



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

**CLASIFICADOR DE CURVAS DE LUZ UTILIZANDO MODELO XGBOOST Y
TÉCNICAS DE BALANCE DE DATOS**

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

JAVIER ANTONIO MOLINA FERREIRO

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

MIEMBROS DE LA COMISIÓN:
PABLO ZEGERS FERNÁNDEZ
IGNACIO REYES JAINAGA

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: JAVIER ANTONIO MOLINA FERREIRO
FECHA: 30/03/2022
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA

CLASIFICADOR DE CURVAS DE LUZ UTILIZANDO MODELO XGBOOST Y TÉCNICAS DE BALANCE DE DATOS

Los telescopios de *survey* recolectan todas las noches grandes cantidades de datos sobre las variaciones en el brillo de objetos estelares o bien, de su movimiento, denominadas alertas astronómicas. Dado el volumen de datos y la velocidad a la que se producen, se requieren agentes intermediarios, denominados *brokers*, quienes realizan la clasificación de alertas.

ALeRCE es un *broker* que recibe las alertas provenientes del *survey* astronómico ZTF (*Zwicky Transient Facility*) y entre sus principales labores está la rápida clasificación de las alertas, siendo capaz de separar las alertas falsas de las reales, y dentro de estas últimas, identificar hasta 15 clases distintas.

En el presente trabajo se evaluó el potencial del algoritmo de aprendizaje XGBoost en la tarea predictiva de clasificación de curvas de luz. Actualmente ALeRCE utiliza el modelo *Balanced Random Forest* (BRF). La motivación detrás de este estudio es el gran desbalance de los datos, el cual es agravado debido a las múltiples clases existentes. Por este motivo se propone el estudio e implementación de técnicas para evitar el efecto de entrenar modelos con desbalance de datos.

Para el entrenamiento del modelo XGBoost usando distintas técnicas de balance, se implementó el procedimiento *Nested Cross Validation* mediante el cual se entrena y evalúa cada modelo 10 veces con distintos grupos de entrenamiento y test, de forma de obtener valores promedio para el desempeño. Este mismo procedimiento fue realizado además para replicar el clasificador de ALeRCE con BRF, para así poder comparar el desempeño de ambos modelos.

Fueron varias técnicas de desbalance con las que XGBoost mejoró sus resultados. Al analizar las matrices de confusión resultantes se comprobó una disminución en el sesgo hacia las clases mayoritarias por parte del modelo predictivo. Se destaca la técnica de balance *Cost Sensitive Learning*, con la cual XGBoost superó a BRF en todos clasificadores que componen el clasificador de curvas de luz, obteniendo en la unión de niveles del clasificador valores de 0.67, 0.79 y 0.70 para Precision, Recall y F1-score respectivamente, en contraste con BRF que obtuvo valores de 0.57, 0.76 y 0.60 para las mismas métricas respectivamente, demostrando además que las diferencias de desempeño obtenidas fueron estadísticamente significativas.

Por último, se realizó un test final en los modelos con los que se obtuvo mejores resultados. Las curvas de luz de este test corresponden a aquellas que no fueron utilizadas ni durante la experimentación del presente trabajo ni por ALeRCE en el paper en el que presentaron su clasificador de curvas luz. De los resultados obtenidos se confirma que *Cost Sensitive Learning* es la mejor técnica de balance de datos para utilizar en conjunto con XGBoost.

A mi familia y amigos, por su constante apoyo y ayuda para seguir adelante.

Tabla de Contenido

1. Introducción	1
1.1. Identificación y formulación del problema	2
1.2. Objetivos	6
1.2.1. Objetivo general	6
1.2.2. Objetivos específicos	6
1.3. Organización del trabajo de memoria	6
2. Marco Teórico y Estado del Arte	8
2.1. Modelos predictivos	8
2.1.1. Balanced Random Forest	8
2.1.2. Extreme Gradient Boosting	8
2.2. Factores que dificultan el aprendizaje	11
2.3. Técnicas de balanceo de datos	13
2.3.1. Sobremuestreo	13
2.3.2. Submuestreo	18
2.3.3. Combinación de aumento y limpieza de datos	20
2.4. Técnicas de balanceo a nivel algoritmo	22
2.5. Evaluación	24
3. Metodología	28
4. Experimentos y resultados	33
4.1. Replica del clasificador de curvas de luz de ALeRCE	33
4.2. Entrenamiento de XGBoost con selección de hiperparámetros	37
4.3. Entrenamiento de XGBoost en conjunto con primeras técnicas de balance de datos	42
4.4. Entrenamiento con técnicas de Sobremuestreo	53
4.5. Entrenamiento con técnicas de Submuestreo y Sobremuestreo	59
4.6. Entrenamiento con técnicas híbridas	66
4.7. Entrenamiento con combinación de técnicas de balance a nivel de datos y Cost Sensitive Learning	69
4.8. Entrenamiento con Focal Loss Cross-Entropy como función de pérdida	74
5. Análisis Estadístico de Resultados	82
6. Evaluación de mejores modelos con datos nuevos de ALeRCE	87
7. Conclusiones	96

8. Anexos	102
8.1. Matrices de confusión	102
8.1.1. Resultados obtenidos por ALeRCE en [4]	102
8.1.2. Glosario hiperparámetros	103
8.1.2.1. XGBoost	103
8.1.2.2. Técnicas de <i>Oversampling</i>	104
8.1.2.3. Técnicas de <i>Undersampling</i>	104
8.2. Cálculo de gradientes y Hessiano	104
8.2.1. Gradiente de la función <i>softmax</i>	104
8.2.2. Gradiente de la función Focal Loss Cross-Entropy	105
8.2.3. Hessiano de la función Focal Loss Cross-Entropy	107

Capítulo 1

Introducción

El uso de algoritmos y modelos de *machine learning* tiene como finalidad resolver problemas del mundo real en base a los datos de manera eficiente y optimizada. Uno de los tipos de problemas con los cuales se emplea el aprendizaje automatizado es la clasificación de instancias o eventos dentro de distintas categorías o clases.

Para que estos modelos puedan aprender a clasificar, deben pasar por un proceso de entrenamiento el cual puede requerir la disponibilidad de un gran volumen de datos. Pero para poder entrenar un buen clasificador lo ideal es que se tenga una cantidad suficiente de instancias por clase, de forma que el modelo pueda captar las características intrínsecas y relaciones entre atributos que comparten los datos de una misma clase, además de poder aprender las diferencias con el resto los otros tipos de instancias, de manera de lograr un buen desempeño a la hora de clasificar.

En problemas reales es bastante común que los datos presenten un desbalance en la cantidad de instancias por clase. Esto debido a que los eventos de ciertas clases no tienen la misma probabilidad de ocurrencia que otras, ya sea porque se necesitan ciertas condiciones especiales para que sucedan o simplemente no son frecuentes o usuales, por lo que adquirirlos no es fácil.

Por tanto, si a un modelo de clasificación se le entrega un grupo de datos que presentan una significante desproporción en la cantidad de instancias por clases, es esperable que el modelo no adquiera la suficiente capacidad de discriminar entre clases. Más aun, se puede presentar un sesgo hacia la clase mayoritaria, prediciendo instancias de la clase minoritaria de manera errónea, las cuales suelen ser las de mayor relevancia a la hora de reconocer.

Un área sobre la cual se utiliza bastante el aprendizaje de máquinas es la astronomía. Todas las noches son varios los telescopios que hacen un barrido por el cielo buscando cambios en el firmamento, en particular lo que se conoce como una alerta astronómica, que es una variación en el brillo en alguna zona del espacio observable o el movimiento de un objeto estelar.

Esta tarea de observación produce grandes volúmenes de datos que luego deben ser procesados haciendo uso de una gran capacidad computacional, para luego poder clasificar las alertas observadas. Es en esta última tarea que es necesario tener profesionales con cierto nivel de experticia para poder identificar los objetos de los cuales provienen las alertas, esto por medio de analizar en el tiempo las alertas anteriores y nuevas del mismo objeto. Pero es

debido al gran número de alertas por noche que ésta es una tarea imposible de realizar en plenitud. La solución a este problema ha sido implementar modelos predictivos entrenados para que sean capaces de analizar las alertas astronómicas para poder discernir entre objetos reales (y clasificarlos entre los distintos tipos conocidos) o bien identificar si corresponde una falsa alerta.

Al igual que tantos otros problemas de la vida real, la observación de los distintos tipos de objetos astronómicos está ligado a su propia naturaleza estocástica, en la que ciertos eventos tienen una probabilidad de ocurrencia mayor que otros, por lo que no es la excepción a la problemática de desbalance de datos a la hora de entrenar un modelo de inteligencia artificial, provocando que sea necesario incorporar técnicas de balanceo de datos.

En el presente trabajo se abordará el tema de clasificación multi-clase haciendo uso de un algoritmo de *machine learning* conocido como XGBoost [1], todo esto inserto en un problema real de identificación de curvas de luz provenientes de alertas astronómicas.

1.1. Identificación y formulación del problema

ALeRCE [2] es una iniciativa chilena liderada por un grupo de científicos provenientes de múltiples disciplinas e instituciones como la Universidad de Chile, Universidad Católica, Universidad Católica de Valparaíso, Universidad de Concepción, Instituto Milenio de Astrofísica y varias otras instituciones junto con otros colaboradores internacionales.

ALeRCE se define como un *alert broker*, lo cual podría traducirse como un agente ó intermediario de alertas astronómicas. Actualmente recibe en tiempo real el flujo de alertas observadas del Zwicky Transient Facility (ZTF), el cual realiza un sondeo del hemisferio norte por medio de una cámara montada en el telescopio Samuel Oschin ubicado en el estado de California, Estados Unidos. Las alertas recibidas son procesadas, almacenadas, clasificadas, etiquetadas y agregadas a bases de datos que se encuentran a disposición de todo público por medio de plataformas [3] que permiten su descarga, su visualización y más.

Entre los distintos servicios y modelos que ofrece ALeRCE se ha implementado un clasificador de curvas de luz [4]. Una curva de luz es una forma de caracterizar un objeto astronómico por medio de la luminosidad de este en función del tiempo [5]. En esta caracterización se suelen presentar variaciones en la luminosidad en el tiempo que son particulares de varios tipos de eventos astronómicos identificables.

La figura 1.1 muestra ejemplos de curvas obtenidas de ALeRCE. En estos gráficos aparecen graficados de izquierda a derecha un Blazar, un objeto estelar joven y una Supernova del tipo Ia. Estas curvas son obtenidas a partir de una secuencia de observaciones de alertas astronómicas del mismo objeto, graficando en el tiempo (dado por la fecha en el calendario juliano modificado, MJD) la diferencia de magnitud o la magnitud aparente (según el caso) de la luz en las bandas g y r (frecuencia de la luz dentro del espectro verde y rojo respectivamente).

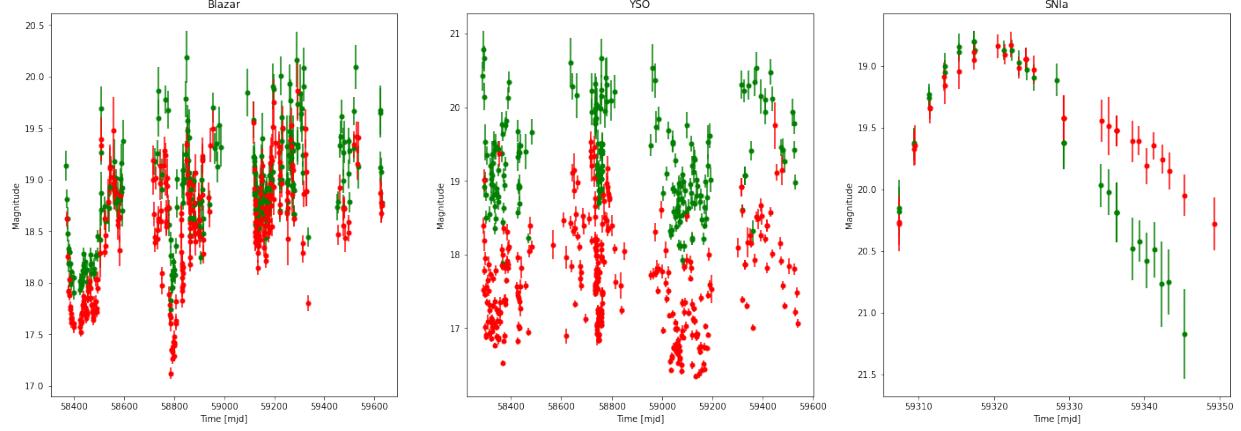


Figura 1.1: Ejemplos de curvas de luz

El clasificador implementado por ALeRCE utiliza un total de 152 características, donde gran parte de estas son calculadas a partir de los datos públicos del ZTF de las mediciones de fotometría de las bandas g y r. Además, se incluyen características provenientes del uso de ambas bandas en conjunto y de coordenadas galácticas, entre otras.

ALeRCE considera un total de 15 subclases de objetos astronómicos los cuales siguen la taxonomía ilustrada en la figura 1.2. Como se puede apreciar se utiliza un orden jerárquico para ordenar el total de las subclases, haciendo que cada una de estas pertenezca a una de las tres clases superiores (*Transient*, *Stochastic* y *Periodic*) en base tanto a sus características físicas y las propiedades de la variación de sus curvas de luz.

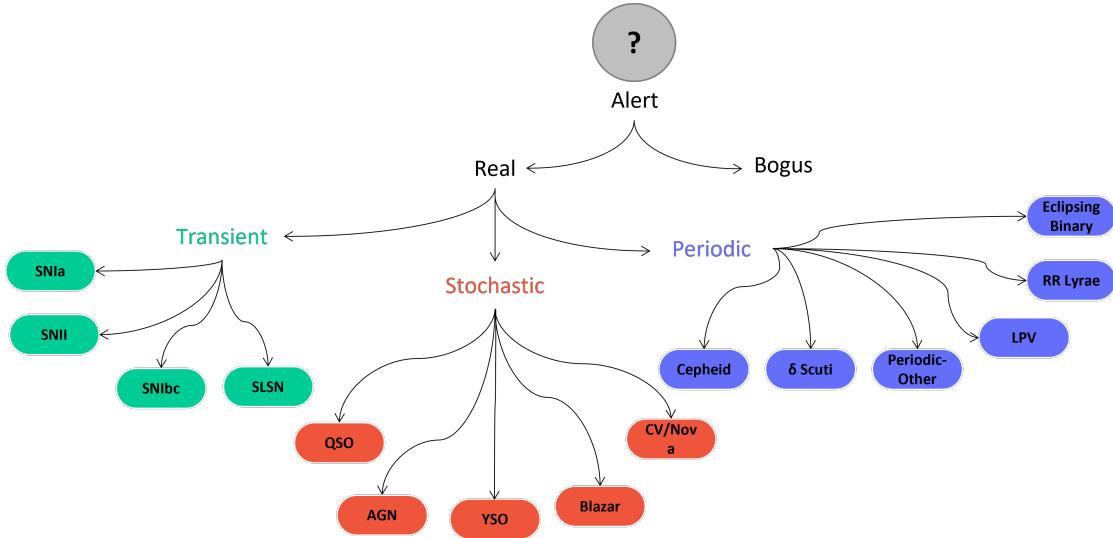


Figura 1.2: Taxonomía de las clases jerárquicas (transiente, estocástica y periódica) y subclases de curvas de luz.

La base de datos que utilizo ALeRCE en [4] para entrenar y probar su clasificador corresponde a curvas de luz reales, las cuales se pudo obtener sus correspondientes etiquetas a partir de cruces con catálogos astronómicos disponibles, logrando un total de 123.496 curvas de luz.

Un problema que presenta la base de datos utilizada es que existe un alto desbalance de datos entre sus clases. La figura 1.3 muestra la cantidad de instancias por clase en escala logarítmica. Se puede observar que entre las clases jerárquicas hay desproporción de datos, habiendo una menor cantidad de instancias de la clase Transiente. Pero más problemático aún es la desproporción de subclases dentro de una misma clase jerárquica, en particular dentro de la clase Transiente se tiene la subclase Supernovas Super Luminosas (SLSN) con solo 24 instancias en contraste con las Supernovas de clase Ia (SNIa) con 1.272 instancias.

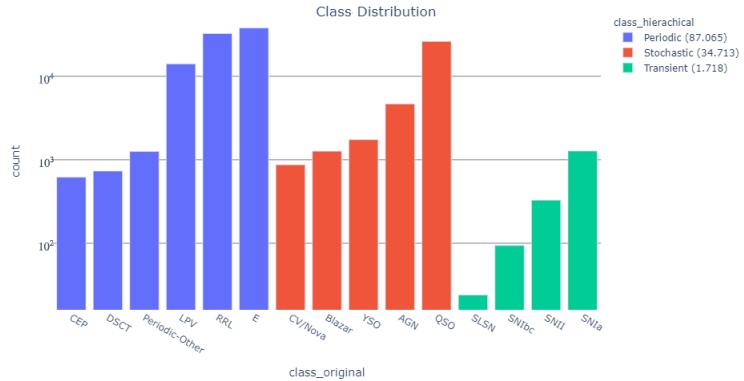


Figura 1.3: Distribución de instancias por clase en el dataset de ALeRCE.

En cuanto al clasificador implementado por ALeRCE, este sigue una estructura similar a la taxonomía propuesta para ordenar las clases. La figura 1.4 ilustra un esquema del clasificador jerárquico, donde en un nivel superior se entrena un clasificador capaz de separar las tres clases jerárquicas, es decir Transiente, Periódicas o Estocásticas, utilizando estas clases como etiquetas. Luego en el nivel inferior, se entrenan otros tres clasificadores independientes, uno para cada clase jerárquica, los cuales son entrenados para clasificar las subclases dentro de una clase jerárquica, por lo que solo se usan curvas de luz de esa clase jerárquica y sus etiquetas para entrenar.

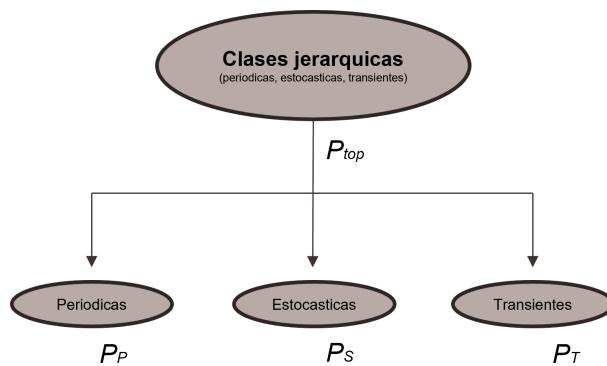


Figura 1.4: Esquema de dos niveles del clasificador de curvas de luz.

Para obtener la clase predicha por el modelo se unen ambos niveles por medio de multiplicar el puntaje asignado por el clasificador superior a la pertenencia de cierta curva a cada una de las clases jerárquicas [$P_{top}(\text{periodic})$, $P_{top}(\text{stochastic})$, $P_{top}(\text{transient})$] por los puntajes asignados en sus correspondientes clasificadores $\{P_P(\cdot), P_S(\cdot), P_T(\cdot)\}$ a la pertenencia de la

misma alerta a las distintas subclases a distinguir. Por ejemplo, el puntaje de que una curva de luz sea un QSO se obtiene como $P(QSO) = P_{top}(stochastic) \times P_S(QSO)$. Siguiendo el mismo procedimiento con las 15 subclases se obtiene el puntaje para cada una de estas, las cuales en total suman 1. Finalmente, la predicción de todo el modelo será aquella clase que obtenga el puntaje más alto.

El modelo predictivo usado por ALeRCE es *Balanced Random Forest* [6] debido a que este tiene implementado en su algoritmo una técnica de balanceo de datos, el que para esta tarea de clasificación mostró un buen desempeño.

El clasificador BRF se entrenó 20 veces ocupando distintos sets de entrenamiento y prueba provenientes del dataset original en una proporción 80 - 20 respectivamente, procurando siempre mantener la misma proporción original de clases en cada set. La tabla 1.1 muestra los resultados del modelo con los grupos de prueba, en donde las métricas corresponden a su versión macro, la cual se obtiene por medio de promediar entre si los valores obtenidos para cada clase. Además, en la figura 8.1 en el anexo se encuentra la matriz de confusión correspondiente.

Tabla 1.1: Resultados del clasificador con modelo de BRF [6].

Nivel	M-Precision	M-Recall	M-F1 score
BRF-top	0.96±0.01	0.99±0.01	0.97±0.01
BRF-bottom	0.57±0.01	0.76±0.02	0.59±0.01

Los resultados del clasificador en el nivel superior son bastante buenos, pero en cuanto al nivel inferior hay una clara baja de desempeño, que, al complementarlo con su matriz de confusión, se puede ver una clara confusión del clasificador al momento de separar ciertas clases, en especial los distintos Supernovas de la clase Transiente.

A modo de comparación, se probaron otros dos algoritmos de *machine learning* manteniendo el mismo esquema de dos niveles. De estas pruebas los resultados más interesantes se obtuvieron con el modelo de XGBoost, que es una versión avanzada de *Gradient Boosting Trees*. Este es un algoritmo similar a Random Forest en cuanto a ocupar múltiples árboles de decisión y del que más adelante se entrará en detalles. Los resultados obtenidos ocupando este modelo se muestran en la tabla 1.2. En la figura 8.2 del anexo se muestra la matriz de confusión obtenida.

