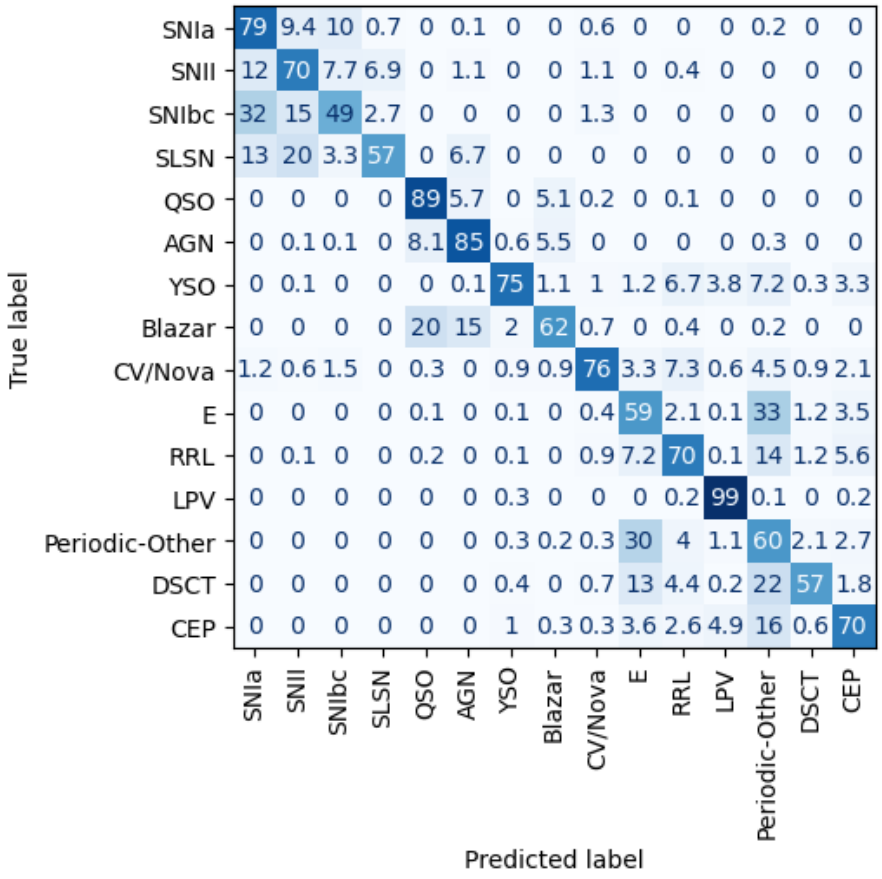


Figura 35: Matriz de confusión de clasificador *XGBoost* completo utilizado con datos actualizados de ALeRCE

La Figura 35 presenta la matriz de confusión del clasificador propuesto al predecir la clase de datos nuevos. Se puede observar que existe un mayor grado de confusión entre clases en comparación al caso mostrado en el capítulo 4.2.6. Sin embargo, este aumento en la confusión entre clases se observa también al clasificar los datos nuevos utilizando el modelo *Balanced Random Forest* con la configuración de ALeRCE, según se observa en la Figura 36.

Con esto, se puede inferir que en el conjunto de datos actualizado se presenta un desplazamiento

en la distribución de los objetos representados, principalmente dentro del subconjunto de curvas periódicas, según se observa en las Figuras 37 y 38. A esto se suma la incertidumbre que se tiene al obtener resultados para un problema de clasificación específico y no mediante una validación cruzada, como se realiza al evaluar el desempeño de un clasificador con datos que se tienen disponibles para su entrenamiento y evaluación, dado que se observan resultados únicos en lugar de valores promedios con sus respectivas desviaciones estándar.