Tabla 1.2: Resultados del clasificador con modelo de XGBoost [1].

	M-Precision	M-Recall	F1-score
XGBoost-top	0.99	0.99	0.99
XGBoost-bottom	0.72	0.72	0.71

Se puede observar que, en cuanto a las métricas utilizadas, el modelo de XGBoost supera a BRF excepto en el Recall del nivel inferior. Sin embargo al analizar la matriz de confusión resultante se puede ver que la tasa de predicciones correctas del modelo varía para las distintas subclases entre un 5 y 100 %, presentando una clara correlación entre estos valores y la

cantidad de instancias en las respectivas clases, alcanzando un peor desempeño en las clases de menor proporción.

Aun así, los resultados obtenidos por XGBoost fueron bastante prometedores, teniendo en cuenta que se ocuparon casi todos sus hiperparámetros en sus valores por defecto y que al no tener incorporado en su algoritmo alguna técnica de balance como BRF, se ocupó una técnica básica de balanceo de datos llamada *Random Oversampling*, que aumenta mediante repetición de los ejemplos la cantidad de instancias de las clases minoritarias.

A raíz de los relativos buenos resultado de XGBoost, es que se identifica una oportunidad para mejorar los resultados de este, y evaluar si puede sobreponer el desempeño obtenido con BRF. Esto por medio de implementar técnicas más sofisticadas para tratar el desbalance de datos y profundizar en el funcionamiento y elección de hiperparámetros del algoritmo XGBoost.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo general es incorporar al clasificador XGBoost técnicas de balance de datos y aplicar el modelo resultante a la clasificación de curvas de luz astronómicas.

1.2.2. Objetivos específicos

- Aplicar métricas de desempeño de clasificadores multiclase que reflejen de manera directa el posible efecto del desbalance de datos en el entrenamiento de un modelo.
- Comparar el algoritmo de XGBoost con técnicas de desbalance de datos con el algoritmo *Balanced Random Forest* en la clasificación de curvas de luz con el dataset de ALeRCE.
- En caso que el algoritmo XGBoost no supere al algoritmo de *Balanced Random Forest*, justificar de manera clara esta causa.
- Concebir e implementar un método propio para el aprendizaje con datos desbalanceados.

1.3. Organización del trabajo de memoria

En el siguiente capítulo, Marco Teórico y Estado del Arte, se presentan los conceptos importantes para el presente trabajo, luego en el capítulo 3 se detalla la metodología de trabajo a seguir para la experimentación y obtención de resultados.

En el capítulo 4 se exponen los resultados de los distintos experimentos realizados con XGBoost y técnicas de balance de datos, además de su análisis y comparación con los resultados del algoritmo BRF.

En el capítulo 5 se procede a evaluar si los mejores resultados obtenidos con XGBoost son significativamente diferentes a los obtenidos con BRF.

En el capítulo 6 se realiza un test final a XGBoost con las técnicas de balance de datos con mejores resultados, este test consiste en utilizar las curvas de luz nuevas de ALeRCE.

Finalmente en el capítulo 7 se procede a concluir, analizando los resultados obtenidos y destacando las técnicas con las que se obtuvo un mayor desempeño al combinarlas con XGBoost.

Capítulo 2

Marco Teórico y Estado del Arte

2.1. Modelos predictivos

2.1.1. Balanced Random Forest

Como se mencionó en la introducción, el clasificador de curvas de luz de ALeRCE utiliza como modelo predictivo el algoritmo de *Balanced Random Forest* (BRF) [6]. Este modelo es bastante similar al clásico modelo *Random Forest* (RF) donde se entranan varios árboles de decisión para luego unir sus salidas para obtener una sola predicción.

La principal diferencia de RF con su versión balanceada, es que esta última en el muestreo de instancias para entrenar cada árbol de decisión (*bootstrap sampling*) se asegura la presencia de las clases minoritarias, para luego hacer un muestreo aleatorio en el resto de las clases. Esta implementación es de gran utilidad para evitar el efecto del desbalance de clases, ya que el procedimiento normal de muestreo se realiza de manera completamente aleatoria sobre el total de datos de entrenamiento. Por lo que es esperable que en el entrenamiento de los árboles de decisión haya una baja o nula presencia de las clases minoritarias, provocando que estos árboles no aprendan lo suficiente de todas las clases a clasificar.

Otra diferencia que se destaca en el algoritmo BRF es la posibilidad de inducir los árboles de decisión que lo componen a alcanzar su máximo tamaño, para que así los nodos destinados a las clases minoritarias no sean podados y estén presentes en la salida predictiva del modelo.

2.1.2. Extreme Gradient Boosting

Extreme Gradient Boosting o XGBoost [1], es un software abierto (*open source*) que contiene un modelo de aprendizaje de máquinas para problemas de regresión y clasificación. Actualmente, el modelo de XGBoost es uno de los más populares para resolver problemas que emplean datos tabulares. Una prueba de esto es analizar a los ganadores de las competencias de *data science* que anfitriona el sitio web de competiciones de *machine learning* Kaggle [7], donde en el último tiempo más de la mitad de sus ganadores han ocupado en sus soluciones el modelo XGBoost.

Este modelo es una versión avanzada del algoritmo de *Gradient Boosting Trees*, el cual es un

método de *machine learning* basado en el entrenamiento de varios árboles de decisión cuyas salidas son sumadas para obtener una salida final.

A grandes rasgos, el proceso de entrenamiento se realiza en varias iteraciones. En cada iteración se adiciona un único árbol de decisión al modelo el cual es entrenado para que pueda predecir los errores cometidos por el modelo en la iteración pasada. De esta forma, la salida del modelo será igual a la predicción en la iteración anterior más la predicción del nuevo árbol de decisión, ponderado por una tasa de aprendizaje para así reducir la influencia de cada árbol a la salida de todo el modelo y evitar el sobreajuste.

XGBoost presenta varias optimizaciones en su implementación tanto a nivel de su algoritmo como también a nivel sistema, las cuales hacen que su tiempo de entrenamiento sea notablemente más rápido en comparación con otros modelos y que presente una mayor precisión en sus predicciones. Las cinco optimizaciones más relevantes son:

- *Weighted Quantile Sketch.*

Un método para proponer mejores candidatos de split al momento de construir los árboles de decisión. Este se basa en dividir los datos en distintos cuantiles de forma que cada uno sume una cantidad similar de peso entre las instancias que contiene. El peso de cada instancia corresponde al Hessiano de la función de pérdida de la instancia. Finalmente, los candidatos serán los valores en los límites de los cuantiles, resultando así en una búsqueda más detallada en áreas en que el modelo presenta mayor deficiencia.

- Paralelización.

Ayuda a la velocidad al momento de construir los árboles de decisión, esto por medio de probar distintas ramificaciones y divisiones de nodos de manera paralela.

- *Sparsity-Aware Split Finding.*

Para tratar zonas con escasos datos, estos pueden ser valores faltantes (NaN) o frecuentes valores iguales a cero. Al aparecer este tipo de valores el método prueba asignar las correspondientes muestras a ambas direcciones (mayor o menor) de los valores candidatos a dividir los nodos, seleccionando el candidato con el cual se obtenga una mayor ganancia. Luego en la construcción de próximos arboles se seleccionara la misma dirección hacia donde asignar este tipo de valores conflictivos, evitando repetir el mismo computo.

- Optimizaciones del Hardware

Utilización de memoria caché para minimizar el costo de acceso a los datos. Además, en presencia de múltiples discos los datos son compartidos por motivos de optimización.

- Submuestreo de columnas y filas

Para reducir el tiempo de entrenamiento y sobreajuste de datos se agrega la opción de muestreo aleatorio de los datos entrenamiento, tanto a nivel de fila (por instancia) como columna (por feature).

Sea el conjunto $D = \{(x_i, y_i)\}$, donde $x_i \in \mathbb{R}^m$ son los datos de entrada de dimensión m e $y_i \in \mathbb{R}$ las etiquetas de los correspondientes datos. Como se mencionó el modelo está basado en la adición de varios árboles de decisión f_k , por ello la función de salida se modela como:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^K f_k(x_i), \quad f_k \in \mathcal{F}, \quad (2.1)$$

donde $\mathcal{F} = \{f(x) = w_{q(x)}\}(q : \mathbb{R}^m \rightarrow T, w \in \mathbb{R}^T)$ es el espacio de árboles donde w_i es el peso de la hoja i -ésima, q representa función de estructura del árbol y T el número de hojas. Para entrenar el modelo se minimiza la siguiente función objetivo:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \quad (2.2)$$

donde l es la función de pérdida que mide la diferencia entre la predicción del modelo y salida esperada. Esta función es diferenciable y convexa. La función Ω tiene un rol regulador, penaliza la complejidad de cada árbol f_k y se calcula como $\Omega = \gamma T + \frac{1}{2}\lambda||w||^2$, donde T es la cantidad de hojas del árbol, γ y λ son parámetros reguladores y w los pesos de las hojas. Esta función es incluida ya que sirve para evitar el sobreajuste a los datos de entrenamiento por medio de preferir árboles más sencillos.

El modelo se entrena de manera iterativa por medio de entrenar un árbol a la vez. Basándose en la ecuación 2.1, la salida del modelo en la iteración t -ésima se puede escribir como $\hat{y}^{(t)} = \hat{y}^{(t-1)} + f_t$ donde f_t es el árbol entrenado en la iteración t -ésima. Reemplazando esta expresión en la ecuación 2.2 se obtiene la función de objetivo en la iteración t -ésima:

$$\mathcal{L}^{(t)} = \sum_{i=1}^n l(\hat{y}^{(t-1)}(x_i) + f_t(x_i), y_i) + \Omega(f_t). \quad (2.3)$$

En cada iteración se agrega al modelo en cada iteración el árbol de decisión f_t que minimice en mayor medida el error en las predicciones. Con el fin de optimizar la función objetivo del modelo se realiza una aproximación de segundo grado a la función de pérdida l . Aplicando dicha aproximación y quitando los valores constantes se obtiene la siguiente función objetivo simplificada:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t), \quad (2.4)$$

donde $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ y $h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$ son las derivadas parciales de primer y segundo grado de la función de pérdida con respecto a la predicción del modelo en la iteración anterior. Es a raíz del uso de gradientes para optimizar que el modelo se denomina como Gradient Boosting.

Definiendo $I_j = \{i | q(x_i) = j\}$ como el conjunto de instancias que caen en la hoja j -ésima de un árbol de decisión de estructura q y expandiendo la función reguladora Ω se tiene:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2. \quad (2.5)$$

Ya que todas las instancias que caen en la misma hoja son ponderadas por el mismo peso, se procede a reformular la ecuación 2.5 por medio de reemplazar f_t por su respectiva predicción:

$$\begin{aligned}\tilde{\mathcal{L}}^{(t)} &= \sum_{j=1}^T \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T \\ \tilde{\mathcal{L}}^{(t)} &= \sum_{j=1}^T \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T,\end{aligned}\tag{2.6}$$

donde G_j y H_j representan la suma de los gradientes de primer y segundo orden, respectivamente, de las instancias que caen en la hoja j -ésima. Finalmente, ocupando la última ecuación en 2.6 se calcula el valor óptimo del peso por hoja w_j^* de un árbol de estructura $q(x)$ por medio de tomar la derivada de la función de pérdida con respecto al peso de cada hoja e igualando a cero:

$$\begin{aligned}\frac{\partial \tilde{\mathcal{L}}^{(t)}}{\partial w_j^*} &= 0 \\ G_j + \frac{1}{2} (H_j + \lambda) \cdot 2w_j^* &= 0 \\ \Rightarrow w_j^* &= -\frac{G_j}{H_j + \lambda}.\end{aligned}\tag{2.7}$$

De este modo se calcula el valor óptimo en la función objetivo en la iteración t -esima resulta en:

$$\tilde{\mathcal{L}}^{(t)} = -\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T.\tag{2.8}$$

La ecuación 2.8 se ocupa para comparar distintos árboles de decisión, donde mientras menor sea el valor, mejor es el árbol. Además, la ecuación 2.8 se utiliza para comparar el rendimiento de diferentes candidatos de partición en cada nodo del arbol. Estableciendo G_L y H_L como la suma de los gradientes de primer y segundo grado de las instancias que caen en el nodo izquierdo y G_R y H_R a las del derecho, se define el puntaje por partición en un árbol de decisión como:

$$\tilde{\mathcal{L}}_{split} = -\frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma.\tag{2.9}$$

El entendimiento del planteamiento matemático detrás del algoritmo de XGBoost es beneficioso a la hora de elegir los hiperparámetros del modelo a implementar. Será necesario establecer de manera adecuada estos valores para poder sacarle el máximo provecho a la capacidad de aprendizaje y predicción a este modelo.

2.2. Factores que dificultan el aprendizaje

Usualmente cuando se entrena un modelo clasificador con un dataset que presenta desbalance de datos se entiende que el problema recae en la desproporción de la cantidad de instancias por clase. Sin embargo estudios de las propiedades y la distribución subyacente de las instancias que forman las clases minoritarias, han demostrado que no sólo la desproporción de clases dificulta el aprendizaje de los clasificadores, sino que también otros factores

que sumados a la pequeña cantidad de datos, componen lo que se denomina *Data difficulty factors* [8].

Uno de los factores es la descomposición de la clase minoritaria en distintos subgrupos de pocas instancias cada uno, lo cual provoca espacios disjuntos dentro de la misma clase, tal como se ejemplifica en el gráfico de la izquierda en la figura 2.1, resultando en una mayor dificultad al momento de separar las clases de manera correcta. Esto se podría ver como un desbalance dentro de la misma clase.

Otro factor que dificulta el proceso de entrenamiento es el traslape (*overlapping*), que ocurre cuando dos clases tienen regiones sobrepuertas entre si o bien existen instancias de la clase minoritaria que se encuentran dentro de una región, típicamente perteneciente a la clase mayoritaria como es el caso del ejemplo en el gráfico de la derecha de la figura 2.1.

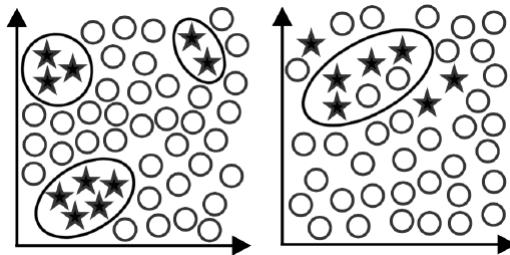


Figura 2.1: Ejemplos de factores que dificultan el aprendizaje, *Data difficulty factors*. Figura obtenida de [8].

Para entender mejor la distribución de los datos en las clases minoritarias se puede realizar una tipificación de las distintas instancias de cada clase [9]. Una vez realizada esta tarea se le puede sacar provecho por medio de estudiar el comportamiento del clasificador en cada tipo de instancia y, además, se pueden tomar decisiones o medidas en la data misma o en el entrenamiento para poder mejorar el desempeño resultante.

Los dos principales tipos de ejemplos son las instancias seguras y las inseguras. Las instancias seguras se ubican en regiones homogéneas, las cuales son pobladas por solo o casi solo instancias de la clase minoritaria. Por otro lado, dentro de las instancias inseguras se distinguen aquellas del tipo *borderline* que son las que se encuentran en el límite entre dos clases, *outliers* que son las instancias de las clases minoritarias rodeadas de solo clases mayoritarias y ejemplos raros que son subgrupos pequeños de instancias minoritarias que se encuentran juntas pero dentro de regiones de la clase mayoritaria.

Para poder asignar cada instancia a un determinado tipo, se analizan los vecindarios de éstas, el método más común para esta tarea es *k-nearest neighbours* con $k = 5$. Definiendo la notación $x : y$ del vecindario de una instancia, siendo x los vecinos de la misma clase e y los vecinos de otras clases, en [9] se categorizan las instancias como:

- 5 : 0 o 4 : 1 la instancia es segura (S)
- 3 : 2 o 2 : 3 la instancia es *borderline* (B)
- 1 : 4 la instancia es un ejemplo raro (R)

- 0 : 5 la instancia es *outlier* (O)

2.3. Técnicas de balanceo de datos

Una de las principales formas de abordar el problema de aprendizaje en un escenario con desbalance de datos es aplicar técnicas de balanceo en la etapa de preprocesamiento [10], o sea en los mismos datos antes de utilizarlos para entrenar algún modelo predictivo, con el objetivo de compensar la distribución no uniforme de los datos.

Son varias las técnicas o metodologías que abordan este problema desde los datos, pero siguiendo la misma lógica. Estas técnicas se pueden clasificar en tres categorías principales: técnicas de sobremuestreo (*oversampling*), técnicas de submuestreo (*undersampling*) y técnicas híbridas, que básicamente son una combinación de las dos anteriores.

A continuación, en esta sección se abordará cada una de las tres categorías mencionadas junto con algunas técnicas en específico que fueron utilizadas en este trabajo.

2.3.1. Sobremuestreo

El sobremuestreo (*oversampling*), se basa en aumentar la cantidad de instancias de las clases minoritarias a niveles que sean más proporcionales a las otras clases en cuestión. Entre las variantes existentes, la más común y conocida es *Random Oversampling* (ROS) [10], la cual selecciona instancias minoritarias de manera aleatoria para ser duplicadas y agregadas al conjunto de datos.

La principal ventaja de ocupar sobremuestreo es que no se pierde información alguna en el muestreo, pero por otro lado, al repetir instancias minoritarias en los datos de entrenamiento se tiene el peligro de sobreajustarse a estos datos, resultando en un bajo desempeño al evaluar el desempeño del modelo en el grupo de testeо.

Una manera alternativa de aumentar las instancias minoritarias evitando sobreajustar al modelo a los datos de entrenamiento es generar nuevas instancias de manera sintética a partir de las originales. Esta técnica es conocida como SMOTE o bien *Synthetic Minority Over-sampling Technique* [11].

SMOTE escoge una instancia minoritaria y considera su vecindario, es decir aquellas otras instancias de la misma clase que se encuentren cercanas. Luego realiza una interpolación entre sus vecinos incorporando ponderadores aleatorios para introducir cierta interferencia, de manera de obtener nuevas instancias de la clase minoritaria que se encuentren dentro del vecindario de la instancia original.

De manera más detallada, se elige una instancia de la clase minoritaria, luego se obtienen sus k vecinos más cercanos, de estos k vecinos n son seleccionados al azar para realizar la interpolación. Este cálculo se realiza obteniendo la diferencia entre los valores de las características de la instancia en cuestión y sus n vecinos seleccionados. Estas diferencias son ponderadas por un número aleatorio entre 0 y 1 para luego ser sumadas a las características de la instancia original, resultando así en una nueva instancia que se encuentra entre medio de la original

y sus vecinos. La figura 2.2 ilustra el sobremuestreo usando SMOTE, donde los cuadrados azules representan una clase mayoritaria, en naranja se tienen las instancias minoritarias originales y en verde las generadas de manera sintética.

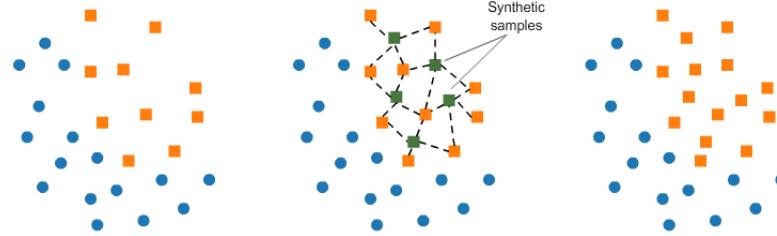


Figura 2.2: Ilustración de sobremuestreo ocupando SMOTE. Figura tomada de [11].

Si bien SMOTE es un método muy popular en temas de balanceo de datos, su principal desventaja es que las instancias sintéticas generadas introducen ruido al conjunto de datos resultante. Además, a medida que aumenta la dimensionalidad del problema, el resultado final empeorará.

A partir de la técnica SMOTE se han generado muchas variantes que modifican los distintos niveles del algoritmo, por ejemplo como escoger las instancias sobre las que generar nuevos datos, la selección del vecindario o bien como calcular las características de las nuevas instancias sintéticas.

Un ejemplo donde se ha implementado SMOTE junto con un proceso de selección de instancias o regiones sobre las cuales generar nuevas instancias sintéticas es Borderline-SMOTE [12]. Esta técnica se basa en la idea que aquellas instancias lejanas a las zonas limítrofes entre clases aportan menos información para la clasificación que aquellas que están más cercanas.

Borderline-SMOTE crea instancias sintéticas a partir de instancias minoritarias que se encuentren cerca del límite de alguna clase mayoritaria. Sean p_i y n_i las instancias minoritarias y mayoritarias respectivamente. Para identificar los p_i cercanos a la frontera se utiliza algún método de *clustering* para obtener sus vecinos más cercanos. Una vez obtenidos estos se compara la cantidad de instancias de la misma clase con aquellas de la clase mayoritaria dentro del vecindario.

Si todos los vecinos de la instancia en cuestión corresponden a la clase mayoritaria se estima que p_i corresponde a ruido, si menos de la mitad de los vecinos son de la clase mayoritaria, entonces se considera como segura o lejana al límite, por lo que no participa en el proceso de generación de instancias. En el caso que más de la mitad de las instancias en el vecindario sean mayoritarias entonces se estima que p_i se encuentra en el límite o cercano a este, por lo que estas instancias son clasificadas como peligrosas.

A partir del subconjunto de instancias p_i consideradas como peligrosas se generan nuevas instancias aplicando la técnica de SMOTE. El algoritmo Borderline-SMOTE tiene dos variantes, la primera denominada *borderline-smote1* solo utiliza instancias de la misma clase minoritaria para generar nuevas, pero en *borderline-smote2* también se utilizan instancias

mayoritarias para generar instancias sintéticas, pero utilizando un ponderador menor a la hora de interpolar las instancias.

La razón de ocupar la clase mayoritaria para obtener nuevas instancias es que las instancias en peligro tienen vecinos de otra clase en su mayoría, por lo que utilizando solo instancias minoritarias se estaría alejando del límite a las nuevas instancias. La idea es que al utilizar sus reales vecinos más cercanos (sin discriminar su clase) se mantendrían cerca del límite las instancias generadas.

Otra técnica que busca aumentar la presencia de la clase minoritaria en la frontera y que también utiliza como base el método SMOTE es SVM-SMOTE [13], la cual el entrenamiento de un modelo predictivo de *Support Vector Machine* (SVM). Al utilizar este método se espera que el área de la clase minoritaria se expanda hacia el lado de las clases mayoritarias, así no solo se aumenta su proporción, sino que además se asegura una mayor presencia las fronteras de las clases.

Para poder encontrar el borde entre clases, se utiliza la frontera encontrada por un modelo de SVM estándar entrenado con los datos originales. A partir de esta frontera se analiza que tan poblado se encuentra de instancias minoritarias y mayoritarias.

En caso de que en la frontera no haya una desproporción muy grande entre clases, se procede a generar instancias sintéticas con la técnica SMOTE por medio de interpolar las instancias minoritarias. La figura 2.3 ilustra un ejemplo donde en la imagen de la izquierda se ejemplifica la interpolación con el vecino más cercano a la instancia seleccionada. En la imagen de la derecha se muestra el resultado donde las instancias minoritarias se representan por un signo positivo, las mayoritarias por un negativo, la línea sólida es el borde establecido por SVM y la línea entrecortada lo que sería el límite ideal para esa distribución.

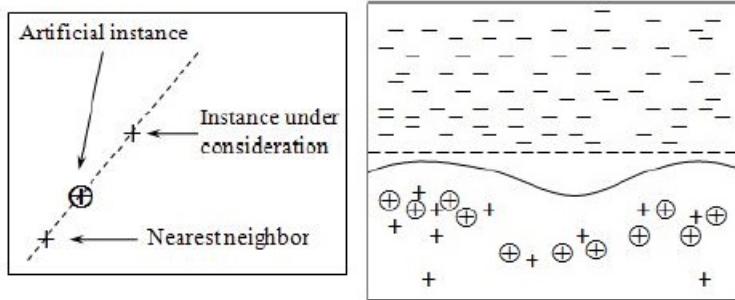


Figura 2.3: Ilustración de interpolación y frontera resultante con la técnica SVM-SMOTE. Tomado de [13].

En caso contrario, cuando se estima que el límite está poblado en gran parte por instancias mayoritarias por sobre las minoritarias, se toma la decisión de generar instancias por medio de la extrapolación, esto es, generar instancias sintéticas en el sentido contrario a donde se encuentra el vecino más cercano de la instancia en cuestión. Esto se realiza para cambiar la distribución de la clase minoritaria en dirección a la clase mayoritaria y así aumentar la presencia en la frontera.

En la figura 2.4 izquierda se ilustra de la extrapolación antes mencionada y a la derecha el resultado de generar instancias sintéticas con esta metodología, lo cual en comparación con la figura 2.3 se puede ver que el límite establecido es deformado en favor de la clase minoritaria.

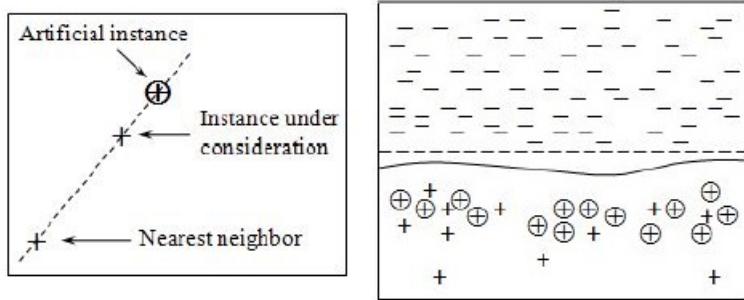


Figura 2.4: Ilustración de extrapolación y frontera resultante con la técnica SVM-SMOTE. Figura tomada de [13].

Otra variante de la técnica SMOTE es generar de manera adaptativa, instancias minoritarias sintéticas a partir de aquellas instancias que se estime que tienen mayor dificultad de aprendizaje en comparación a otras de la misma clase. Esta variante es conocida como *Adaptive Synthetic Sampling Approach* (ADASYN) [14], la cual no sólo tiene como objetivo aumentar la proporción de las clases minoritarias para reducir el sesgo hacia las clases mayoritarias, sino que además mover adaptativamente los límites de decisión hacia las instancias de mayor dificultad de aprendizaje.

Sean m_s y m_l el número de instancias minoritarias y mayoritarias respectivamente. El algoritmo ADASYN comienza calculando el grado de desbalance como $d = m_s/m_l$, luego si este grado es menor que un umbral se procede a estimar el número de instancias minoritarias sintéticas por generar, a través de la expresión:

$$G = (m_l - m_s) \cdot \beta,$$

donde β es un parámetro que define qué tan balanceado resultará el conjunto de datos final. Notar que si $\beta = 1$ se obtiene un balanceo completo. Luego para la instancia x_i perteneciente a la clase minoritaria se procede a encontrar sus k vecinos más cercanos usando el algoritmo de clustering KNN.

Para cada vecindario de las instancias minoritarias, se calcula la razón r_i que refleja la dificultad de aprender cierta instancia minoritaria:

$$r_i = \frac{\Delta_i}{K_i},$$

donde Δ_i es el número de instancias mayoritarias y K_i el número de instancias en el vecindario. Las proporciones obtenidas son normalizadas por la suma total de estas, para así obtener lo que sería una distribución de densidad \hat{r}_i , tal que $\sum_i \hat{r}_i = 1$.

Finalmente, para cada instancia minoritaria x_i se procede a generar $g_i = \hat{r}_i \cdot G$ instancias

sintéticas a partir de las instancias minoritarias en su vecindario obtenido por el algoritmo SMOTE.

Los métodos arriba mencionados en general priorizan las zonas de dificultad de aprendizaje y límites entre clases para generar nuevas instancias sintéticas. La idea es que un modelo de clasificación pueda aprender de mejor manera las diferencias entre clases y no presentar un sesgo hacia las clases mayoritarias. Sin embargo al realizar este procedimiento existe el riesgo de aumentar la sobreposición de clases, lo cual se agrava cuando es un problema de clasificación de múltiples clases.

Para resolver la desventaja mencionada se utiliza una técnica distinta de *oversampling* llamada Mahalanobis Distance Oversampling (MDO) [15], la cual tiene por objetivo generar nuevas instancias en aquellas zonas de clases minoritarias que tengan una mayor densidad, de forma de preservar la covarianza y la estructura de las clases. MDO postula que al aumentar la proporción de clases con esta metodología se generarán nuevas instancias que van a ser más útiles para los algoritmos de aprendizaje.

La distancia de Mahalanobis es una medida que usualmente se utiliza para calcular la similitud de dos variables en un espacio multidimensional. Sean $x_i = (x_{i1}, x_{i2}, \dots, x_{id})^T$ una instancia de d dimensiones con promedio en sus características $\mu_i = (\mu_{i1}, \mu_{i2}, \dots, \mu_{id})^T$ y matriz de covarianza S . Se define la distancia de Mahalanobis como:

$$D_M(x) = \sqrt{(x - \mu)^T S^{-1} (x - \mu)}.$$

MDO toma como esquema una elipse para poder generar nuevas instancias que tengan la misma distancia de Mahalanobis que la instancia desde la cual están siendo originadas, La figura 2.5 ilustra contornos de distancia constante con respecto al centro que corresponde al promedio de las instancias.

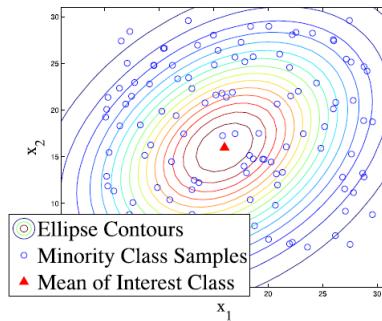


Figura 2.5: Representación de instancias en forma de elipse con la distancia de Mahalanobis. Figura tomada de [15].

La ecuación de la elipse en dos dimensiones x e y con centro en h y k es:

$$\frac{(x^2 - h)^2}{a^2} + \frac{(y^2 - k)^2}{b^2} = 1 \quad (2.10)$$

Generalizando la ecuación 2.10 para el caso de una elipse en d dimensiones se tiene:

$$\frac{x_1^2}{a_1^2} + \frac{x_2^2}{a_2^2} + \cdots + \frac{x_d^2}{a_d^2} = 1, \quad (2.11)$$

donde d es la cantidad de dimensiones y a_i se podría considerar como la variación de la data en su respectiva dimensión. Haciendo uso de 2.11 se puede reemplazar los a_i como:

$$\frac{x_1^2}{\alpha V_1} + \frac{x_2^2}{\alpha V_2} + \cdots + \frac{x_d^2}{\alpha V_d} = 1, \quad (2.12)$$

donde α es una constante a encontrar y V_i corresponde a la varianza de cada dimensión de la data. Los V_i se encuentran a partir de los principales componentes de todas las instancias de la clase objetivo normalizadas por su promedio.

Finalmente, seleccionando instancias al azar y ocupando los V_i , se procede a obtener el α correspondiente a partir de la ec. (2.12). Luego se escogen de manera aleatoria los valores de cada característica dentro del rango $[-\alpha V_i, \alpha V_i]$, garantizando así que cada instancia sintética generada tenga igual distancia de Mahalanobis que la original.

2.3.2. Submuestreo

Otra categoría de técnicas de muestreo es el submuestreo (*undersampling*) que trabaja sobre las clases mayoritarias reduciendo su proporción con el objetivo de que sea similar al de las clases minoritarias. La técnica más conocida y simple es *Random Undersampling* (RUS) [10]. Esta técnica crea un nuevo conjunto de datos, pero con solo parte de las instancias de la clase mayoritaria, las cuales son escogidas de manera aleatoria, manteniendo igual el total de las otras clases minoritarias, para así tener un conjunto de datos más pequeño pero balanceado.

Las ventajas y desventajas de la técnica de *undersampling* se podría decir que son las inversas a las de *oversampling*, es decir que no introduce riesgos de sobreajuste, pero por otro lado se pierde información del dataset que podría ser valiosa en el proceso de entrenamiento.

Una de las formas de sacarle más provecho a las técnicas de muestreo es escoger de manera más selectiva cuales instancias mantener, como aquellas que se estimen esenciales para representar su clase. También otro enfoque puede ser seleccionar cuales instancias mayoritarias remover según qué tan sobrepuestas se encuentran a las otras clases, de modo de sea realizar una limpieza de aquellas instancias conflictivas para el aprendizaje.

Varias técnicas implementan medidas de distancia y/o técnicas de cluster para poder hacer una mejor selección de instancias mayoritarias ya sea para remover o bien mantener en el proceso de reducir la desproporción de las clases. Una de estas técnicas es *Near Miss* [16] que incorpora KNN para realizar un *clustering* para encontrar aquellas instancias de la misma clase que sean más cercanas entre sí, y luego calcular la distancia promedio entre las instancias agrupadas y las de las otras clases.

En base a las distancias obtenidas se proponen cuatro distintos métodos para seleccionar aquellas instancias a remover del set de entrenamiento, estas son las siguientes:

- NearMiss-1: Seleccionar aquellas instancias de clase mayoritaria cuya distancia promedio a las tres instancias minoritarias más cercanas sea la más pequeña.
- NearMiss-2: Seleccionar las instancias de clase mayoritaria cuya distancia promedio a las tres instancias minoritarias más lejanas sea la más pequeña.
- NearMiss-3: Se selecciona un cierto número de instancias mayoritarias que se encuentren cercanas a cada una de las instancias minoritarias, garantizando así que cada instancia minoritaria tenga una algunas instancias mayoritarias cercanas.
- Most Distant: Seleccionar instancias de la clase mayoritaria cuya distancia promedio hacia las tres instancias minoritarias más cercanas sea la más grande.

Otra técnica para reducir el número de instancias de un conjunto de datos es la de Condensed Nearest Neighbours (CNN) [17], la cual también emplea *k-nearest neighbour*. El objetivo de CNN es obtener a partir del total de datos un conjunto representativo de este, pero de menor tamaño y más condensado, evitando mantener aquellas instancias más lejanas de las clases (*outliers*).

El algoritmo CNN comienza creando un subconjunto de datos C que contenga una sola instancia al azar de la clase objetivo (mayoritaria) y todas las minoritarias. Luego se entrena un modelo de KNN con el subconjunto creado para después iterar sobre todo el resto de las instancias. A estas instancias se les predice su clase usando el modelo KNN ya entrenado, en caso de no coincidir con las instancias del subconjunto, las instancias objetivo son incorporadas a C . Este proceso se repite con el subconjunto C actualizado hasta que se predigan de manera correcta todas las instancias restantes o bien todas hayan sido incorporadas a C , el cual será el nuevo conjunto de datos a utilizar.

La lógica detrás de CNN es que es necesario conservar solo aquellas instancias que aportan nueva información a la clasificación, lo cual es estimado en base a si se puede o no predecir su clase real con el subconjunto de datos seleccionados.

A diferencia de CNN que aplica KNN a solo un subconjunto de datos, la técnica Edited Nearest Neighbours (ENN) [18] aplica KNN al total de datos. De KNN solo interesan las clases mayoritarias obtenidas (las clases objetivo) por lo que solo se verifica si la clase real de las instancias mayoritarias coincide con la clase mayoritaria de KNN. En caso de no coincidir se procede a eliminar las instancias correspondientes junto a sus vecinos.

Varias técnicas de submuestreo que siguen lógicas similares a las de las técnicas arriba mencionadas, algunas de estas son: Repeated Edited Nearest Neighbour (RENN) [19] la cual repite varias veces el procedimiento de ENN; All-KNN [19] que solo se diferencia de RENN en que para cada repetición varía el número de vecinos a considerar; Neighborhood Cleaning Rule (NCR) [20] la cual además de remover las instancias mayoritarias al igual de ENN, limpia el vecindario de las instancias minoritarias por medio de quitar los vecinos que correspondan a otras clases.

Un enfoque distinto para reducir el número de instancias de las clases mayoritarias es el de Instance Hardness Threshold (IHT) [21], la cual identifica las instancias más conflictivas (*outliers* o superpuestas) por medio de entrenar un modelo de clasificación. Luego en base a la

probabilidad de clasificar correctamente una instancia en particular se establece la dificultad de clasificación de ésta. Finalmente, se establece un valor umbral de manera que aquellas instancias con probabilidad superior al umbral establecido se mantienen.

2.3.3. Combinación de aumento y limpieza de datos

Como se ha mencionado, se puede sacar gran provecho a las mediciones de distancia entre instancias y a la obtención de clusters al momento de seleccionar instancias ya sea para aumentar la cantidad de instancias de las clases minoritarias o bien reducir la proporción de las clases mayoritarias. Sin embargo, cuando la desproporción de clases es muy desigual, la cantidad de instancias a aumentar de la clase minoritaria o a reducir de las mayoritarias es muy grande, provocando que las desventajas del sobremuestreo o submuestreo sean mayores.

Una buena alternativa para hacer frente al aumento de los efectos negativos que tienen las técnicas de muestreo es combinarlas de forma que se obtenga un conjunto de datos balanceado, pero a un nivel que no sea necesario aumentar ni reducir de manera drástica ninguna de las clases presentes.

Como se ha expuesto hasta el momento, existen varias las técnicas de aumento o reducción de datos las cuales toman decisiones solo en base al análisis del vecindario de las instancias objetivo, sin embargo este procedimiento podría combinarse con información a priori que se tenga de la relación entre las distintas clases del problema. Una técnica que aprovecha el análisis de vecindarios junto con información a priori de las clases es la denominada SPIDER [22], la cual es una técnica híbrida de muestreo enfocada a problemas con múltiples clases.

SPIDER es una técnica enfocada a la selección y preprocesamiento datos en escenarios con desbalance de datos (incluyendo los *data difficulty factors*). Además se enfoca en problemas de múltiples clases, incorporando una nueva categoría que es la clase intermedia, diferenciando así algunas clases que no son ni mayoritarias ni minoritarias.

En su algoritmo incorpora técnicas de submuesteo local, es decir, dentro de cada vecindario de las instancias objetivo se realiza una limpieza ya sea removiendo instancias minoritarias poco seguras o bien de las distintas..

También incorpora lo que sería un sobremuestreo local el cual básicamente es duplicar las instancias de la clase objetivo dentro de un vecindario (como un *random oversampling* hasta que este pueda considerarse seguro).

Por último, SPIDER incorpora en el análisis del vecindario de las instancias de la clase objetivo una matriz de costos asociada a la mala clasificación, la cual puede ser definida por el mismo usuario en base a información de la naturaleza de las clases en cuestión.

El algoritmo SPIDER comienza analizando el vecindario de cierta instancia minoritaria, el que es obtenido por medio del algoritmo *k-nearest neighborhood*. El análisis comienza identificando las clases presentes en el vecindario de mínimo costo (*minimum-cost classes*), es decir las que tengan el menor costo a la mala clasificación.

Las clases de mínimo costo se obtienen minimizando la ecuación 2.13 donde x es la instancia minoritaria a estudiar, C el conjunto de todas las clases, k el número de vecinos, S el conjunto de datos a preprocesar, $cost[c_i, c_j]$ el costo por clasificar mal una instancia de la clase j como si fuese de la clase i y la función $knn(x, k, S, c_i)$ retorna los k -vecinos más cercanos de x que pertenecen a la clase c_i .

$$min_cost_classes(x, k, S, cost) = \arg \min_{c_j \in C} \sum_{c_i \in C} \frac{|knn(x, k, S, c_i)|}{k} \cdot cost[c_i, c_j] \quad (2.13)$$

Al aplicar (2.13) al vecindario de una instancia x , se categorizan las clases presentes por medio de si aparecen o no entre las clases de mínimo costo, en caso de aparecer éstas se estiman como seguras y en caso contrario como inseguras.

Una vez obtenidas las instancias seguras del vecindario, si éstas son de la clase mayoritaria pueden sufrir un cambio de etiqueta a la clase de la instancia minoritaria en observación. Luego procede la limpieza del vecindario de instancias mayoritarias inseguras. Finalmente, con el vecindario más cercano resultante, mientras aún se presenten instancias de clases inseguras (que no sean de mínimo costo) se procede a duplicar las instancias del vecindario que si lo sean hasta que el vecindario este compuesto solo de instancias seguras.

Otra técnica híbrida de muestreo de datos es *Similarity Oversampling and Undersampling Preprocessing* (SOUP) [23]. La particularidad de esta técnica es que además de estar enfocada para problemas multi-clase, toma en cuenta los *data difficulty factors*, en particular la desproporción de instancias por clase y la sobreposición de éstas.

En primer lugar, para medir la superposición entre clases se aplica *k-nearest neighbors* para analizar el vecindario de las instancias y así evaluar que tan seguras son. Una instancia segura a priori serían aquella que su vecindario en su mayoría se compone de instancias de la misma clase, en caso contrario, cuando la mayoría del vecindario no sea de la misma clase se puede considerar como poco segura la instancia y se podría tener el caso de una instancia de borde o *outlier*.

Junto con el análisis del vecindario, se incluye una medición de la desproporción de clases entre las múltiples a tener en cuenta. Para esto, SOUP introduce el valor μ_{ij} , el cual representa la proporción entre la clase C_i y a la clase C_j y es calculado como:

$$\mu_{ij} = \frac{\min(|C_i|, |C_j|)}{\max(|C_i|, |C_j|)}, \quad (2.14)$$

donde $|C_i|$ es el número de instancias de la clase C_i . Luego, utilizando μ_{ij} junto con el análisis del vecindario de cada instancia, se calcula el grado de seguridad de una instancia x perteneciente a la clase C_i como:

$$safe(x_{C_i}) = \frac{1}{n} \sum_{j=1}^l n_{C_j} \mu_{ij}, \quad (2.15)$$

donde n_{C_j} es el número de instancias de la clase C_j dentro del vecindario de la instancia x y n es el número total de vecinos considerados.