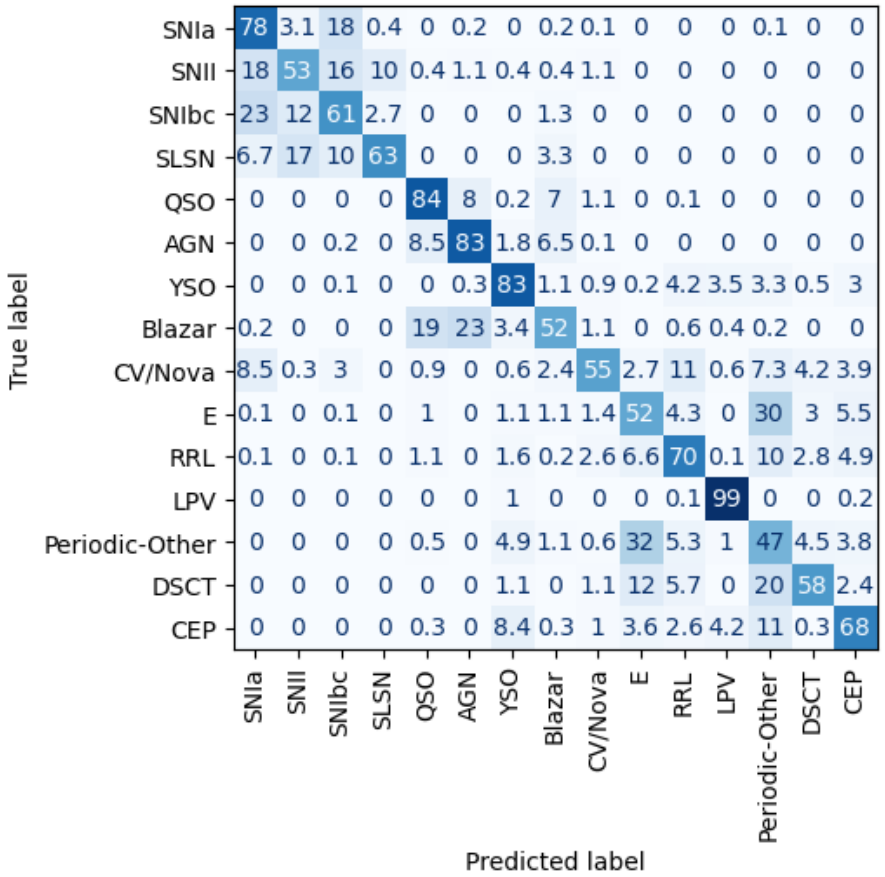


Figura 36: Matriz de confusión de clasificador *Balanced Random Forest* completo utilizado con datos actualizados de ALeRCE

Tabla 22: Desempeño en métricas de clasificación obtenidas para el *XGBoost* y *BRF* completos usando datos actualizados. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Clasificador	*Precision*	*Recall*	$F1 - Score$	*Min_recall*	*Recall_mod*
XGBoost	0,57	0,71	0,58	0,49	0,58
BRF	0,51	0,67	0,52	0,47	0,63

En la Tabla 22 se observa *XGBoost* obtuvo un mejor desempeño que *Balanced Random Forest* en todas las métricas a excepción de $recall_{mod}$, lo que se debe a que en este caso dicha métrica prioriza

la clasificación de las curvas transientes, curvas para las que *Balanced Random Forest* realiza una mejor predicción.

Por otro lado, de las métricas presentadas en las Tablas 23 y 24 se obtiene que al predecir los datos actualizados, *XGBoost* con diferentes técnicas de balance de datos obtuvo un desempeño superior a *Balanced Random Forest* en todas las métricas en los clasificadores periódico, estocástico y jerárquico, y obtuvo resultados similares en el clasificador transiente, siendo superado por *Balanced Random Forest* en las métricas de *recall* y $recall_{mod}$.