En base al grado de seguridad SOUP decide qué instancias de las clases minoritarias volver a muestrear y cuáles de la clase minoritaria remover. Para esto se sigue la lógica de que a mayor seguridad en una instancia minoritaria, mayor será su contribución a la clasificación de su clase. Por el otro lado, entre menor la seguridad de una instancia mayoritaria, más dificultades introducirá al modelo para identificar las clases minoritarias.

SOUP aplica *undersampling* y *oversampling* a todas las clases mayoritarias y minoritarias respectivamente, esto hasta que todas alcancen cierta cardinalidad m , la cual es calculada como el promedio entre el número de instancias de la clase minoritaria más pequeña y el número de instancias de la clase mayoritaria de mayor tamaño.

El algoritmo SOUP comienza por la etapa de *undersampling*, partiendo por la clase mayoritaria de mayor tamaño, se calculan los grados de seguridad de sus instancias y luego se procede a quitar las $|C_i| - m$ instancias menos seguras, donde C_i es la clase en cuestión. Luego se repite este mismo procedimiento con la siguiente clase mayoritaria de mayor tamaño, actualizando siempre los niveles de seguridad de las instancias.

Una vez terminada la etapa de *undersampling* se procede a aumentar la proporción de las clases minoritarias, siguiendo un procedimiento parecido a la etapa anterior, se comienza con la clase minoritaria de menor tamaño, a esta se le calculan los grados de seguridad de sus instancias para luego duplicar las $m - |C_i|$ instancias más seguras. Luego, se repite este mismo procedimiento con la siguiente clase minoritaria de menor tamaño, actualizando los grados de seguridad.

2.4. Técnicas de balanceo a nivel algoritmo

Otra forma de abordar el entrenamiento de modelos con datos desbalanceados es incorporar a la función de costos que se esté ocupando un factor dependiente de la clase a la cual se esté evaluando su predicción, de manera que se penalice en mayor proporción las malas clasificaciones de las clases minoritarias.

Esta técnica se conoce como *Cost-sensitive learning* [10] y puede ser incorporada tanto a nivel de preprocesamiento de datos como a nivel del algoritmo. Usualmente se calcula un vector de costos C_i de igual tamaño que la cantidad de clases en cuestión, donde cada costo c_i es inversamente proporcional a la cantidad de instancias de la clase i , es decir, a menor cantidad de instancias, mayor el costo. Otra forma de incorporar costos en el entrenamiento es por medio de una matriz de costos C_{ij} en la que cada componente c_{ij} representa el costo de clasificar erróneamente una instancia de la clase i como clase j .

Si bien la implementación de *Cost-sensitive learning* es computacionalmente más eficiente, en especial para datasets de grandes volúmenes, no es tan popular como las técnicas de remuestreo las cuales no presentan el mismo nivel de eficiencia. Esto principalmente por la dificultad de establecer los valores de costos para las distintas clases, la idea no es tan intuitiva y directa de aplicar, en especial para aquellas personas con bajo grado de experticia.

Otro enfoque para abordar el problema de desbalance desde el algoritmo es desde la función de pérdida que este utiliza para su entrenamiento. De estas funciones la más utilizada en problemas de clasificación binarios o multi-clase es la Entropía Cruzada (*Cross Entropy*).

La Entropia Cruzada se expresa mediante la siguiente ecuación:

$$CE = -\frac{1}{N} \sum_{i=1}^N \log(p_{ti}) \quad (2.16)$$

donde N es el total de instancias del conjunto de entrenamiento y p_{ti} la probabilidad asignada por el clasificador a que la instancia i -esima pertenezca a su verdadera clase.

La figura 2.6 muestra en azul la magnitud de pérdida obtenida por la función logaritmo de (2.16) en función a la probabilidad p_t . Se puede observar que instancias que son relativamente fáciles de clasificar por el modelo empleado ($p_t > 0.6$) obtienen una magnitud de pérdida no despreciable. En conjunto de datos desbalanceados donde las instancias de clases mayoritarias son varias, las fáciles de clasificar terminan sumando una cantidad no menor de pérdida que opaca la obtenida por aquellas instancias más difíciles de clasificar, que suelen ser las de clases minoritarias.

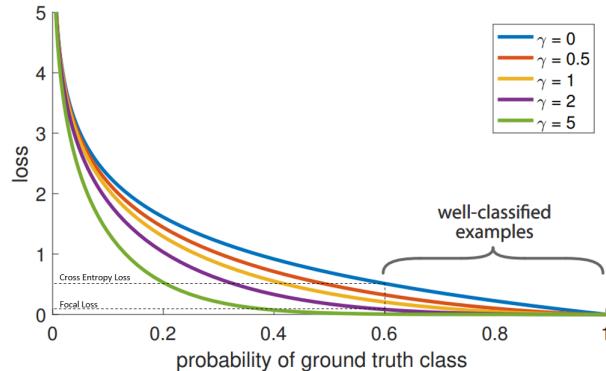


Figura 2.6: Función de pérdida de Cross Entropy vs Focal Loss para distintos γ . Figura tomada de [24].

Una forma de abordar el problema del desbalance desde la función de pérdida de un modelo es utilizar una variante de *Cross Entropy* [24], ideada inicialmente para la detección de objetos. La función llamada *Focal Loss* [24] busca disminuir la pérdida obtenida con entropía cruzada en aquellas instancias fáciles de clasificar (con mayor p_t) por un modelo.

El *Focal Loss* nace de ponderar la entropía cruzada definida en (2.16) por la probabilidad de que la instancia en cuestión pertenezca a la otra clase (esta función es propuesta para clasificación binaria) elevado a un factor $\gamma \geq 0$, y se define como sigue:

$$FL = -\frac{1}{N} \sum_{i=1}^N (1 - p_{ti})^\gamma \log(p_{ti}) \quad (2.17)$$

Al ocupar $\gamma = 0$ se obtiene la función Entropia Cruzada, pero para $\gamma > 0$ se reduce el valor de pérdida, tal como se puede ver en la figura 2.6, pero con la particularidad de que entre mayor sea la confianza del clasificador, mayor es la disminución de la función de pérdida,

provocando que aquellas instancias de menor confianza y mal clasificadas tengan un mayor enfoque en el entrenamiento del modelo.

Implementar Focal Loss podría ayudar a tratar el desbalance de datos puesto que aumenta el peso de aquellas instancias difíciles de clasificar para un modelo, que en su efecto suelen ser pertenecientes a las clases minoritarias. En [25] se implementó la función Focal Loss para algoritmo XGBoost con una librería enfocada a resolver problemas de clasificación binaria con datos no balanceados.

2.5. Evaluación

Una vez entrenado un modelo de *machine learning* es importante establecer cuáles serán las métricas para evaluar su desempeño, ya que deberán ser suficientemente representativas para poder tomar decisiones o realizar conclusiones. Sin embargo cuando se trata de un problema con desbalance de datos hay que tener cuidado con las métricas elegidas ya que las más comunes, en problemas multi-clase en particular, suelen ser afectadas por el sesgo hacia las clases mayoritarias en las predicciones [26].

Actualmente, lo más común para evaluar modelos de clasificación multi-clase es emplear métricas a nivel micro (por clase) y macro (en promedio) para la precision que es la proporción de clasificaciones positivas correctas entre los casos predichos como positivos, el *recall* que es la proporción de clasificaciones positivas correctas entre los casos verdaderamente positivos y el F1-score que es el promedio armónico entre precision y *recall*.

Sea TP_i la cantidad de instancias clasificadas correctamente como pertenecientes a la clase i , FP_i aquellas mal clasificadas como pertenecientes a la clase i , C el total de clases, μ la notación para las métricas a nivel micro (por clase) y M a nivel macro. Las métricas arriba mencionadas se definen como:

$$Precision_{\mu i} = \frac{TP_i}{TP_i + FP_i}, \quad Precision_M = \frac{\sum_{i=1}^C Precision_{\mu i}}{C} \quad (2.18)$$

$$Recall_{\mu i} = \frac{TP_i}{TP_i + FN_i}, \quad Recall_M = \frac{\sum_{i=1}^C Recall_{\mu i}}{C} \quad (2.19)$$

$$F1\text{-score}_{\mu i} = 2 \cdot \frac{Precision_{\mu i} \cdot Recall_{\mu i}}{Precision_{\mu i} + Recall_{\mu i}}, \quad F1\text{-score}_M = \frac{\sum_{i=1}^C F1\text{-score}_{\mu i}}{C} \quad (2.20)$$

Otra métrica que se propone utilizar es la denominada Class Balance Accuracy (CBA) [27] la cual toma en cuenta la cantidad real de instancias en una clase y la cantidad de predicciones realizadas por el modelo hacia la misma la clase.

EL CBA podría interpretarse como una combinación entre las métricas de precision y recall, ya que se obtiene dividiendo la tasa de instancias correctamente clasificadas (TP) por el máximo entre la cantidad de falsos positivos (como la precision) y la cantidad de falsos negativos (como el recall). Ocupando las mismas notaciones anteriores, el CBA se calcula como:

$$CBA = \frac{1}{C} \sum_{i=1}^C \frac{TP_i}{TP_i + \max(FP_i, FN_i)} \quad (2.21)$$

Para comparar el desempeño entre dos modelos predictivos se suelen realizar repetidas pruebas con distintos grupos de entrenamiento y test, para luego analizar los valores promedios de las métricas seleccionadas junto a su respectiva desviación estándar. Debido a las desviaciones obtenidas el valor promedio no es suficiente para concluir si la diferencia entre el desempeño de dos modelos es significativa o es solo debido a una casualidad de los grupos de entrenamiento-test respectivos.

Por ello se implementan test estadísticos que, en base a la evidencia, permiten comprobar si la diferencia entre dos medias (el desempeño de dos modelos distintos) es real y significativa.

Una de estas pruebas estadísticas corresponde al test *t-student* el que se utiliza para evaluar si las distribuciones de dos grupos de muestras es similar o diferente. Llevado al contexto de modelos de *machine learning*, los grupos de muestras corresponden al desempeño de cada modelo al ser entrenado con distintos grupos de entrenamiento y evaluado en sus respectivos grupos de testeo.

Primero se establece lo que se conoce como la hipótesis nula (H_0) y la hipótesis alternativa (H_1). La hipótesis nula postula que no se evidencia diferencia entre el desempeño de los modelos a comparar. Por otro lado, la hipótesis alternativa postula que hay diferencia significativa entre el desempeño de los modelos. A continuación se resumen ambas hipótesis, donde μ_i es el promedio del grupo i de muestras para cierta métrica a evaluar:

$$\begin{aligned} H_0 : \mu_1 &= \mu_2 \\ H_1 : \mu_1 &\neq \mu_2. \end{aligned} \quad (2.22)$$

El procedimiento para comprobar o refutar las hipótesis establecidas comienza calculando las diferencias entre las muestras. Luego se obtiene el promedio de las diferencias \hat{x}_d con el cual se procede a calcular el *t-value* para $n - 1$ grados de libertad y el nivel de confianza α como:

$$t_{n-1,\alpha} = \frac{\hat{x}_d}{m}, \quad (2.23)$$

donde $S_{\hat{x}_d}$ se obtiene dividiendo la desviación estándar de las diferencias por la raíz cuadrada de la cantidad de muestras. Con el *t-value* se procede a obtener lo que se conoce como el *p-value* a partir de la distribución de probabilidad de *t-student*. Estableciendo un nivel de confianza del 95 %, para poder rechazar H_0 se requiere que el *p-value* sea menor al nivel de significancia $\alpha = 0.05$, indicando así que ambos desempeños son cuantitativamente diferentes.

El *p-value* representa la probabilidad de que la diferencia observada sea a causa de la suerte, por lo que un bajo *p-value* indicaría una mayor significancia estadística en la diferencia observada. Por lo tanto el nivel de significancia representa el límite de probabilidad por el cual se puede llegar a aceptar H_0 .

El test de *t-student* trabaja bajo ciertas suposiciones necesarias para que los resultados obtenidos sean validos, estas son que las muestras en estudio presenten una distribución normal, que las muestras provengan de fuentes independientes y que no se presente ningún valor atípico.

Debido al supuesto de que las muestras en estudio distribuyan de manera normal, es que es necesario implementar pruebas para verificar la distribución en los datos. Este tipo de test se conoce como *normality test* y son varias las formas por las que se puede comprobar la normalidad de una distribución, donde una de las más recomendadas para tamaños de muestra pequeños es el Shapiro Wilk test.

El Shapiro Wilk test funciona de manera similar al *t-student*, planteando una hipótesis nula (H_0) y una hipótesis alternativa (H_1), calculando luego un valor estadístico a partir de las muestras en observación, para obtener el *p-value*. Las hipótesis del test de Shapiro Wilk son:

$$\begin{aligned} H_0 &: \text{Los datos distribuyen normalmente} \\ H_1 &: \text{Los datos no distribuyen normalmente} \end{aligned} \quad (2.24)$$

Tal como con el *t-student*, se establece un nivel de confianza (p.ej. 95 %, $\alpha = 0.05$) para poder rechazar la hipótesis nula. Por lo que finalmente, en caso de que el *p-value* sea mayor a α , la hipótesis nula será aceptada y será válido utilizar el *t-student* para comparar grupos de muestras. En caso contrario, si el *p-value* del test de Shapiro Wilk es menor a α , sería necesario utilizar otro tipo de test estadístico para comparar los desempeños.

Una alternativa al test *t-student* es el *Permutation test* el cual es un test estadístico que no se basa en que las muestras siguen alguna distribución conocida, lo cual se conoce en estadística como un test no paramétrico.

El *Permutation test* propone hipótesis similares a las de la ecuación 2.22. Una vez planteadas las hipótesis, se calcula un valor estadístico inicial para comparar dos muestras distintas, este puede ser el mismo que el de la ecuación 2.23.

Una vez calculado el valor estadístico inicial, se procede a permutar entre si los grupos de muestras, de manera de calcular el mismo valor estadístico para todas las combinaciones posibles. Esta acción se realiza debido a que se quiere comprobar la hipótesis nula, que es que no hay una diferencia significativa entre las dos muestras, por lo que realizar permutaciones entre los grupos no debería influir significativamente en el valor estadístico.

Habiendo calculado los valores estadísticos de todas las posibles combinaciones de los grupos en observación, con estos valores se obtiene la distribución estadística de las muestras bajo la hipótesis nula. Utilizando esta distribución es posible calcular la probabilidad asociada a la hipótesis nula, por lo que analizando en que región de la distribución cae el valor estadístico inicial calculado es que se obtiene el *p-value*.

Finalmente se compara el *p-value* con el nivel de significancia establecido (p.ej. 95 %, $\alpha =$

0.05), para que así, en caso de que el valor de probabilidad obtenido sea menor, la hipótesis nula será rechazada, de forma que se comprueba estadísticamente que hay una diferencia significativa entre las dos muestras a comparar.

Capítulo 3

Metodología

Los datos a utilizar para el entrenamiento y evaluación de los modelos corresponden a los utilizados por ALeRCE en el paper de su clasificador de curvas de luz [4], estos constan de un total de 123.496 objetos astronómicos, con un total de 152 características obtenidas a partir de sus curvas de luz formadas con las alertas astronómicas del Zwicky Transient Facility (ZTF).

Además, se tiene un conjunto con dos etiquetas para cada objeto, la primera, *class_hierarchical*, identifica a que clase jerárquica pertenece el objeto (Periódica, Estocástica, Transiente) y en segundo lugar la etiqueta *class_original* que identifica a que subclase pertenece (dentro de la clase jerárquica). En la tabla 3.1 se resume la cantidad de instancias del conjunto de datos para cada clase jerárquica y sus subclases.

Tabla 3.1: Numero de instancias por clase jerárquica y sus subclases del conjunto de datos utilizado.

Clase jerárquica	Subclase	#
Periódicas (87.065)	E	37.901
	RRL	32.482
	LPV	14.076
	Periodic-Other	1.256
	DSCT	732
	CEP	618
Estocásticas (34.713)	QSO	26.168
	AGN	4.667
	YSO	1.740
	Blazar	1.267
	CV/Nova	871
Transientes (1.718)	SNIa	1272
	SNII	328
	SNIbc	94
	SLSN	24

Los clasificadores a entrenar son cuatro en total, uno correspondiente a la capa superior (Hierarchical) el cual se entrena con instancias provenientes de las tres clases jerárquicas, pero ocupando *class_hierarchical* como etiqueta. Los otros tres clasificadores corresponden a cada una de clases jerárquicas, es decir, un clasificador para las subclases provenientes de la clase Periódica, otro para las subclases de la clase Estocástica y otro para las subclases de Transiente, cada uno de estos es entrenado de manera independiente al clasificador superior y entre si, utilizando exclusivamente instancias pertenecientes a su respectiva clase jerárquica y utilizando *class_original* como etiquetas.

La unión de niveles en el clasificador se realiza multiplicando los puntajes obtenidos en el nivel superior para cada clase jerárquica, por los puntajes de sus correspondientes subclases obtenidos de los clasificadores del nivel inferior. Decidiendo finalmente por la subclase con la que se obtenga un mayor puntaje total.

Usualmente para el entrenamiento de los modelos predictivos se realiza una partición del total de datos que se tiene a disposición para obtener un grupo de entrenamiento y otro para testeo, típicamente en una razón cercana al 80 % y 20 % respectivamente. De esta forma se reserva una parte de los datos que el modelo no conoce para poder evaluar que tan bien aprendió a clasificar las distintas clases del problema al mismo tiempo que se toma en cuenta su capacidad de generalización.

Si bien la partición del total de datos en grupos de entrenamiento y test puede ser aleatoria, nunca se sabe a ciencia cierta si aquella fue la mejor separación posible en cuanto a similitud de datos entre grupos o bien si ambos grupos representan bien el contexto del problema. Esto a causa de que existirán instancias más difíciles o fáciles de clasificar, las cuales dependiendo del grupo caigan en que puede variar bastante el desempeño del modelo entrenado, teniendo como consecuencia distintos escenarios con mejores o peores resultados.

A causa de lo anterior es necesario adoptar una metodología con la cual se pueda medir el real desempeño de un modelo o técnica de manera objetiva y generalizada, sin sufrir del efecto que pueda provocar la elección de una partición de datos en particular. La solución propuesta a este problema es ocupar el procedimiento de entrenamiento y evaluación de Validación Cruzada Anidada (*Nested Cross-Validation*).

El procedimiento de Validación Cruzada Anidada se basa en separar los datos a disposición de manera iterativa en dos niveles distintos, asegurando que solo se evalúe el desempeño del modelo con datos que sean desconocidos para éste. En el primer nivel, entendido como loop exterior, se obtienen distintas configuraciones de grupos de entrenamiento y test, con las que se entrenaran y evalúan modelos. Luego se promedian los resultados obtenidos con las distintas métricas seleccionadas, tomando en cuenta además la desviación estándar para futuras comparaciones.

Para el segundo nivel, entendido como loop interior, se obtienen distintas configuraciones de grupos de entrenamiento y validación, las cuales son originadas a partir de los datos de entrenamiento de cada configuración del loop exterior. El objetivo del loop interior en este caso, es obtener la mejor configuración de hiperparámetros del modelo en observación para que luego en el loop exterior se proceda a evaluar con la configuración seleccionada. El

procedimiento para seleccionar la mejor combinación de hiperparámetros es que para cada combinación se entrena repetidas veces el modelo en observación con los distintos grupos de entrenamiento y se promedia el desempeño de éste con los respectivos grupos de validación. Al final se seleccionan la combinación que obtuvo el mejor resultado.

Para obtener las distintas configuraciones de grupos de entrenamiento y test del loop exterior se utiliza el algoritmo *Stratified Shuffle Split* [28] el cual toma muestras aleatorias (sin repetir) del total de datos para formar el grupo de test, dejando el resto de los datos para el entrenamiento, asegurando además que cada grupo tenga una distribución de clases similar a la distribución en el grupo original de datos.

La figura 3.1 ilustra este procedimiento en el nivel definido como *Outer loop*. Para este caso se seleccionó una partición de datos en una proporción de 80 %-20 % para grupos de entrenamiento y test respectivamente, además se realizan 10 iteraciones en este loop.

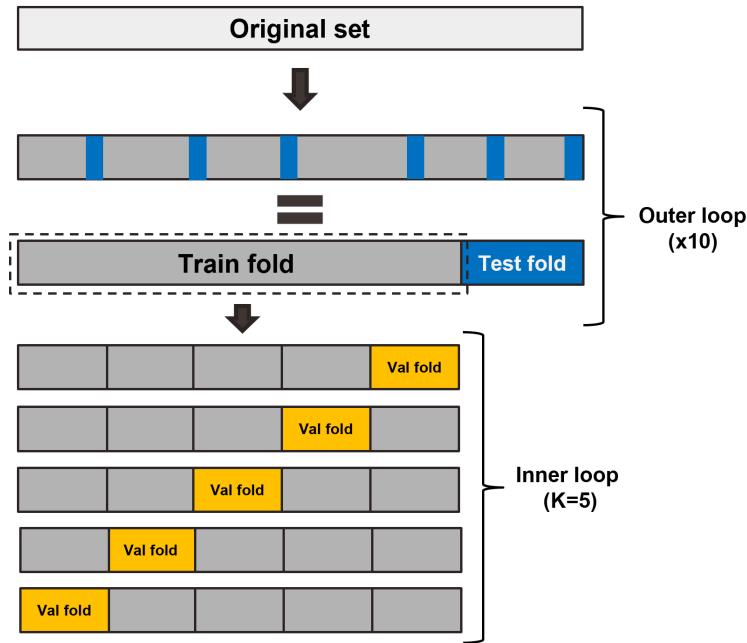


Figura 3.1: Procedimiento de Validación Cruzada Anidada.