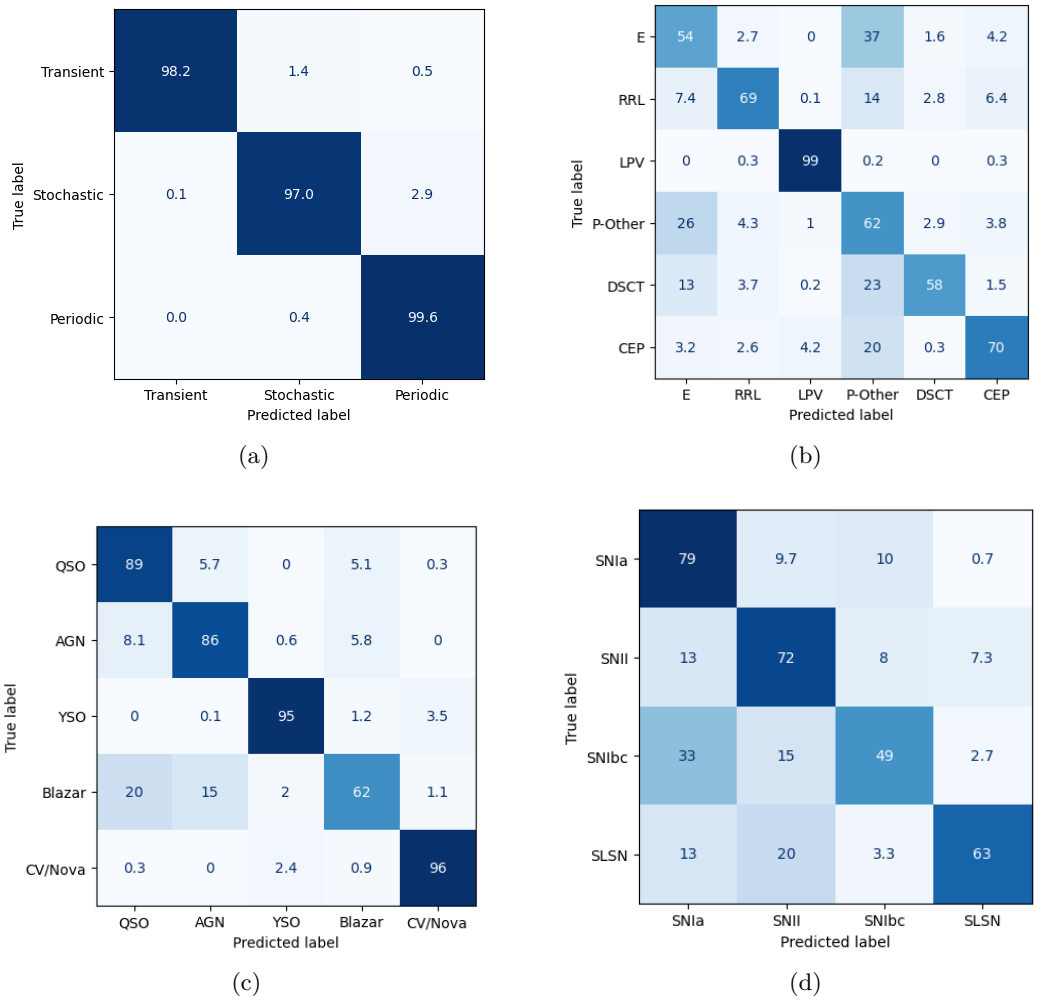
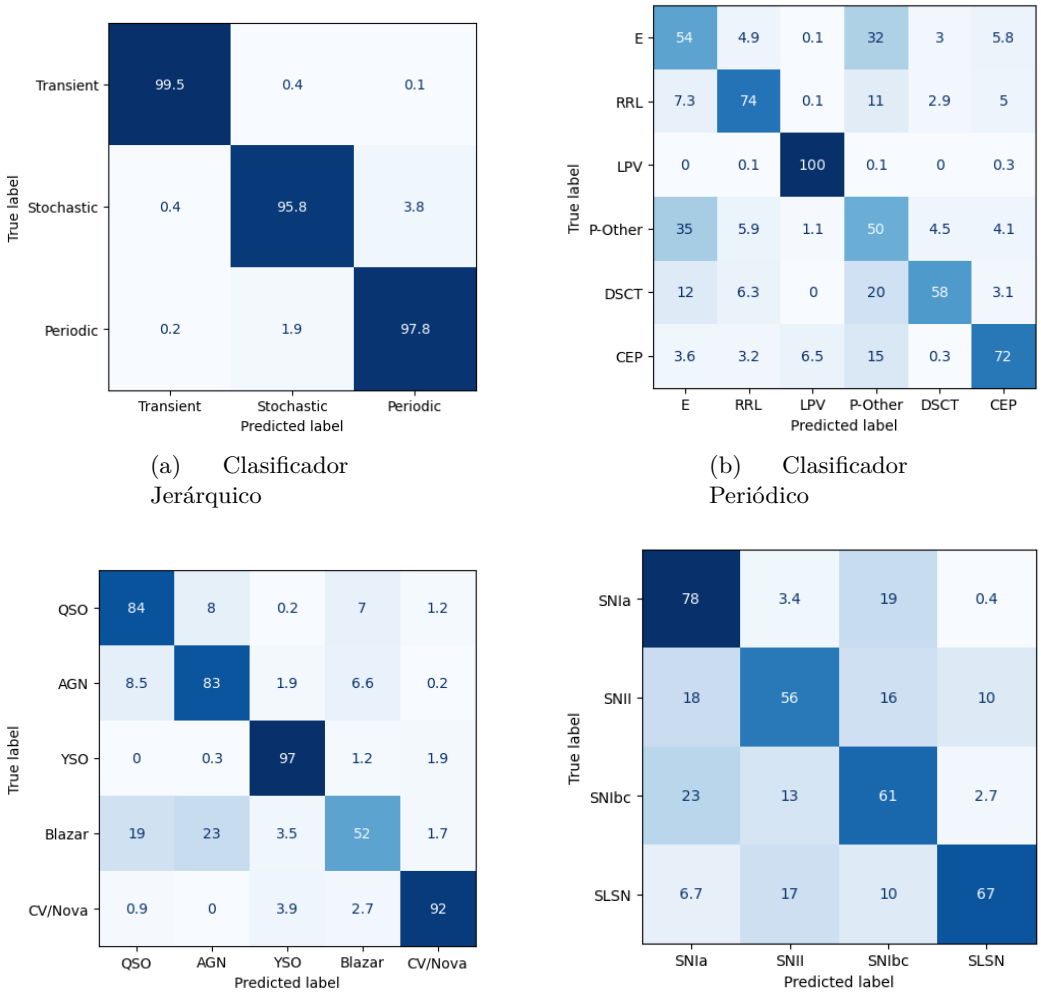