Para el loop interno, se utilizó el procedimiento de Stratified Kfolds en el que se divide el total de datos (en este caso el grupo de entrenamiento del loop exterior) en K distintos grupos que mantienen la misma proporción de clases que la del conjunto total de entrenamiento del que provienen. Luego para cada iteración se va turnando cuál de los K grupos se ocupará para validación y el resto para entrenamiento. En la figura 3.1 se ejemplifica este procedimiento en el *Inner loop* ocupando $K = 5$.

Una vez terminado el procedimiento de Nested Cross-Validation se tienen los modelos ya entrenados y evaluados con las métricas escogidas. A los modelos entrenados se les establece una condición mínima para que sean aceptables y se profundice en el análisis de su desempeño. Esta condición es que el recall mínimo obtenido por clase sea superior a un valor umbral que sea acorde al clasificador en cuestión (Periódico, Estocástico o Transiente) y que será establecido en base a los resultados de las pruebas con las primeras técnicas de balanceo.

Solo aquellas combinaciones de modelo y técnicas que cumplan el requerimiento mínimo se procederá a analizar su matriz de confusión, la cual es obtenida a partir tomar la mediana de cada elemento entre las matrices de confusión individuales resultantes de los modelos entrenados en el loop exterior. Además se reporta la diferencia de resultados con el percentil 90 y 10 (el peor y mejor resultado) para cada elemento de la matriz.

Finalmente, en caso que las diferencias entre los modelos entrenados no sean claras, se propone evaluar sus desempeños a través de los test estadísticos mencionados en la sección anterior. Estos tests corresponden al *t-student* y al *Permutation test*. La idea de utilizar estos tests es que estos reflejan si la diferencia en el desempeño de dos modelos podría considerarse significativa.

El test *t-student* funciona bajo el supuesto de que las muestras distribuyen de manera normal, por lo que si no cumple con dicha condición, esto provocaría que los resultados obtenidos no sean válidos. Por ello antes de su realización se comprobará la normalidad de los datos por medio del test de Shapiro Wilk. En caso de que no se compruebe la normalidad de los datos se realizará el *Permutation test*, ya que al ser un test no paramétrico, no necesita condiciones en la distribución de los datos para poder utilizarse.

Debido a que la metodología de trabajo propuesta se basa en entrenar y evaluar repetidas veces distintos modelos y técnicas con los mismos datos, el comportamiento de los modelos con los datos de test puede llegar a ser conocido por el usuario, provocando una pérdida de generalización a causa de tomar decisiones en el entrenamiento en base a los resultados esperados.

Para hacer frente al problema de pérdida de una posible generalización se suele reservar un conjunto de datos que no van a ser utilizados en el proceso de experimentación, para que así en una instancia final sean utilizados solo como test final para evaluar el desempeño del modelo con datos totalmente desconocidos.

Por ello se propone que luego de la comparación de desempeño entre modelos, se realice un test final utilizando aquellas curvas de luz recibidas por ALeRCE desde el sondeo ZTF con fecha posterior a la construcción de la base de datos de ALeRCE [4].

Esto permitiría medir de manera realmente objetiva el desempeño de los modelos, evaluar el desempeño de las mejores técnicas de balance en conjunto con XGBoost y hacer comparaciones con *Balanced Random Forest*, modelo actualmente utilizado por ALeRCE.

En la tabla 3.2 aparece el conteo de instancias por clase jerárquica y subclase de las nuevas curvas de luz de ALeRCE que conforman el conjunto de test final. Éstas están representadas en la parte superior de las barras de cada subclase de la figura 3.2, en donde se ilustra la distribución de instancias por subclase, mostrando la totalidad de curvas de luz a disposición.

Tabla 3.2: Conteo de instancias por clase jerárquica y sus subclases de nuevas curvas de luz de ALeRCE.

Clase jerarquica	Subclase	#
Periódicas (44.994)	E	13.061
	RRL	5.353
	LPV	25.273
	Periodic-Other	880
	DSCT	273
	CEP	154
Estocásticas (18.802)	QSO	15.379
	AGN	1.434
	YSO	1.545
	Blazar	290
	CV/Nova	154
Transientes (1.789)	SNIa	1.385
	SNII	263
	SNIbc	98
	SLSN	43

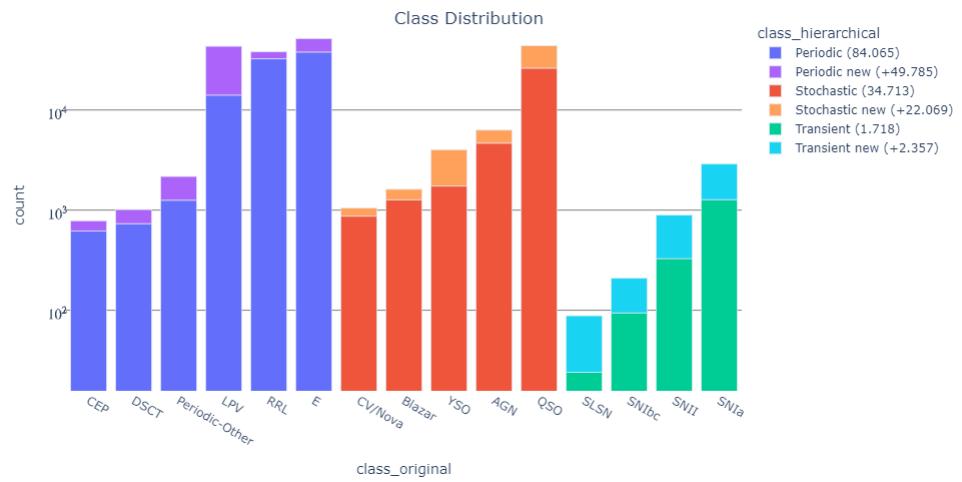


Figura 3.2: Distribución de instancias por clase jerárquica y subclase de nuevas curvas de luz de ALeRCE.

Capítulo 4

Experimentos y resultados

En el presente capítulo se presentan los experimentos realizados, los modelos entrenados y las técnicas de balanceo utilizadas junto con sus resultados y posterior análisis.

Las métricas utilizadas son las mencionadas en el 2. Los valores reportados corresponden al promedio de los resultados de las iteraciones del loop exterior del procedimiento de Nested Cross-Validation mencionado en el capítulo 3, además se incluye la desviación estándar de estos resultados.

Los resultados corresponden al clasificador de la capa superior (denominado Hierarchical), los clasificadores individuales de la capa inferior (Periódico, Estocástico y Transiente) y además a la unión de ambas capas (denominado LCC por *Light Curve Classifier*).

Los códigos con los cuales se entrenaron y evaluaron los modelos se encuentran disponibles en un repositorio de Github [29].

4.1. Replica del clasificador de curvas de luz de ALeRCE

En la tabla 4.1 se muestran los resultados reportados por ALeRCE en el paper [4] del clasificador de luz al utilizar el algoritmo Balanced Random Forest (BRF) [6] para la capa superior (Hierarchical) e inferior (LCC) del clasificador.

Tabla 4.1: Valores reportados por ALerCE en [4] en métricas de Precision, Recall y F1-score para el desempeño del nivel superior y la unión de este con el nivel inferior para el clasificador de curvas de luz con BRF.

Clasificador	Precision _M	Recall _M	F1- score _M
Hierarchical	0.96±0.01	0.99±0.01	0.97±0.01
LCC	0.57±0.01	0.76±0.02	0.59±0.01

Con fines de poder comparar directamente los resultados de la experimentación con XG-Boost y técnicas de balanceo, se replicó el modelo del clasificador de curvas de luz de ALeRCE ocupando el algoritmo de Balanced Random Forest (BRF) [6].

En la tabla 4.2 se muestran los resultados promedio de Precision, Recall, F1-score y minimo Recall en el grupo de test junto con su desviación estándar de los 10 modelos entrenados en el loop exterior del procedimiento de Nested Cross-Validation

Tabla 4.2: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador jerárquico, para los clasificadores de objetos periódicos, estocásticos y transientes, y su unión (LCC) ocupando BRF.

Clasificador	Precision_M	Recall_M	F1-score_M	min Recall
Hierarchical	0.96 ± 0.00	0.99 ± 0.00	0.97 ± 0.00	0.97 ± 0.00
Periódico	0.56 ± 0.00	0.82 ± 0.01	0.58 ± 0.00	0.72 ± 0.02
Estocástico	0.77 ± 0.01	0.88 ± 0.01	0.81 ± 0.01	0.73 ± 0.02
Transiente	0.52 ± 0.04	0.66 ± 0.05	0.51 ± 0.04	0.49 ± 0.07
LCC	0.57 ± 0.01	0.76 ± 0.02	0.60 ± 0.01	0.47 ± 0.07

Se puede ver que los valores en las métricas de Precision_M , Recall_M y F1-score_M son prácticamente idénticos a los obtenidos por ALeRCE en [4], los cuales se muestran en la tabla 4.1

La obtención de resultados similares a los originales es relevante debido a que uno de los objetivos del presente trabajo es comparar el rendimiento del modelo XGBoost con el obtenido del clasificador de ALeRCE, por lo que es necesario contar con una propia versión de este último clasificador que sea un reflejo del original.

Comparando entre si el desempeño de los clasificadores de la capa inferior, se puede ver que el correspondiente a las clases Estocásticas supera al resto de los clasificadores en las métricas Precision y F1-score, teniendo diferencias significativas exceptuando el Recall tanto en su versión macro como mínimo por clase, en los cuales el clasificador de Periódicas obtuvo resultados similares.

Por otro lado, el clasificador de clases Transientes, obtuvo el peor rendimiento en todas las métricas en evaluación, destacando el mínimo recall por clase obtenido con un valor de 0.49.

En la figura 4.1 se muestran las matrices de confusión de los cuatro clasificadores del clasificador de curvas de luz entrenados con el modelo de BRF. En la figura 4.2 se muestra la matriz de confusión resultante de la capa inferior del clasificador con las 15 subclases.

Los resultados obtenidos en la tabla 4.2 se condicen con lo obtenido en sus respectivas matrices de confusión. En la matriz de confusión del clasificador jerárquico de la figura 4.1.a se puede ver una alta tasa de verdaderos positivos (TP) para cada una de las clases a separar, presentando bajas confusiones al momento de clasificar curvas de luz estocásticas, con un 3 % de las cuales son predichas como falsas periódicas.

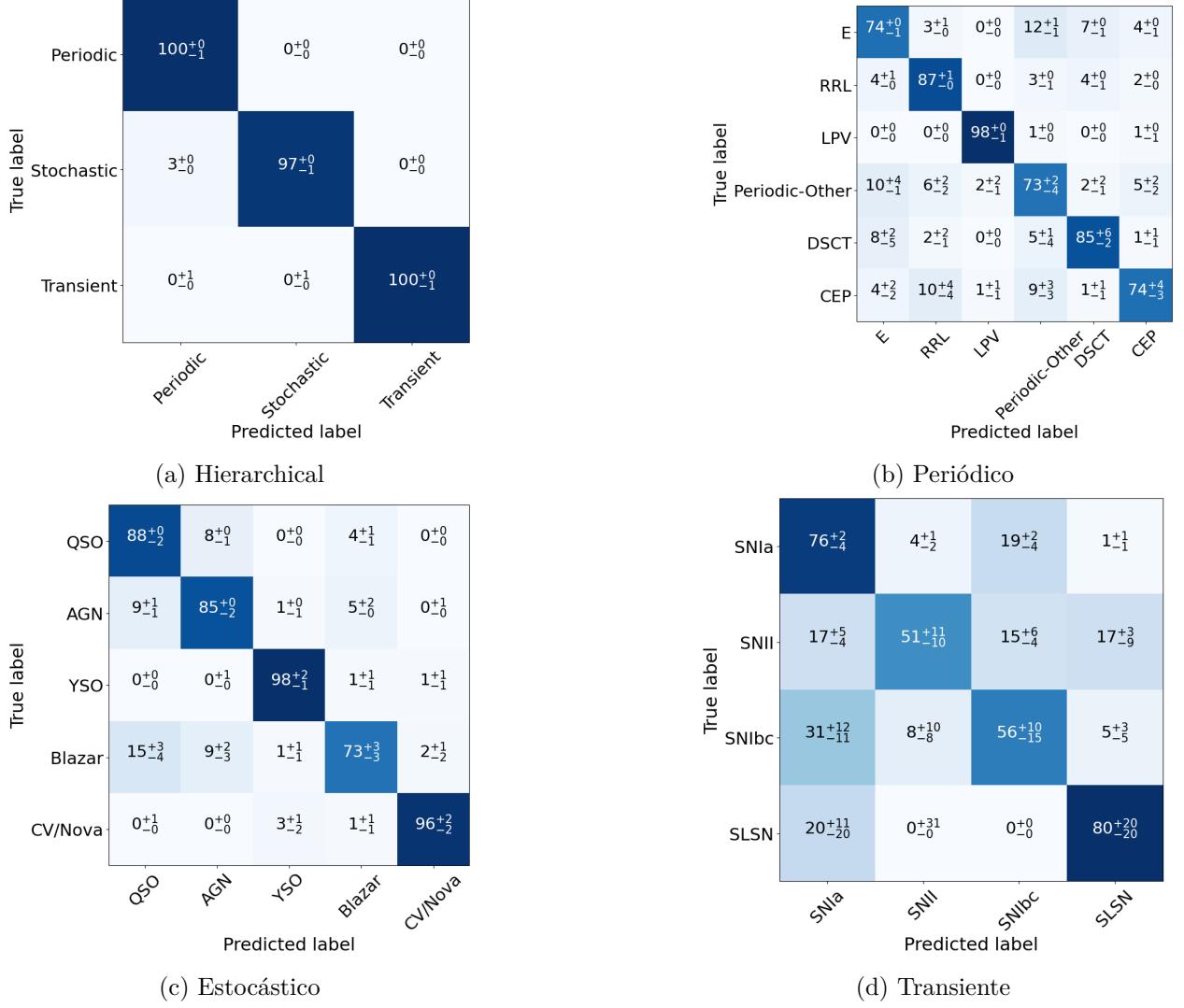


Figura 4.1: Matrices de confusión de los cuatro clasificadores que componen el clasificador de curvas de luz basado en *Balanced Random Forest*.

La confusión presente en el clasificador jerárquico entre los objetos estocásticos y periódicos se ve reflejada en la matriz de confusión de la figura 4.2 de la salida de la capa inferior. Los objetos YSO y CV/Nova son los más afectados presentando confusiones con objetos del tipo periódico. Este es un ejemplo del principal problema que presentan los clasificadores de estructura jerárquica, donde una mala clasificación en los niveles superiores arrastra a errores en la salida final.

	SNIA	SNI	SNIbc	SLSN	QSO	AGN	YSO	Blazar	CV/Nova	E	RRL	LPV	Periodic-Other	DSCT	CEP	
True label	76 ⁺¹ ₋₄	4 ⁺¹ ₋₂	19 ⁺² ₋₄	1 ⁺¹ ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀						
SNIA	76 ⁺¹ ₋₄	4 ⁺¹ ₋₂	19 ⁺² ₋₄	1 ⁺¹ ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀						
SNI	17 ⁺⁵ ₋₄	52 ⁺¹¹ ₋₁₀	15 ⁺⁶ ₋₃	16 ⁺⁵ ₋₇	0 ⁺⁰ ₋₀	0 ⁺³ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀					
SNIbc	32 ⁺¹³ ₋₁₁	5 ⁺¹⁴ ₋₅	58 ⁺¹⁰ ₋₁₆	5 ⁺³ ₋₅	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺³ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀						
SLSN	22 ⁺¹² ₋₂₂	0 ⁺³⁴ ₋₀	0 ⁺⁰ ₋₀	78 ⁺²² ₋₃₄	0 ⁺⁰ ₋₀	0 ⁺¹² ₋₀	0 ⁺⁰ ₋₀	0 ⁺²² ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
QSO	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	88 ⁺⁰ ₋₂	8 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀	4 ⁺¹ ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
AGN	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	9 ⁺¹ ₋₁	85 ⁺¹ ₋₁	1 ⁺⁰ ₋₁	5 ⁺¹ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
YSO	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	78 ⁺¹ ₋₄	1 ⁺¹ ₋₁	0 ⁺¹ ₋₀	1 ⁺¹ ₋₁	1 ⁺¹ ₋₀	5 ⁺¹ ₋₂	11 ⁺² ₋₁	1 ⁺⁰ ₋₁	3 ⁺² ₋₁	
Blazar	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	16 ⁺² ₋₅	9 ⁺² ₋₂	1 ⁺¹ ₋₁	73 ⁺³ ₋₂	2 ⁺² ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
CV/Nova	4 ⁺² ₋₂	1 ⁺¹ ₋₁	2 ⁺¹ ₋₂	0 ⁺¹ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	1 ⁺² ₋₁	1 ⁺¹ ₋₁	68 ⁺³ ₋₇	3 ⁺² ₋₂	11 ⁺⁴ ₋₃	1 ⁺¹ ₋₁	2 ⁺² ₋₁	4 ⁺² ₋₁	4 ⁺¹ ₋₂	
E	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₀	74 ⁺¹ ₋₁	3 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	12 ⁺¹ ₋₁	6 ⁺¹ ₋₀	4 ⁺⁰ ₋₁	
RRL	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	4 ⁺¹ ₋₀	88 ⁺¹ ₋₁	0 ⁺⁰ ₋₀	2 ⁺¹ ₋₀	4 ⁺⁰ ₋₁	2 ⁺⁰ ₋₀	
LPV	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	1 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	97 ⁺¹ ₋₀	1 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₁		
Periodic-Other	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	2 ⁺⁰ ₋₂	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₁	10 ⁺³ ₋₁	6 ⁺² ₋₂	2 ⁺¹ ₋₁	72 ⁺⁴ ₋₃	3 ⁺¹ ₋₂	5 ⁺² ₋₂	
DSCT	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	8 ⁺² ₋₅	2 ⁺² ₋₁	0 ⁺⁰ ₋₀	5 ⁺¹ ₋₄	85 ⁺⁶ ₋₂	1 ⁺¹ ₋₁	
CEP	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	1 ⁺² ₋₁	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	4 ⁺² ₋₂	10 ⁺⁴ ₋₄	1 ⁺⁰ ₋₁	8 ⁺⁴ ₋₂	1 ⁺¹ ₋₁	74 ⁺⁴ ₋₄	
Predicted label	SNIA	SNI	SNIbc	SLSN	QSO	AGN	YSO	Blazar	CV/Nova	E	RRL	LPV	Periodic-Other	DSCT	CEP	

Figura 4.2: Matriz de confusión de la capa inferior del clasificador de curvas de luz basado en BRF.

En la matriz de confusión del clasificador Periódico, se puede observar que las clases E, Periodic-Other y CEP tienen menor tasa de TP, lo cual es interesante teniendo en cuenta que son aquellas que podrían considerarse como la clase mayoritaria, intermedia y minoritaria respectivamente. Por otro lado las mayores confusiones entre clases se dan para los casos en que se clasifican erróneamente objetos como pertenecientes a E o RRL (las de mayor proporción) y para cuando se predicen curvas de la clase E como Periodic-Other.

En cuanto al clasificador Estocástico, las clases YSO y CV/Nova tienen un mayor desempeño, las cuales en base a la distribución de instancias por clase se podrían considerar como una intermedia y minoritaria respectivamente. La clase Blazar por otro lado, aun cuando tiene una mayor cantidad de instancias que la clase minoritaria, es la que presenta peor desempeño, con porcentajes de 15 % y 9 % de instancias mal clasificadas como pertenecientes a las clases mayoritarias, QSO y AGN respectivamente.

Como era de esperar el clasificador Transiente obtuvo un bajo desempeño en cuanto a las métricas de evaluación. En la matriz de confusión de la figura 4.1.d se pueden observar confusiones de hasta un 31 % entre las clases. Si bien existe un sesgo hacia la clase mayoritaria

(SNIa) al momento de predecir el resto de las clases, es la clase SLSN la que presenta una mayor tasa de TP en su valor mediano.

A pesar del buen desempeño en la clase minoritaria (SLSN) del grupo Transiente, hay que tener en cuenta que esta clase presenta la mayor diferencia en sus percentiles 90 y 10, lo cual es esperable ya que en el dataset utilizado solo se tienen 24 instancias de SLSN, por lo que ciertas instancias pueden afectar bastante al resultado final dependiendo de a que grupo sean asignadas (entrenamiento o test).

4.2. Entrenamiento de XGBoost con selección de hiperparámetros

Para poder dimensionar el efecto que tienen las de desbalance de datos, se propone entrenar en una primera instancia el modelo XGBoost sin ninguna de estas técnicas. Para esto se sigue el procedimiento de *Nested Cross Validation* incluyendo la selección de mejores hiperparámetros en los loops internos del procedimiento.

Los hiperparámetros del modelo XGBoost a seleccionar y el rango de estos son:

- *learning_rate*, [0.09, 0.6]
- *max_depth*, [3,35]
- *min_child_weight*, [2, 6]
- *gamma*, [0, 7]
- *reg_alpha*, [0, 15]
- *reg_lambda*, [0, 0.8]
- *colsample_bytree*, [0.5, 0.9]
- *subsample*, [0.5, 0.9]

La explicación de cada uno de estos hiperparámetros se encuentra en la sección 8.1.2.1 del anexo.

En los futuros experimentos con distintas técnicas de balanceo, estos mismos hiperparámetros pasaran nuevamente por el procedimiento de selección para obtener su mejor combinación de valores.

Al ser varios los hiperparámetros a seleccionar se deberían realizar muchas pruebas con distintos valores para cada uno de estos y sus distintas combinaciones para así obtener la mejor combinación de valores de hiperparámetros. El problema de esto es que computacionalmente es muy costoso y poco escalable a la hora de entrenar repetidas veces el modelo XGBoost con distintas técnicas de balance.

A raíz de este problema es que se optó por ocupar técnicas que implementan optimización bayesiana para crear un espacio de densidad con los valores de los parámetros a obtener. A grandes rasgos, estas técnicas crean un modelo probabilístico sustituto de la función objetivo del modelo predictivo a optimizar, de esta forma se utiliza el modelo sustituto para seleccionar la combinación de parámetros más prometedora que luego será probada en la función real del modelo predictivo en cuestión. En base a los resultados obtenidos con el modelo real se actualiza el espacio de parámetros y el modelo sustituto para así obtener la próxima combinación de parámetros a evaluar. Para realizar la búsqueda de hiperparámetros se utilizo la librería de python Hyperopt [30].

Cabe mencionar que XGBoost para problemas de clasificación multiclasa utiliza *Cross Entropy* como función de pérdida. Además, se optó por el siguiente criterio de detención del entrenamiento: si luego de 50 iteraciones no se obtiene una mejora en el Recall_M , el entrenamiento se detiene y se utilizará el modelo obtenido en la iteración en la cual se obtuvo el mejor desempeño.

Los resultados obtenidos con el modelo XGBoost con selección de hiperparámetros y sin técnicas de balance se muestran en la tabla 4.3. Las matrices de confusión de los cuatro clasificadores entrenados son las de la figura 4.3.

Tabla 4.3: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador jerárquico, para los clasificadores de objetos periódicos, estocásticos y transientes, y su unión (LCC) con modelo XGB.

Clasificador	Precision_M	Recall_M	F1-score_M	min Recall
Hierarchical	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.98 ± 0.00
Periódico	0.91 ± 0.01	0.74 ± 0.01	0.80 ± 0.01	0.33 ± 0.03
Estocástico	0.92 ± 0.00	0.85 ± 0.01	0.88 ± 0.01	0.60 ± 0.03
Transiente	0.74 ± 0.10	0.52 ± 0.04	0.57 ± 0.05	0.11 ± 0.07
LCC	0.85 ± 0.03	0.69 ± 0.02	0.74 ± 0.02	0.09 ± 0.07

En base a los resultados obtenidos se pueden observar claras diferencias de desempeño entre los algoritmos XGBoost y BRF al ser implementados para el clasificador de curvas de luz.

Respecto al clasificador del nivel superior, XGBoost tiene un mejor desempeño que BRF en cuanto a separar las clases superiores, teniendo valores casi perfectos en las métricas de evaluación de la tabla 4.3. Al comparar la matriz de confusión de la figura 4.3.a con la obtenida con el modelo de BRF en la figura 4.1.a se puede observar que disminuyó la confusión entre las clases Estocásticas y Periódicas.

Sin embargo, está clara mejora en el desempeño de los clasificadores del nivel superior no se repite para los clasificadores del nivel inferior. Al comparar estos clasificadores con los de sus pares con el modelo de BRF, se tiene un aumento notable en los valores de Precision_M , una disminución en Recall_M y min Recall, y un aumento moderado en F1-score_M .

Al observar el escenario completo de predicción por medio de las matrices de confusión de

las figuras 4.3 y 4.4 se puede evidenciar el efecto del desbalance en la clasificación de las distintas clases.

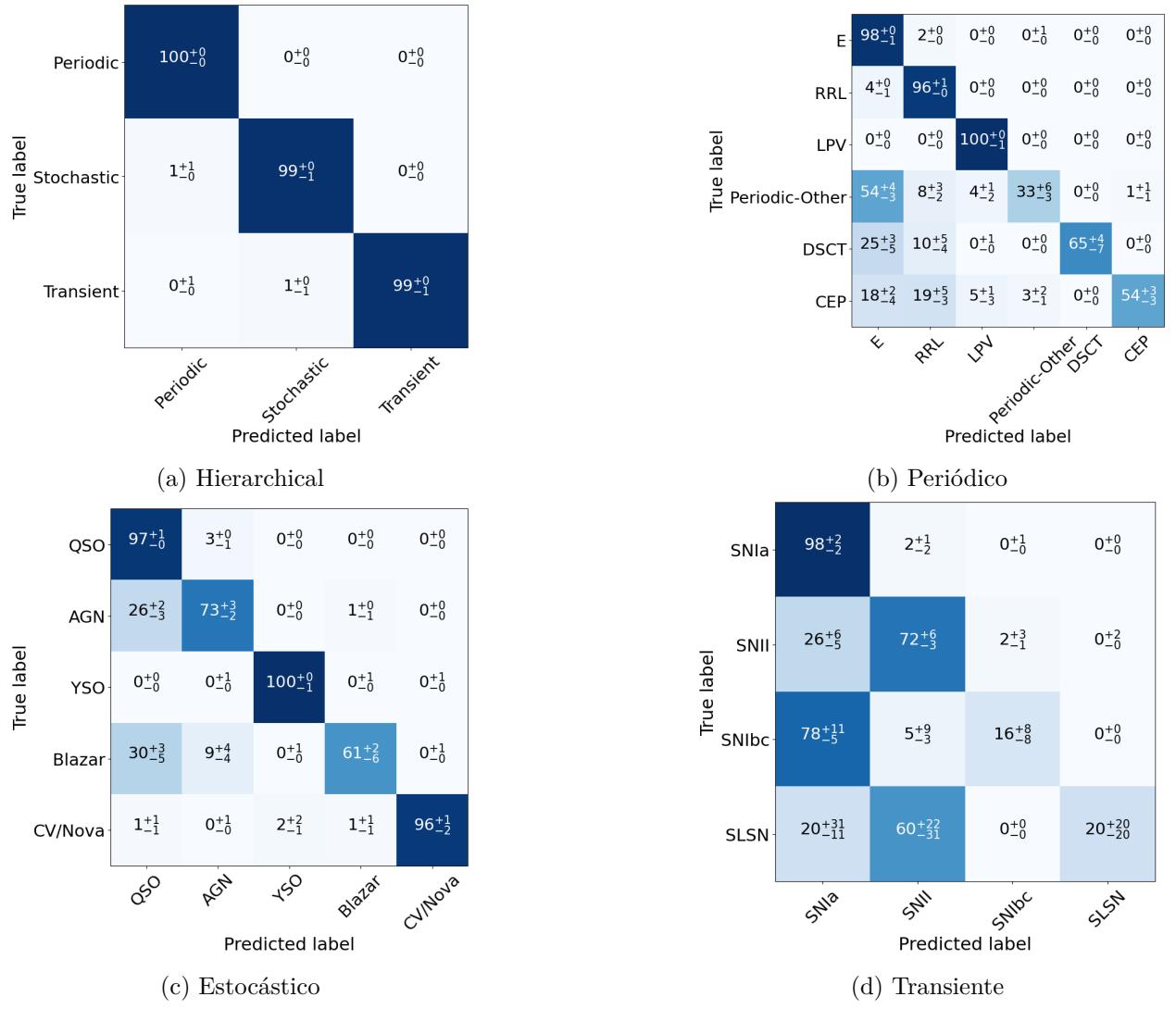


Figura 4.3: Matrices de confusión de los cuatro clasificadores del clasificador de curvas de luz ocupando XGBoost.

	SNIa	SNII	SNIbc	SLSN	QSO	AGN	YSO	Blazar	CV/Nova	E	RRL	LPV	Periodic-Other	DSCT	CEP	
True label	98 ⁺¹ ₋₂	2 ⁺¹ ₋₂	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	25 ⁺⁷ ₋₄	72 ⁺⁷ ₋₄	2 ⁺³ ₋₁	0 ⁺² ₋₀	0 ⁺¹ ₋₀	1 ⁺² ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺² ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	78 ⁺¹³ ₋₄	5 ⁺⁹ ₋₃	17 ⁺⁸ ₋₁₂	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	25 ⁺³⁹ ₋₁₄	50 ⁺³⁹ ₋₂₅	0 ⁺⁰ ₋₀	25 ⁺²⁵ ₋₂₅	0 ⁺¹⁴ ₋₀	0 ⁺²⁵ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹⁴ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	QSO	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	97 ⁺¹ ₋₀	3 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀				
	AGN	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	26 ⁺² ₋₃	73 ⁺³ ₋₂	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	YSO	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	84 ⁺³ ₋₂	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	6 ⁺² ₋₂	1 ⁺¹ ₋₀	5 ⁺¹ ₋₂	2 ⁺¹ ₋₁	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₁
	Blazar	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	30 ⁺³ ₋₅	9 ⁺³ ₋₄	0 ⁺⁰ ₋₀	61 ⁺¹ ₋₆	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	CV/Nova	2 ⁺¹ ₋₁	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	1 ⁺¹ ₋₁	0 ⁺¹ ₋₀	1 ⁺¹ ₋₁	1 ⁺⁰ ₋₁	74 ⁺⁴ ₋₈	7 ⁺³ ₋₂	13 ⁺⁷ ₋₂	1 ⁺⁰ ₋₁	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	E	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	98 ⁺¹ ₋₀	2 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	RRL	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	4 ⁺⁰ ₋₁	96 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	LPV	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	100 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	Periodic-Other	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	2 ⁺¹ ₋₂	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	53 ⁺⁵ ₋₃	8 ⁺³ ₋₂	3 ⁺¹ ₋₁	33 ⁺⁶ ₋₃	0 ⁺⁰ ₋₀	1 ⁺¹ ₋₁
	DSCT	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	24 ⁺⁴ ₋₄	10 ⁺⁵ ₋₄	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	65 ⁺⁴ ₋₇	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
	CEP	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	1 ⁺¹ ₋₁	0 ⁺¹ ₋₀	18 ⁺³ ₋₄	19 ⁺⁵ ₋₃	4 ⁺³ ₋₂	4 ⁺² ₋₂	0 ⁺⁰ ₋₀	54 ⁺³ ₋₂	0 ⁺⁰ ₋₀
Predicted label	SNIa	SNII	SNIbc	SLSN	QSO	AGN	YSO	Blazar	CV/Nova	E	RRL	LPV	Periodic-other	DSCT	CEP	

Figura 4.4: Matriz de confusión de la capa inferior del clasificador de curvas de luz con XGBoost.

Para el clasificador Periódico, en la figura 4.3.b se obtuvo una tasa mucho mayor de TP en las tres clases de mayor proporción, en comparación a las otras tres clases minoritarias. Esta situación fue distinta para BRF donde además de obtener mayores tasas de aciertos en las clases minoritarias, también fue menor la cantidad de predicciones erróneas que se confunden con la clase mayoritaria E.

Similarmente el clasificador Transiente presenta altas tasas de TP para las dos clases de mayor proporción SNIa y SNII pero muy bajas para las minoritarias correspondientes a SNIbc y SLSN. Esta situación no se da con el modelo BRF donde la clase minoritaria SLSN a pesar de tener baja proporción se obtiene una buena tasa de aciertos en la predicción.

Como se observa en la figura 4.1.c el clasificador Estocástico presenta resultados similares a los obtenidos con BRF en la figura 4.1.c donde la clase Blazar obtiene la menor tasa de TP. Pero se da el caso que con XGBoost es menor la cantidad de aciertos para dicha clase, además se observa una mayor confusión en instancias de la clase AGN predichas como QSO (clases mayoritarias).

Mediante el análisis de resultados de las métricas mostradas en la tabla 4.3 y las matrices de confusión de la figura 4.3 y comparando las diferencias con lo obtenido con el modelo de BRF se pueden sacar ciertas ideas sobre el efecto de este escenario desbalanceado.

La alta Precision_M junto con la baja Recall_M es un reflejo del sesgo del modelo XGBoost hacia las clases mayoritarias. Esto debido a que precision es una métrica muy sensible a las tendencias en las predicciones de un modelo.

Como las clases mayoritarias se llevan gran parte de las predicciones, se le está sumando a la capacidad del modelo a distinguir correctamente estas clases la casualidad o suerte de haber realizado una buena predicción, resultando en una alta tasa de aciertos. En cuanto precision, ya que proporcionalmente las instancias de las clases minoritarias mal predichas son pocas, las clases mayoritarias obtendrán buenos resultados en esta métrica.

El problema con las clases minoritarias, es que las instancias mal predichas, tienen una proporción significativa en estas, resultando en un bajo recall (reflejando el mal desempeño del modelo). Esto no sucede con precision, pues al no ser predichas como minoritarias no son tomadas en cuenta, obteniendo el modelo de igual forma una buena precision, pues lo poco que se predice como perteneciente a la clase minoritaria es correcto (e.g. casos fáciles de distinguir con respecto a las otras clases).

En cambio como el recall, evalua qué proporción del total de instancias por clase se predice correctamente, no sufre el efecto del desbalance y su valor refleja de manera directa cuando una clase (usualmente minoritaria) es dejada de lado en las predicciones.

Esto explicaría porqué los clasificadores Periódico y Transiente con XGBoost, que evidencian claros efectos de desbalance en la clasificación, tuvieron notables aumentos de Precision_M pero disminuciones en Recall_M . Esta situación no tan marcada con el clasificador Estocástico que tal como se dijo, sufrió efectos de desbalances, pero en menor grado, resultando en una leve baja de Recall_M pero con un aumento del Precision_M .

La métrica F1-score_M al ser una combinación de Precision y Recall, está sujetas tanto a la sensibilidad al desbalance de Precision_M como a la insensibilidad de Recall_M , sin embargo como Precision suele aumentar en mayor magnitud que la disminución de la Recall, se obtiene un aumento moderado en F1-score.

Es recomendable poner especial atención a los valores de Recall tanto a nivel macro como el mínimo entre las clases para evaluar el desempeño de un modelo en un escenario desbalanceado. Esto no significa que se debe dejar de lado el resto de las métricas, en particular Precision, esta métrica se debe analizar en conjunto con Recall de manera crítica y detallada, pues su aumento puede reflejar tanto una mejora en el desempeño o bien que se intensificó el sesgo hacia las clases mayoritarias.

En base a las comparaciones realizadas de los resultados obtenidos con BRF y XGBoost, se identifica la necesidad de implementar técnicas de balance de datos que complementen XGBoost, ya que cuando se entrena por si solo presenta una tendencia en sus predicciones hacia las clases mayoritarias.

4.3. Entrenamiento de XGBoost en conjunto con primeras técnicas de balance de datos

En una primera fase experimental se probaron dos métricas de evaluación distintas en el entrenamiento de XGBoost para definir el criterio de detención del algoritmo. Las métricas a probar son las de Macro-Recall ($Recall_M$) y Class Balance Accuracy (CBA) debido a que, según la literatura, éstas no son sensibles al desbalance de clases, por lo que serían confiables a la hora de evaluar un modelo en escenarios desbalanceados.

Además de probar distintas métricas, se experimentó con dos métodos de balance de datos con distinto enfoque. El primero corresponde a un método a nivel algoritmo, en particular del tipo *Cost Sensitive Learning*, donde se le asigna un ponderador a la función de pérdida, de forma de asignar un mayor o menor peso a la clasificación errónea en base al tipo de instancia que se está evaluando. La elección de los pesos se hizo acorde a la cantidad de instancias por clase, teniendo un mayor peso las de menor proporción.

La asignación de pesos para una instancia perteneciente a la clase i se realiza en base a la ecuación:

$$class_weight_i = \frac{total_samples}{n_classes \cdot n_samples_i}, \quad (4.1)$$

donde $n_classes$ es el número de clases y $n_samples_i$ el número de instancias de la clase i . Esta asignación se hizo tomando en cuenta solo las instancias del grupo de entrenamiento para cada uno de los 4 clasificadores.

La segunda técnica de balance ocupada corresponde a una combinación de *Random Undersampling* (RUS) y *Random Oversampling* (ROS), las cuales son técnicas a nivel de datos con las cuales se disminuye o aumenta la proporción de instancias para ciertas clases, por medio de la selección de instancias de manera aleatoria.

Para simplificar la notación a la hora de exponer los resultados, se definieron nombres para ambos métodos de balance, denominando “Weight” a la técnica basada en *Cost Sensitive Learning* y “Sampling” a la combinación de RUS y ROS.

Los resultados de la tabla 4.4 corresponden a los valores promedio obtenidos junto a su desviación estándar de los resultados en el grupo de test de los modelos entrenados para el clasificador del nivel superior del clasificador de curvas de luz. Además en las figuras 4.5 y 4.6 aparecen las matrices de confusión obtenidos.

Tabla 4.4: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador jerárquico con modelo XGB usando distintos métodos y métricas a maximizar.

Método	Métrica	Precision_M	Recall_M	F1-score_M	min Recall
Weight	Recall_M	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
Weight	CBA	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
Sampling	Recall_M	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
Sampling	CBA	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00

Se puede observar que para las cuatro combinaciones de métrica de evaluación - técnica de balance con el clasificador de la capa superior, se tuvieron valores casi perfectos para cada una de las métricas en estudio.

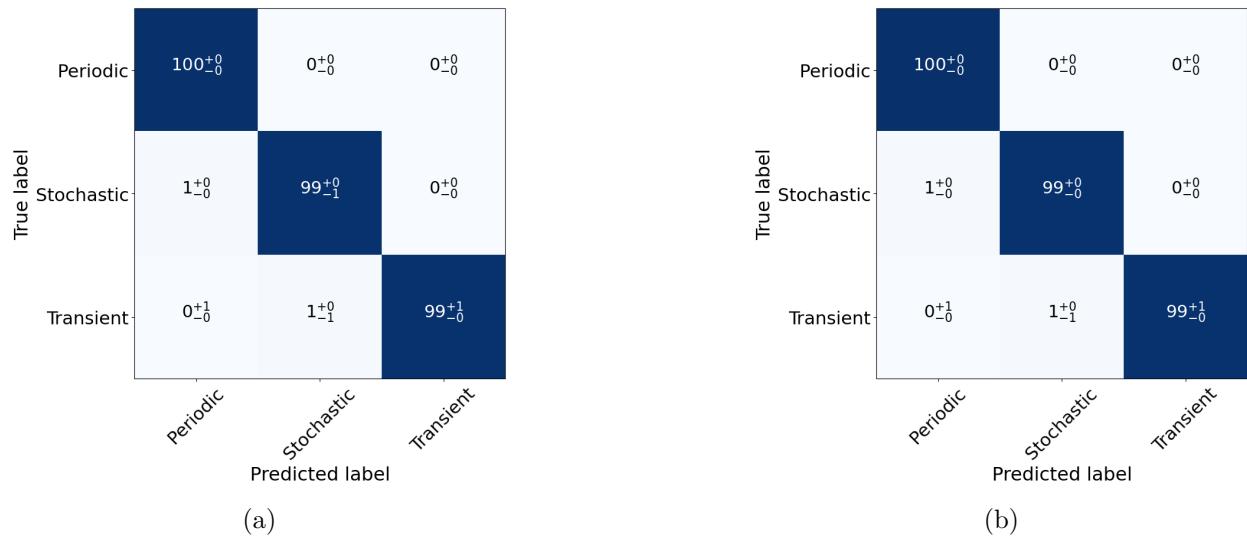
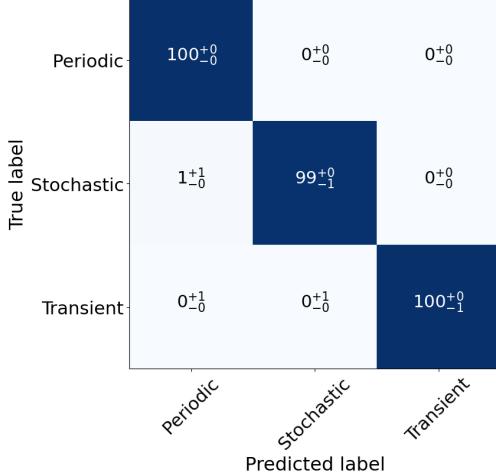
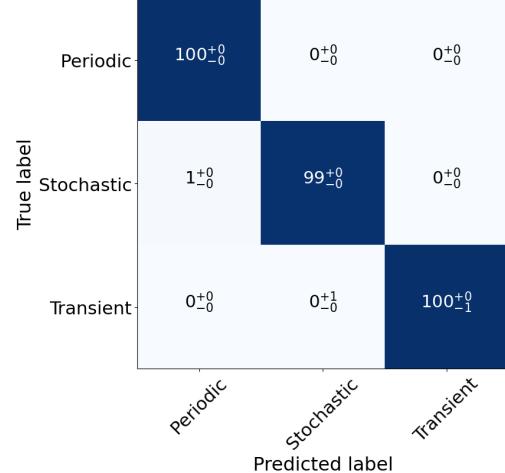


Figura 4.5: Matrices de confusión de la capa superior ocupando XGBoost con técnica de balance Weight y distintas métricas a maximizar: a) Recall_M y b) CBA.