Figura 37: Matrices de confusión de los clasificadores *XGBoost* utilizados con datos actualizados de ALeRCE. (a) Clasificador Jerárquico, (b) Clasificador Periódico, (c) Clasificador Estocástico, (d) Clasificador Transiente.



(a) Clasificador Jerárquico

(b) Clasificador Periódico

(c) Clasificador Estocástico

(d) Clasificador Transiente

Figura 38: Matrices de confusión de los clasificadores *Balanced Random Forest* utilizados con datos actualizados de ALeRCE. (a) Clasificador Jerárquico, (b) Clasificador Periódico, (c) Clasificador Estocástico, (d) Clasificador Transiente.

Tabla 23: Desempeño en métricas de clasificación obtenidas para los cuatro clasificadores basados en *XGBoost* con datos actualizados. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Clasificador	*Precision*	*Recall*	$F1 - Score$	*Min_recall*	*Recall_mod*
Jerárquico	0,99	0,98	0,98	0,97	0,98
Periódico	0,55	0,69	0,52	0,54	0,65
Estocástico	0,67	0,86	0,73	0,61	0,84
Transiente	0,52	0,66	0,56	0,49	0,61

Tabla 24: Desempeño en métricas de clasificación obtenidas para los cuatro clasificadores basados en *Balanced Random Forest* con datos actualizados. *Precision, Recall, y F1-Score* corresponden a métricas *macro*.

Clasificador	$Precision$	$Recall$	$F1-Score$	Min_recall	$Recall\_mod$
Jerárquico	0,93	0,98	0,95	0,96	0,99
Periódico	0,52	0,68	0,51	0,5	0,63
Estocástico	0,6	0,82	0,65	0,53	0,81
Transiente	0,53	0,65	0,54	0,56	0,65

6. Conclusiones

Considerando el estado del arte sobre algoritmos de aprendizaje de máquinas, problemas de clasificación con clases desbalanceadas, uso de modelos *XGBoost*, y clasificación de datos astronómicos de diferentes clases, se pudieron identificar diferentes metodologías de balance de datos para profundizar un estudio de su optimización con el fin de obtener un desempeño de clasificación más balanceado en un modelo *XGBoost* que al utilizarse en sus configuraciones predeterminadas. Todo lo anterior aplicado a la clasificación de curvas de luz de diferentes eventos astronómicos.

En [5] se pudo identificar que entre las diferentes metodologías de balance a nivel de datos, *XGBoost* presentaba mejor desempeño al utilizarse en conjunto con una técnica de balance a nivel de algoritmo, correspondiente a la estrategia de *cost-sensitive learning*, que permite al modelo priorizar la clasificación de clases determinadas durante su proceso de entrenamiento, específicamente para este caso las clases menos representadas en el conjunto de datos.

Sin embargo, en dicho trabajo se observó que utilizando esta estrategia de balance no se resolvía completamente el problema del sesgo del clasificador hacia las clases mayoritarias, reflejándose en que a pesar de que se obtuvieron valores altos de métricas como la *precision* y el *f1-score*, los valores promedio de *min_recall* se mantenían relativamente bajos, lo que se puede notar también en las matrices de confusión presentadas. Esto sirvió de motivación para hacer énfasis en el presente trabajo a estudiar diferentes métricas de desempeño con el fin de tener una mejor interpretación sobre el desempeño general de los modelos entrenados.

Debido al bajo número de instancias presentes en el conjunto de datos de las clases minoritarias de curvas transientes, al clasificar estas curvas toma más relevancia el problema del sesgo del clasificador, lo que se pudo observar en los resultados obtenidos con diferentes metodologías durante el desarrollo de este trabajo. Por este motivo, se buscó evaluar la metodología de generación de curvas sintéticas de objetos transientes basada en parámetros SPM [7] aplicada en conjunto con un modelo *XGBoost*, dada la mejora en el desempeño de clasificación que se documentó en dicho trabajo al usarse esta estrategia de aumento de datos en conjunto a un modelo *Balanced Random Forest* y diferentes tipos de clasificadores basados en atención.

Otra metodología que se evaluó en un modelo *XGBoost* corresponde a la estrategia utilizada en *Balanced Random Forest* de *bootstrapping* balanceado. Dado que en el alcance de la revisión bibliográfica no se identificó un estudio o implementación existente de dicha metodología aplicada a un algoritmo de *XGBoost* u otro algoritmo de *Boosting*, se implementó de forma manual al algoritmo de construcción de la clase *Gradient Boosting* de la librería *scikit-learn*. Esto se hizo debido a la complejidad mayor del algoritmo de *XGBoost* en comparación al algoritmo de *Gradient Boosting* tradicional, por lo que una implementación directa a un modelo *XGBoost* se deja como trabajo futuro.

Los distintos clasificadores entrenados durante el desarrollo del presente trabajo se evaluaron usando diferentes métricas de clasificación, con el fin de evaluar las diferencias en sus tendencias de predicción. A pesar de que la métrica *macro-recall* interpreta el desempeño de un modelo determinado a lo largo de todas las clases, ésta trata a todas las clases con el mismo peso, por lo que una clasificación peor en las clases minoritarias no suele verse reflejada de forma notoria, si se

cuenta con una clasificación eficaz de las clases mayoritarias. Debido a esto, se propuso la métrica de *recall* modificado o $recall_{mod}$, que pondera el valor del *recall* para cada clase por un peso inversamente proporcional al porcentaje de instancias de dicha clase dentro del conjunto de datos. De esta forma, analizando el valor de $recall_{mod}$, y especialmente al compararlo con el valor del *recall* correspondiente, se puede tener una noción más clara sobre el sesgo del clasificador hacia las clases mayoritarias.

A partir de los resultados obtenidos con *XGBoost* utilizando *cost-sensitive learning*, se analizó la relación entre la profundidad máxima de los árboles del modelo y su desempeño de clasificación, observándose que valores más bajos favorecen a la predicción de clases minoritarias mientras que una profundidad mayor aumenta el sesgo del clasificador. Esto se observa principalmente en los resultados observados de los clasificadores periódico y estocástico, mientras que los resultados del clasificador transiente, tanto en el caso de *XGBoost* como en el posteriormente estudiado *Gradient Boosting* modificado, muestran que para dichas curvas se tiene dificultad para establecer una relación directa entre los parámetros del modelo y su desempeño de clasificación, observándose así en los clasificadores transientes los mayores valores de desviación estándar entre todos los clasificadores analizados.