(a)



(b)

Figura 4.6: Matrices de confusión de la capa superior ocupando XGBoost con técnica de balance Sampling y distintas métricas a maximizar: a) Recall_M y b) CBA.

De las matrices de confusión se destaca que las confusiones entre clases jerárquicas son mínimas, existiendo solo un porcentaje mínimo (1%) entre las clases Estocástica y Transientd. Para XGBoost no se evidencian diferencias cuantitativas para las distintas combinaciones de métrica-técnica de desbalance probadas.

En la tabla 4.5 se presentan a los valores promedio y la desviación estándar de los resultados obtenidos en el grupo de test de los modelos entrenados para el clasificador de curvas periódicas del nivel inferior del clasificador de curvas de luz.

Tabla 4.5: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Periódico con modelo XGB usando distintos métodos y métricas a maximizar.

Método	Métrica	Precision_M	Recall_M	F1-score_M	min Recall
Weight	Recall_M	0.64 ± 0.01	0.86 ± 0.01	0.69 ± 0.01	0.73 ± 0.02
Weight	CBA	0.83 ± 0.01	0.81 ± 0.01	0.82 ± 0.01	0.51 ± 0.02
Sampling	Recall_M	0.61 ± 0.00	0.86 ± 0.01	0.66 ± 0.01	0.75 ± 0.02
Sampling	CBA	0.85 ± 0.01	0.79 ± 0.01	0.49 ± 0.03	0.49 ± 0.03

Respecto a estos resultados, en todas las métricas excepto en Recall_M y min Recall en los modelos entrenados con CBA como métrica de evaluación superan a los entrenados con Recall_M . Ese es el mismo escenario que tuvo XGBoost sin técnicas de desbalance al ser comparado con BRF, lo cual podría considerarse como un signo preliminar de desbalance por parte de XGBoost con CBA como métrica de evaluación durante el entrenamiento.

Comparando entre si las distintas técnicas de balance (Weight y Sampling) para la misma métrica de evaluación, no se presentan diferencias significativas en los valores obtenidos para

el desempeño del clasificador Periódico en el grupo de test.

En la figuras 4.7 y 4.8 se muestran las matrices de confusión resultantes para la técnica Weight y Sampling respectivamente. De estas se puede ver observar que las técnicas de balance con CBA presentan altas tasas de acierto en las tres clases de mayor proporción, las que son bastante mayores a las obtenidas en las otras tres clases de menor proporción. Este escenario es similar al obtenido con XGBoost sin técnicas de balance, teniendo incluso las mismas confusiones entre clases, pero con la diferencia que las clases minoritarias han aumentado su tasa de aciertos en un poco más de 10 % en promedio.

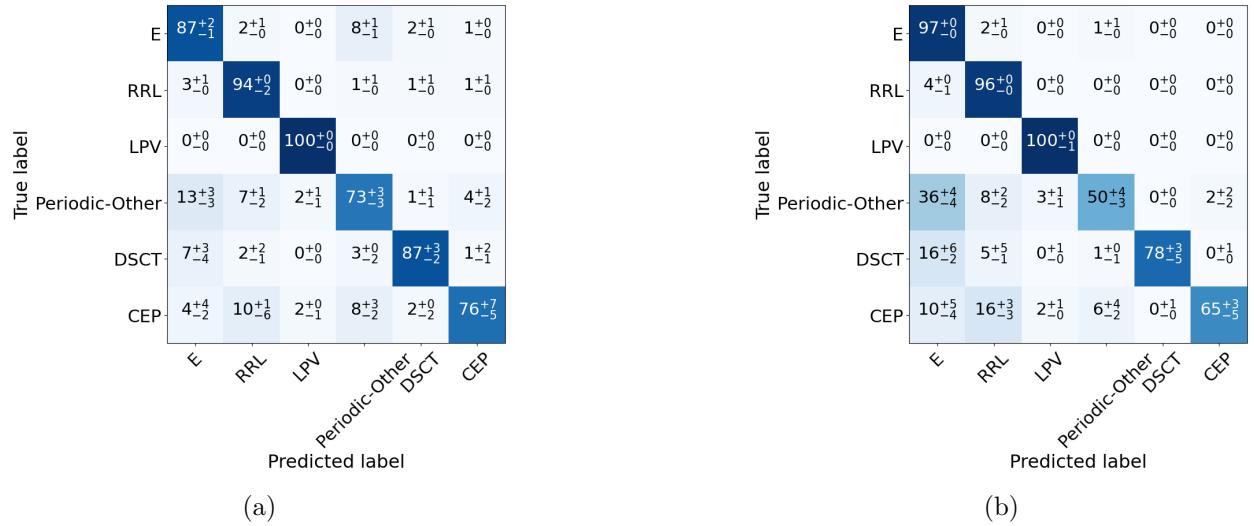


Figura 4.7: Matrices de confusión del clasificador Periódico de la capa inferior al aplicar XGBoost con técnica Weight y distintas métricas a maximizar.
a) Recall_M y b) CBA.

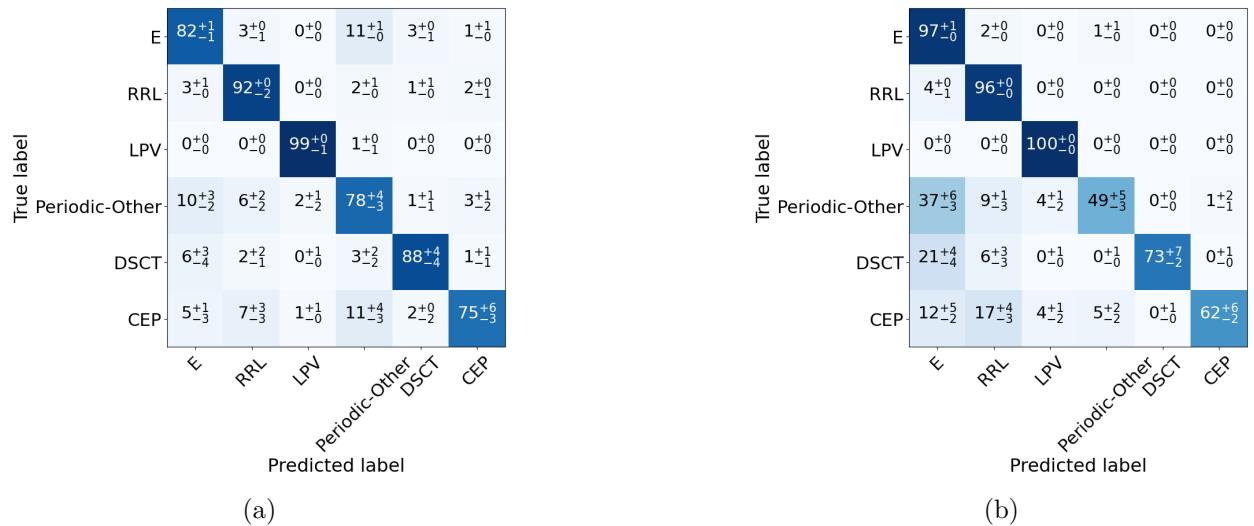


Figura 4.8: Matrices de confusión del clasificador Periódico de la capa inferior al aplicar XGBoost con técnica Sampling y distintas métricas a maximizar.
a) Recall_M y b) CBA.

Respecto al resultado de las técnicas de balance con Recall_M se da una situación parecida a la obtenida con BRF en la figura 4.1.b, donde las clases RRL y LPV son las de mayor tasa de aciertos, pero con XGBoost se obtuvo una mayor cantidad de aciertos para todas las clases.

Podría considerarse que el efecto del desbalance en el modelo predictivo para este clasificador ha sido contrarrestado en buena medida, distribuyendo de manera proporcional las predicciones entre todas las clases en cuestión.

XGBoost sin técnicas de balance predijo incorrectamente como perteneciente a la clase mayoritaria “E” a un 54%, 25% y un 18% del total de instancias de las clases minoritarias Periodic-Other, DSCT y CEP respectivamente. Estos porcentajes disminuyeron a 10%, 6% y 5% al utilizar la técnica de balance de Sampling y Recall_M .

La mayor diferencia entre técnicas de balance junto con Recall_M ocurre con las predicciones hacia la clase mayoritaria E, obteniendo un mayor porcentaje de aciertos con la técnica Weight que con Sampling, pero al mismo tiempo una mayor confusión con las clases minoritarias.

La tabla 4.6 presenta los resultados obtenidos con el clasificador Estocástico sobre sus correspondientes grupos de test, al ser entrenado utilizando distintas métricas a maximizar y métodos para el desbalance.

Tabla 4.6: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Estocástico con modelo XGB usando distintos métodos y métricas a maximizar.

Método	Métrica	Precision_M	Recall_M	F1-score_M	min Recall
Weight	Recall_M	0.82 ± 0.02	0.89 ± 0.01	0.85 ± 0.01	0.72 ± 0.03
Weight	CBA	0.90 ± 0.01	0.87 ± 0.01	0.88 ± 0.01	0.64 ± 0.03
Sampling	Recall_M	0.86 ± 0.01	0.89 ± 0.01	0.87 ± 0.01	0.66 ± 0.02
Sampling	CBA	0.86 ± 0.01	0.89 ± 0.01	0.87 ± 0.01	0.67 ± 0.03

Los valores obtenidos en la tabla 4.6 bastante similares entre las distintas combinaciones de métrica de evaluación y técnica de balance. Las mayores diferencias ocurren en a Precision_M y min Recall, donde en la primera la técnica Weight con CBA obtuvo el mayor puntaje con un 0.90. Para el mínimo Recall por clase, la combinación de Weight y Recall_M obtuvo el mayor valor con un 0.72, que tal como se puede ver en la matriz de la figura 4.9.a corresponde a la clase Blazar, pues es la de menor tasa de TP (al igual que en el resto de las combinaciones).

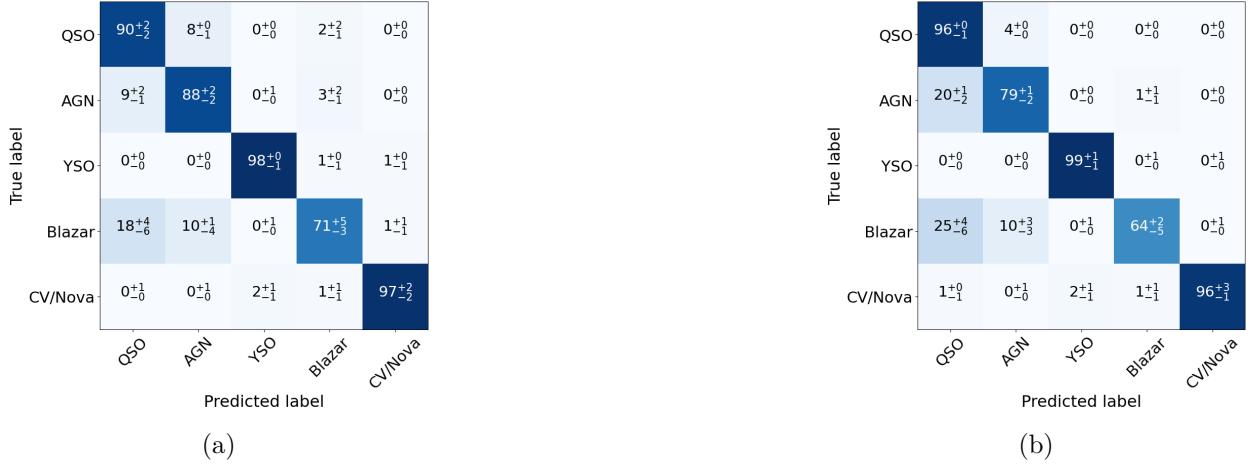


Figura 4.9: Matrices de confusión del clasificador Estocástico de la capa inferior al aplicar XGBoost con técnica Weight y distintas métricas a maximizar. a) Recall_M y b) CBA.

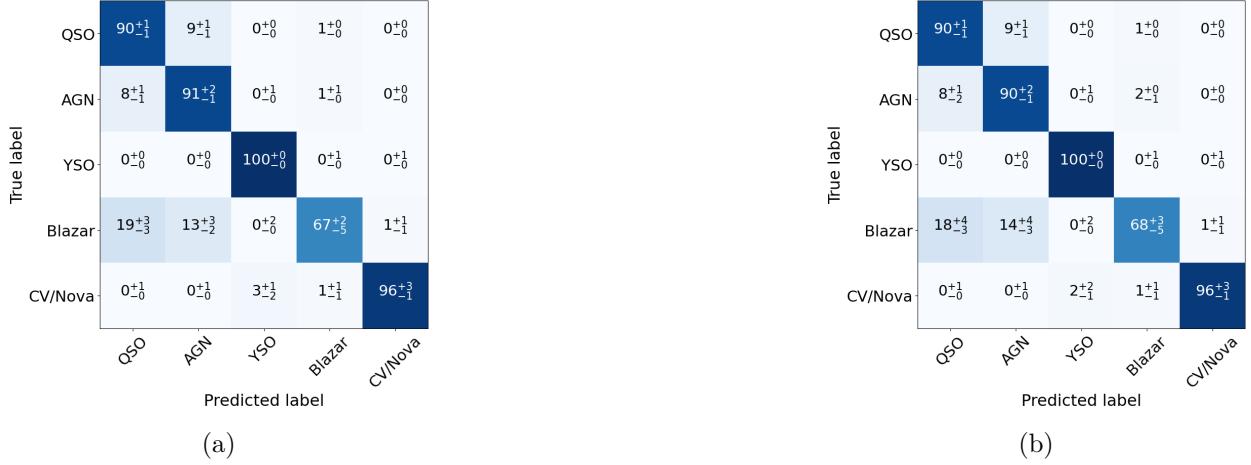


Figura 4.10: Matrices de confusión del clasificador Estocástico de la capa inferior al aplicar XGBoost con técnica Sampling y distintas métricas a maximizar. a) Recall_M y b) CBA.

Entre las combinaciones utilizadas con XGBoost para el clasificador Estocástico, la combinación Weight con CBA obtiene las mayores diferencias en las predicciones en el grupo de test con respecto al resto. Con esta combinación se observa una tendencia del clasificador hacia la clase mayoritaria QSO, provocando que haya una mayor confusión de las clases AGN y Blazar hacia QSO, en comparación con los otros casos.

En la tabla 4.7 se muestran los resultados obtenidos en los grupos de test de los modelos entrenados para el clasificador de curvas transientes de la capa inferior..

Tabla 4.7: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Transiente con modelo XGB usando distintos métodos y métricas a maximizar.

Método	Métrica	Precision_M	Recall_M	F1-score_M	min Recall
Weight	Recall_M	0.57 ± 0.05	0.68 ± 0.06	0.59 ± 0.05	0.57 ± 0.11
Weight	CBA	0.65 ± 0.07	0.67 ± 0.07	0.65 ± 0.07	0.43 ± 0.14
Sampling	Recall_M	0.62 ± 0.10	0.58 ± 0.07	0.58 ± 0.08	0.27 ± 0.13
Sampling	CBA	0.62 ± 0.09	0.58 ± 0.06	0.58 ± 0.07	0.26 ± 0.12

En la tablas 4.7 se observa que la combinación Weight y Recall_M obtuvo el desempeño más bajo en todas las métricas, salvo en el mínimo Recall con un valor de 0.57, que es significativamente mayor al resto, y en Recall_M donde obtuvo el mayor valor (0.68), pero prácticamente igualado con lo obtenido con la misma técnica de balance usando CBA como métrica de evaluación (0.67).

Cabe destacar que los valores obtenidos de mínimo Recall para los modelos entrenados con Sampling son bastante bajos (0.27 con Recall_M y 0.26 con CBA). Este bajo desempeño se ve reflejado en sus matrices de confusión correspondientes de la figura 4.12 en las que las dos mayoritarias tienen bastante más aciertos que las minoritarias.

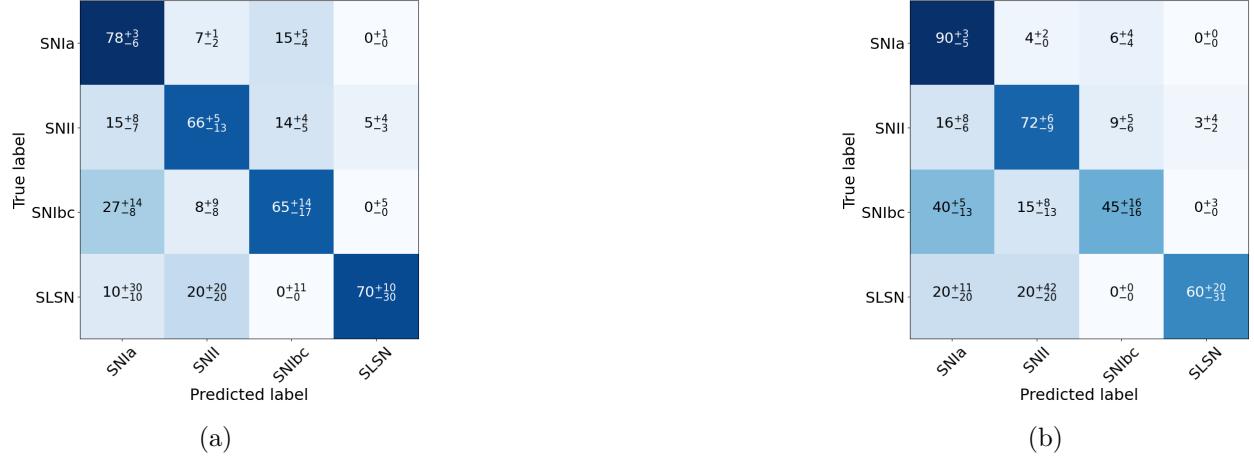


Figura 4.11: Matrices de confusión del clasificador Transiente de la capa inferior al aplicar XGBoost con técnica Weight y distintas métricas a maximizar. a) Recall_M y b) CBA.

	SNIa	SNII	SNIbc	SLSN
SNIa	89 ⁺⁴ ₋₉	7 ⁺⁶ ₋₂	4 ⁺⁴ ₋₃	0 ⁺¹ ₋₀
SNII	16 ⁺⁹ ₋₇	77 ⁺⁶ ₋₇	5 ⁺⁵ ₋₃	2 ⁺² ₋₂
SNIbc	45 ⁺³ ₋₁₂	28 ⁺¹⁰ ₋₁₉	28 ⁺¹² ₋₁₁	0 ⁺⁵ ₋₀
SLSN	20 ⁺¹¹ ₋₂₀	40 ⁺⁴⁰ ₋₂₀	0 ⁺⁰ ₋₀	40 ⁺²⁰ ₋₃₁

(a)

	SNIa	SNII	SNIbc	SLSN
SNIa	89 ⁺³ ₋₆	7 ⁺⁵ ₋₂	4 ⁺³ ₋₂	0 ⁺⁰ ₋₀
SNII	16 ⁺⁷ ₋₇	76 ⁺⁵ ₋₁₁	6 ⁺⁵ ₋₃	2 ⁺³ ₋₂
SNIbc	41 ⁺² ₋₁₂	29 ⁺¹⁵ ₋₂₀	29 ⁺¹⁰ ₋₁₉	0 ⁺⁰ ₋₀
SLSN	20 ⁺²⁰ ₋₂₀	40 ⁺²⁰ ₋₂₀	0 ⁺⁰ ₋₀	40 ⁺²⁰ ₋₃₁

(b)

Figura 4.12: Matrices de confusión del clasificador Transiente de la capa inferior al aplicar XGBoost con técnica Sampling y distintas métricas a maximizar. a) Recall_M y b) CBA.

Las predicciones de XGBoost con Sampling muestran un claro sesgo hacia las clases mayoritarias para las dos métricas de evaluación probadas. En ambos casos se obtuvieron escenarios muy similares a los obtenidos con XGBoost sin incorporar técnicas de balance, pero disminuyendo la cantidad de predicciones hacia la clase mayoritaria SNIa.

El bajo desempeño obtenido con las técnicas de aumento y reducción de datos para las curvas de luz del tipo Transiente puede deberse a la muy baja la cantidad de instancias en las clases minoritarias (SLSN con 19 y SNIbc con 75 para los grupos de entrenamiento. Para equiparar su proporción con el resto de las clases se debe duplicar repetidas veces las curvas de luz, lo cual no es aconsejable pues al no aportar nueva información las instancias repetidas pueden provocar un sobreajuste en el modelo.

Por otro lado los resultados obtenidos con la técnica Weight muestran una clara mejora con respecto a lo obtenido con el clasificador Transiente con XGBoost por sí solo. Analizando en detalle las predicciones de cada combinación se observa que utilizando CBA se obtiene una mayor cantidad de predicciones hacia las dos clases mayoritarias.