Con esta revisión, se pudo observar que los desempeños promedio de los clasificadores estocástico y periódico con profundidad 3 superaron en la mayoría de las métricas de clasificación a *Balanced Random Forest*, con excepción de la métrica propuesta de $recall_{mod}$, en la que *BRF* presenta generalmente un buen desempeño evidenciando su bajo sesgo y capacidad de clasificar correctamente las clases minoritarias. Cabe señalar que como se identificó en el test estadístico realizado posteriormente, los resultados obtenidos por *XGBoost* con profundidad 3 no presentan suficiente normalidad para ser comparados con los resultados obtenidos por *BRF*. Sin embargo, se determinó que si se obtiene esta normalidad cuando se utiliza como profundidad máxima el valor 7, y analizando los resultados promedio obtenidos con dicha profundidad se observa que se tiene una mejora notoria en valores de métricas como *precision* o *f1-score*, pero una caída en la métrica $recall_{mod}$, permitiendo identificar que *XGBoost* en este caso resulta más eficiente al momento de predecir clases mayoritarias.

Por otro lado, el clasificador basado en *XGBoost* que presenta la ventaja mas considerable sobre *BRF* corresponde al clasificador jerárquico, del que se obtiene una clasificación cercana a una clasificación perfecta para todas las clases. Dado que se determinó en el test estadístico que el espacio de resultados de dicho clasificador es estadísticamente diferente al de los obtenidos por *BRF*, se concluye que el clasificador jerárquico basado en *XGBoost* supera al basado en *BRF* al trabajar con el conjunto de datos utilizados. Este efecto se observa también al clasificar los datos actualizados de ALeRCE, sin embargo, dado a que corresponde a un caso particular y no a valores promediados, no se pueden realizar afirmaciones con respecto a la mejoría de un modelo sobre el otro. A pesar de esto, el resultado con los datos actualizados contribuye a la interpretación de los resultados sobre el desempeño superior de *XGBoost* sobre *BRF* al clasificar las curvas según su clase jerárquica.

Al comparar los resultados obtenidos con los modelos de *Gradient Boosting* original con los obtenidos al implementar *bootstrapping* balanceado en su algoritmo, se pudo analizar el efecto que tiene el balance de clases a nivel de datos para un algoritmo de construcción de árboles de decisión secuencial. Se observó en los resultados promedio para los tres clasificadores de nivel

inferior que el *bootstrapping* implementado reduce altamente el sesgo del modelo hacia las clases mayoritarias, especialmente al clasificar curvas periódicas y estocásticas. Con esto, se concluye que esta modificación al algoritmo de *Gradient Boosting* resulta una herramienta útil para obtener un desempeño balanceado en la clasificación.

Para comparar los resultados promedio obtenidos con *Gradient Boosting* con los resultados que se obtienen al utilizar *BRF*, es importante tener en consideración los resultados del test *t-student* de los que se determinó que las métricas min_recall y $recall_{mod}$ no presentan la distribución necesaria para ser comparadas entre estos modelos, y para el caso del clasificador transiente, tampoco se puede comparar de forma correcta los resultados obtenidos para el *recall*. A pesar de esto, analizando los valores obtenidos de *precision*, *recall* y *f1-score*, se observa que para los tres clasificadores se obtuvieron valores promedio superiores a los obtenidos por sus contraparte basada en *BRF*, sin embargo, dado que no se tiene una base para comparar las métricas de min_recall y $recall_{mod}$, resulta difícil concluir la superioridad de un modelo sobre otro en determinados aspectos, dado que son estas métricas en las que *BRF* presenta valores promedio mucho mayores, evidenciando su menor sesgo en la clasificación.

Por otra parte, mediante el uso de curvas generadas sintéticamente a partir del modelo paramétrico SPM se pudo obtener una clasificación más balanceada entre las clases de curvas transientes en comparación al desempeño observado en los clasificadores transientes utilizando *XGBoost* o *Gradient Boosting* modificado, lo que se observa al comparar los valores de las métricas min_recall y $recall_{mod}$. Sin embargo, los resultados obtenidos siguen siendo menores a los obtenidos con *Balanced Random Forest*, que presenta un mejor desempeño al clasificar un conjunto de datos balanceado.

El uso de datos sintéticos permitió también analizar el efecto que tiene la información que agrega cada instancia del conjunto, ejemplificado en que al utilizar dos conjuntos con la misma cantidad de clases, el caso en el que los datos de las clases mayoritarias aportan mayor información al algoritmo por hecho de estar constituidos en mayor cantidad por datos empíricos en lugar de sintéticos mantiene el sesgo visible en los casos en que no se tiene un balance en los datos.

Al realizar el análisis estadístico de las poblaciones de los datos obtenidos, se observó que modelos de *XGBoost* con valores menores de profundidad máxima son más propensos a presentar alta variabilidad en sus datos, lo que se identifica con sus bajos resultados en el test de normalidad realizado. Por otro lado, se pudo concluir que la mayor parte de los datos si presentan distribuciones que se asemejan a una distribución Gaussiana, por lo que se pudo realizar un test de *t-student* para analizar la similitud estadística de los modelos estudiados versus *BRF*. Este test permitió concluir que la métrica min_recall no suele ser apta para comparar modelos diferentes, y tiene mayor utilidad al ser utilizada como herramienta interpretativa. Esto queda en mayor evidencia en el caso de los *p-values* obtenidos en el test para el clasificador *Gradient Boosting* modificado, caso en el que una comparación de su desempeño con el del modelo *BRF* resulta menos directa.

Finalmente, se entrenó un clasificador *XGBoost* completo configurado según los resultados observados durante el desarrollo del trabajo y se utilizó para realizar una predicción a datos actualizados de ALeRCE, a los cuales no se tuvo acceso anteriormente. Esto con el fin de evaluar el desempeño del modelo con datos que no habían sido utilizados en ninguna forma para su análisis anterior. Con motivo de comparación, también se realizó una predicción de estos datos haciendo uso de un modelo *BRF* configurado con los parámetros utilizados por ALeRCE. Para ambos clasificadores

se observó un mayor grado de error al clasificar estos datos, especialmente en las curvas de clase periódica. Esto lleva a inferir que los objetos específicos presentes en el nuevo conjunto de datos tienen generalmente características diferentes a las del conjunto de datos usados anteriormente.

Para finalizar, en vista de los resultados obtenidos tanto durante el desarrollo del trabajo como durante la prueba de desempeño utilizando datos actualizados, se puede concluir que un modelo *XGBoost* configurado de forma óptima resulta una alternativa interesante a *Balanced Random Forest*, particularmente cuando se afronta de forma correcta el problema del sesgo presente originalmente en la clasificación hecha por dicho algoritmo.

6.1. Trabajo Futuro

Dado que se pudo observar que la implementación de *bootstrapping* balanceado reduce el sesgo del clasificador *Gradient Boosting*, se considera de interés un estudio en mayor profundidad sobre la implementación de esta estrategia de balance en *XGBoost* debido a que dicho clasificador presentó un alto desempeño de clasificación pero con un sesgo significativo hacia las clases mayoritarias.

También se tiene como trabajo futuro el estudio de estrategias de balance de datos adicionales a las analizadas en el presente trabajo, tales como el uso de *Auto-Encoders* y *Generative Adversarial Networks* [34] para la generación de datos sintéticos.

Bibliografía

[1] Förster, F. et al. “The Automatic Learning for the Rapid Classification of Events (ALeRCE) Alert Broker,” The Astronomical Journal, vol. 161, no. 5, 2021.

[2] Sanchez-Saéz, P. et al. “Alert Classification for the ALeRCE Broker System: The Light Curve Classifier” The Astronomical Journal, vol. 161, no. 3, 2021.

[3] Carrasco-Davis, R. et al. “Alert Classification for the ALeRCE Broker System: The Real-time Stamp Classifier” The Astronomical Journal, vol. 162, no. 6, 2021.

[4] Chen, C. and Liaw, A., Breiman, L., “Using Random Forest to Learn Imbalanced Data,” University of California, Berkeley, 2004.

[5] Molina, J., “Clasificador de curvas de luz utilizando modelo XGBoost y técnicas de balance de datos”, Memoria de Ingeniería Civil Eléctrica, Universidad de Chile, 2022.

[6] Catelan, M., Smith, H., “Pulsating Stars” Wiley-VCH, 2015.

[7] Pimentel, Ó., Estévez, P., Förster, F., “Deep Attention-based Supernovae Classification of Multiband Light Curves”, The Astronomical Journal, vol. 165, no. 1, 2022.

[8] Rokach, L., Maimon, O., “Decision Trees”, In: Maimon, O., Rokach, L., Data Mining and Knowledge Discovery Handbook. Springer, Boston, 2005

[9] Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., “Classification and regression trees”, Monterey, CA: Wadsworth, Brooks/Cole Advanced Books, Software, 1984

[10] Livingston, F., “Implementation of Breiman’s Random Forest Machine Learning Algorithm”, Machine Learning. 2005

[11] Friedman, J., “Stochastic Gradient Boosting”, Computational Statistics & Data Analysis, vol. 38, no. 4, 2002.

[12] Chen, T., Guestrin, C., “XGBoost: A Scalable Tree Boosting System” University of Washington, 2016.

[13] XGBoost. [Online]. Available: https://www.kaggle.com/code/dansbecker/xgboost/notebook. [Accessed: 02-Nov-2022]

[14] Grandini, M., Bagli, E., Visani, G., “Metrics for Multi-Class classification: an Overview”, 2020. arXiv:2008.05756.

[15] Muschelli, J., “Roc and auc with a binary predictor: a potentially misleading metric”. Journal of Classification, pages 1–13, 2019.

[16] Opitz, J., Burst, S., ’Macro F1 and Macro F1’, 2019, arXiv:1911.03347v3.

[17] Yunqian, M., Haibo, H., “Foundations of imbalanced learning” in *Imbalanced Learning: Foundations, Algorithms, and Applications*, The Institute of Electrical and Electronics Engineers, 2013

[18] Zhang, J., Mani, I. “KNN Approach to Unbalanced Data distributions: A Case Study Involving Information Extraction” Proc. Int’l Conf. Machine Learning (ICML ’2003), Workshop Learning from Imbalanced Data Sets, 2003.

[19] Hart, P., “The condensed nearest neighbor rule”, IEEE Transactions on Information Theory, 1968.

[20] Tomek, I. “An Experiment with the Edited Nearest-Neighbor Rule,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 6, pp. 448-452, 1976.

[21] Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P., “SMOTE: synthetic minority over-sampling technique.” Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.

[22] Fernández, A., García, S. Herrera, F. “SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary,” Journal of Artificial Intelligence Research, vol. 61, pp. 863-905, 2018.

[23] Han, H., Wen-Yuan, W., Bing-Huan, M., “Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning,” Advances in Intelligent Computing, pp. 878-887, 2005.

[24] Nguyen, H., Cooper, E., Kamei, K. “Borderline over-sampling for imbalanced data classification”, International Journal of Knowledge Engineering and Soft Data Paradigms, vol. 3, pp. 4–21, 2009.

[25] Haibo, H., Bai, Y., Garcia, E., Li, S., “ADASYN: Adaptive synthetic sampling approach for imbalanced learning”, IEEE International Joint Conference on Neural Networks, 2008.

[26] Kingma, D. P., Welling, M., “An Introduction to Variational Autoencoders”, Foundations and Trends in Machine Learning: vol. 12 , no. 4, pp 307-392, 2019.

[27] Wojciechowski, S., Wilk, S., Stefanowski, J., “An algorithm for selective preprocessing of multi-class imbalanced data”, Proceedings ofthe 10th International Conference on Computer Recognition Systems, 2017.

[28] Lango, M., Stefanowski, J., “SOUP-Bagging: a new approach for multi-class imbalanced data classification”, Polskie Porozumienie na Rzecz Sztucznej Inteligencji, 2019.

[29] Haixiang, G. et al.,. “Learning from class-imbalanced data: Review of methods and applications”, Expert Systems With Applications 73, pp.220-239. 2017.

[30] Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollar, . “Focal loss for dense object detection”, Proceedings of the IEEE international conference on computer vision, pp. 2980–2988, 2017.

[31] Shapiro, S. S., Wilk, M. B., “An analysis of variance test for normality (complete samples)” Biometrika. 52 (3–4): 591–611, 1965.

[32] Kim, T. K., “T test as a parametric statistic”, Korean J Anesthesiol, 2015.

[33] Krstajic, D. et al., “Cross-validation pitfalls when selecting and assessing regression and classification models”, Journal of Cheminformatics, vol. 6, no. 10, 2014.

[34] García-Jara, G., Protopapas, P., Estévez, P. A., “Improving Astronomical Time-series Clas-

sification via Data Augmentation with Generative Adversarial Networks", The Astrophysical Journal, vol. 953, no. 23, 2022.