Aun que sesgar las predicciones a las clases mayoritarias resulta en una mayor tasa de aciertos en esas clases, esto no resuelve el problema de desbalance, provocando una mayor confusión con las clases minoritarias, las que obtienen un mal desempeño. Este escenario es distinto en la combinación de Weight con Recall_M, pues la tasa de aciertos es bastante equilibrada entre las cuatro clases a clasificar.

En la tablas 4.8 se muestras los resultados obtenidos al unir las predicciones de los datos del grupo de test del clasificador jerárquico de la capa superior, con las predicciones de los clasificadores Periódico, Estocástico y Transiente de la capa inferior del clasificador de curvas de luz.

Tabla 4.8: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para la unión de la capa superior e inferior del clasificador de curvas de luz, con modelo XGB usando distintos métodos y métricas a maximizar.

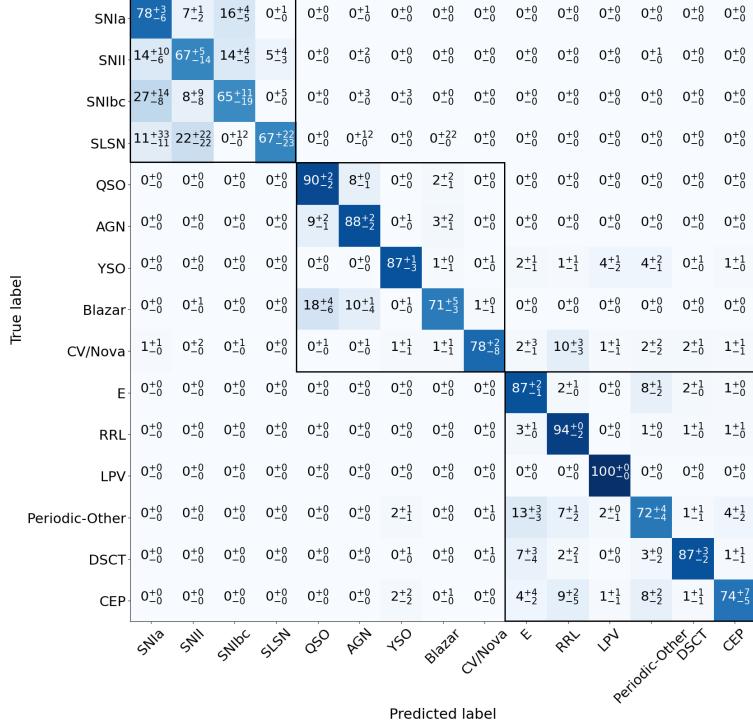
Método	Métrica	Precision_M	Recall_M	F1-score_M	min Recall
Weight	Recall_M	0.67 ± 0.01	0.79 ± 0.02	0.70 ± 0.01	0.53 ± 0.12
Weight	CBA	0.79 ± 0.02	0.76 ± 0.02	0.77 ± 0.02	0.38 ± 0.11
Sampling	Recall_M	0.69 ± 0.03	0.77 ± 0.02	0.69 ± 0.02	0.26 ± 0.12
Sampling	CBA	0.78 ± 0.03	0.74 ± 0.02	0.75 ± 0.02	0.25 ± 0.12

Se desprende de la tabla 4.8 que la tendencia de los valores obtenidos es similar a la de los casos de los clasificadores Periódico y Transiente, en que los resultados de las combinaciones con CBA como métrica de evaluación del entrenamiento son superiores a los obtenidos con Recall_M , excepto en lo que respecta Recall_M y min Recall.

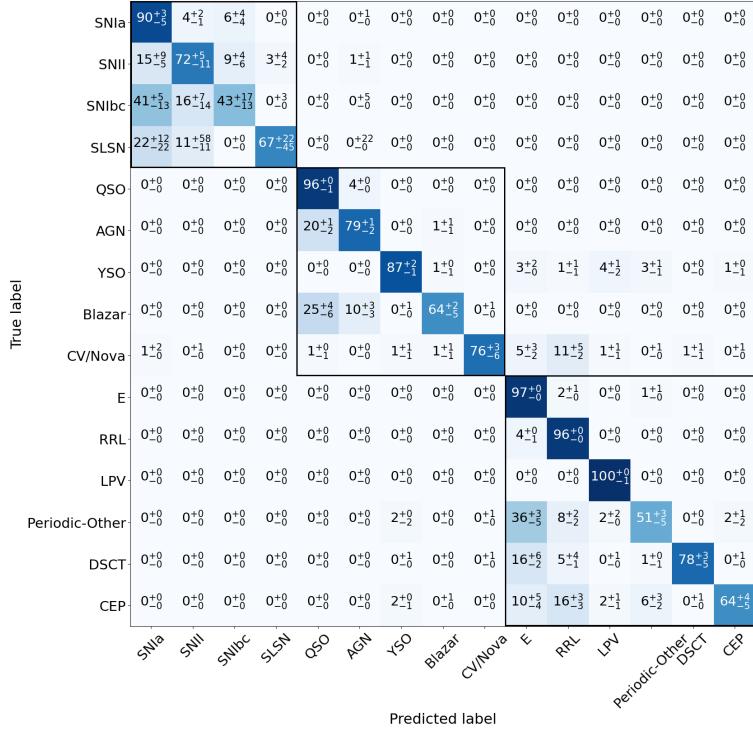
En la figura 4.13 se muestran las matrices de confusión obtenidas para la unión de la capa superior e inferior del clasificador de curvas de luz al ser entrenado con la técnica Weight y distintas métricas a maximizar. Así mismo, en la figura 4.14 muestran las matrices de confusión para la unión de capas del clasificador de curvas de luz al ser entrenado con distintas métricas a maximizar, pero utilizando la técnica de balance Sampling.

A grandes rasgos para la misma métrica de evaluación y distinta técnica de balanceo se obtuvieron resultados bastante similares, salvo para el clasificador Transiente. Esto se debe a que las técnicas Weight y Sampling son a grandes rasgos equivalentes, ya que alterar la cantidad de instancias por clase provocaría que las pertenecientes a las clases minoritarias sean evaluadas con la función de pérdida una mayor cantidad de veces que las mayoritarias, lo cual es equivalente a ponderar por un mayor o menor valor estas mismas instancias.

Que las técnicas Weight y Sampling sean equivalentes podría explicar porque se obtuvieron desempeños tan similares en los clasificadores Periódico y Estocástico. El caso del clasificador Transiente es diferente posiblemente debido al mayor desbalance presente y con menos instancias, lo cual implica duplicar de manera excesiva las instancias de las clases minoritarias.



(a)



(b)

Figura 4.13: Matrices de confusión de la unión de capas del clasificador de curvas de luz aplicando XGBoost con técnica Weight y distintas métricas a maximizar: a) Recall_M y b) CBA.

	SNIa	SNII	SNIbc	SLSN	QSO	AGN	YSO	Blazar	CV/Nova	E	RRL	LPV	Periodic-Other	DSCT	CEP
True label	89^{+4}_{-9}	7^{+5}_{-2}	4^{+3}_{-2}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
QSO	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	90^{+1}_{-1}	9^{+1}_{-1}	0^{+0}_{-0}	1^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
AGN	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	8^{+1}_{-1}	91^{+2}_{-2}	0^{+0}_{-0}	1^{+1}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
YSO	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	88^{+2}_{-2}	0^{+1}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	1^{+2}_{-0}	1^{+1}_{-1}	3^{+2}_{-1}	6^{+2}_{-1}	0^{+1}_{-0}
Blazar	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	19^{+3}_{-3}	13^{+3}_{-2}	0^{+2}_{-0}	67^{+5}_{-5}	1^{+1}_{-1}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
CV/Nova	1^{+2}_{-1}	1^{+1}_{-1}	0^{+1}_{-0}	0^{+0}_{-0}	0^{+1}_{-0}	0^{+1}_{-0}	1^{+1}_{-1}	1^{+0}_{-1}	77^{+4}_{-6}	3^{+3}_{-2}	9^{+5}_{-2}	1^{+1}_{-1}	3^{+2}_{-2}	3^{+2}_{-1}	1^{+1}_{-1}
E	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	82^{+1}_{-1}	3^{+0}_{-1}	0^{+0}_{-0}	11^{+1}_{-1}	3^{+1}_{-1}	1^{+1}_{-1}
RRL	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	3^{+1}_{-0}	92^{+0}_{-2}	0^{+0}_{-0}	2^{+1}_{-0}	1^{+1}_{-0}	2^{+0}_{-0}
LPV	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	99^{+0}_{-1}	1^{+0}_{-1}	0^{+0}_{-0}	0^{+0}_{-0}
Periodic-Other	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	1^{+1}_{-1}	0^{+0}_{-0}	0^{+0}_{-0}	10^{+3}_{-2}	6^{+2}_{-2}	2^{+2}_{-2}	77^{+5}_{-3}	1^{+1}_{-1}	3^{+2}_{-2}
DSCT	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+1}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	5^{+4}_{-3}	2^{+2}_{-1}	0^{+1}_{-0}	3^{+2}_{-2}	88^{+4}_{-4}	1^{+1}_{-1}
CEP	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	2^{+0}_{-0}	0^{+1}_{-0}	0^{+0}_{-0}	4^{+2}_{-2}	7^{+3}_{-3}	1^{+1}_{-1}	10^{+4}_{-2}	1^{+1}_{-1}	74^{+4}_{-4}

(a)

	SNIa	SNII	SNIbc	SLSN	QSO	AGN	YSO	Blazar	CV/Nova	E	RRL	LPV	Periodic-Other	DSCT	CEP
True label	89^{+2}_{-6}	7^{+5}_{-2}	4^{+3}_{-2}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
QSO	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	90^{+1}_{-1}	9^{+1}_{-1}	0^{+0}_{-0}	1^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
AGN	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	8^{+1}_{-1}	90^{+2}_{-2}	0^{+0}_{-0}	2^{+0}_{-1}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
YSO	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	88^{+2}_{-3}	0^{+1}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	3^{+2}_{-1}	1^{+1}_{-0}	4^{+2}_{-2}	3^{+1}_{-1}	0^{+0}_{-0}
Blazar	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	18^{+4}_{-3}	14^{+3}_{-3}	0^{+2}_{-0}	68^{+3}_{-5}	1^{+0}_{-1}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
CV/Nova	1^{+1}_{-1}	1^{+1}_{-1}	0^{+1}_{-0}	0^{+0}_{-0}	0^{+1}_{-0}	0^{+1}_{-0}	1^{+1}_{-1}	1^{+1}_{-1}	75^{+4}_{-5}	6^{+3}_{-3}	12^{+3}_{-4}	1^{+1}_{-1}	0^{+0}_{-0}	1^{+0}_{-1}	0^{+0}_{-0}
E	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	97^{+1}_{-0}	2^{+0}_{-0}	0^{+0}_{-0}	1^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
RRL	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	4^{+0}_{-0}	96^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
LPV	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	100^{+0}_{-1}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
Periodic-Other	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	2^{+2}_{-1}	0^{+0}_{-0}	0^{+1}_{-0}	37^{+6}_{-5}	9^{+3}_{-3}	3^{+2}_{-1}	49^{+3}_{-3}	0^{+0}_{-0}	1^{+2}_{-1}
DSCT	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	1^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	21^{+4}_{-5}	6^{+3}_{-3}	0^{+1}_{-0}	0^{+1}_{-0}	73^{+6}_{-5}	0^{+0}_{-0}
CEP	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	2^{+2}_{-1}	0^{+0}_{-0}	0^{+0}_{-0}	12^{+5}_{-5}	17^{+4}_{-3}	3^{+1}_{-1}	5^{+2}_{-2}	0^{+1}_{-0}	61^{+6}_{-5}

(b)

Figura 4.14: Matrices de confusión de la unión de capas del clasificador de curvas de luz aplicando XGBoost con técnica Sampling y distintas métricas a maximizar: a) Recall_M y b) CBA.

Al comparar una misma técnica con distintas métricas de evaluación, Recall_M obtuvo mejor desempeño. Esto principalmente porque al utilizar Recall_M se obtiene una menor desviación

entre las clases, a diferencia de CBA donde las clases mayoritarias tienen notablemente valores más altos que las minoritarias.

En la tabla 4.9 se resume la proporción de tasa de aciertos por clase para cada combinación a nivel de toda la capa inferior del clasificador.

Tabla 4.9: Promedio, mediana y desviación estándar (std) del porcentaje de aciertos para la unión de niveles del clasificador de curvas de luz con XGBoost con técnicas de balance y distintas métricas de evaluación.

Método	Métrica	Promedio	Mediana	Std
Weight	Recall_M	80 %	78 %	10 %
Weight	CBA	77 %	78 %	17 %
Sampling	Recall_M	77 %	82 %	19 %
Sampling	CBA	75 %	76 %	20 %

Tomando en cuenta los valores obtenidos en la tabla 4.9 y la confusión de instancias de clases minoritarias hacia las mayoritarias, se confirma que Recall_M es la mejor métrica de evaluación a considerar durante entrenamiento. Es decir, numéricamente es la mejor forma de reflejar el real desempeño de las predicciones en escenarios con datos desbalanceados. Además, la técnica Weight con Recall_M es la combinación con la cual se obtuvo el mejor desempeño al ser integrada con XGBoost.

CBA resultó ser la peor métrica de evaluación debido a que Precision (métrica que toma en cuenta junto a Recall) es sensible al efecto del desbalance en los modelos predictivos, provocando que también CBA se vea afectado. Cabe mencionar que el mismo problema sucede con F1-score.

Si bien los resultados de XGBoost utilizando la técnica de balance Weight fueron superiores a los obtenidos con técnicas a nivel de datos (Sampling), el desempeño obtenido con estas últimas fue bastante similar, excepto en el clasificador Transiente.

XGBoost utilizando Recall_M para su entrenamiento superó a BRF en todas las métricas de evaluación para los tres clasificadores de la capa inferior (excepto en Transiente para Recall_M), por lo que en el capítulo 5 se evaluará si las diferencias son significativas.

Debido a que con el clasificador del nivel superior se han obtenido resultados casi perfectos con XGBoost al utilizar técnicas de balance y sin éstas, es que no se continuará con el entrenamiento de este clasificador para la experimentación con distintas técnicas a nivel de datos.

4.4. Entrenamiento con técnicas de Sobremuestreo

A continuación, se presentan los mejores resultados obtenidos al utilizar distintas técnicas de sobremuestreo para preprocesar los grupos de entrenamiento de los tres clasificadores de la capa inferior del clasificador de curvas de luz. El proceso de aumentar la proporción de las

clases minoritarias se repite en cada iteración del procedimiento de Nested Cross-Validation, donde además en los loops interiores se escogen los mejores parámetros para las técnicas utilizadas.

Las técnicas de sobremuestreo utilizadas y sus parámetros a seleccionar fueron:

- ADASYN (Adaptive Synthetic) [14]
 - *strategy, k_neighbors*
- Borderline SMOTE (BSM) [12]
 - *k_neighbors, m_neighbors, kind*
- Random Oversampling (ROS)
 - *strategy*
- SMOTE [11]
 - *strategy, k_neighbors*
- Support Vector Machine SMOTE (SVM) [13]
 - *strategy, k_neighbors, m_neighbors,*
- Mahalanobis Distance Oversampling (MDO) [15]
 - *strategy, k_neighbors*

La explicación de cada uno de estos parámetros se encuentra en la sección 8.1.2.2 del anexo.

En la tabla 4.10 se muestran los resultados obtenidos en los grupos de test del clasificador Periódico al ser entrenado con las distintas técnicas de sobremuestreo. La figura 4.15 muestra las correspondientes matrices de confusión.

Tabla 4.10: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Periódico con modelo XGB usando técnicas de sobremuestreo.

Método	Precision_M	Recall_M	F1-score_M	min Recall
ADASYN	0.81 ± 0.06	0.77 ± 0.02	0.78 ± 0.03	0.42 ± 0.07
BSM	0.87 ± 0.04	0.76 ± 0.02	0.80 ± 0.01	0.37 ± 0.06
ROS	0.86 ± 0.06	0.76 ± 0.03	0.80 ± 0.02	0.42 ± 0.09
SMOTE	0.78 ± 0.08	0.77 ± 0.02	0.77 ± 0.04	0.43 ± 0.05
SVM	0.88 ± 0.03	0.75 ± 0.02	0.80 ± 0.01	0.34 ± 0.04
MDO	0.91 ± 0.01	0.74 ± 0.01	0.80 ± 0.01	0.33 ± 0.04

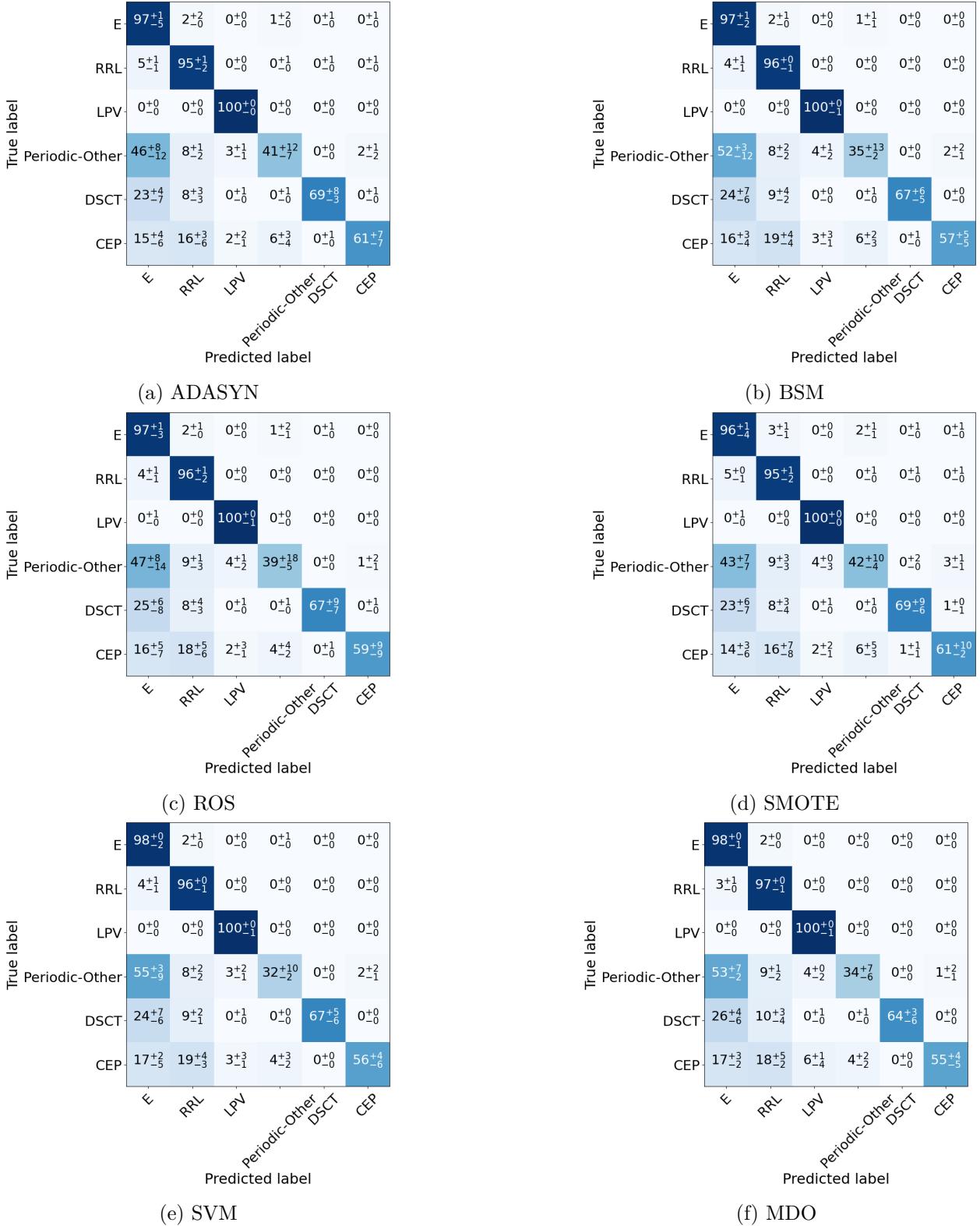


Figura 4.15: Matrices de confusión del clasificador Periódico con distintas técnicas de sobremuestreo.

En la tabla 4.10 se observa que se alcanzaron valores bastante altos de Precision_M , entre 0.78 y 0.91. Sin embargo en Recall, a nivel macro se obtuvo un promedio de 0.76 entre las

distintas técnicas, lo cual es bajo en comparación con lo obtenido con BRF (0.82). Esta situación, sumado al bajo mínimo Recall obtenido, donde el máximo valor fue de 0.43 al utilizar SMOTE, da un claro indicio de un efecto de desbalance de clases.

El efecto del desbalance es confirmado al analizar las matrices de confusión de la figura 4.15, donde se observa que entre las distintas técnicas de sobremuestreo se obtuvieron resultados bastante similares, donde las tres clases mayoritarias tuvieron una tasa de aciertos muy superior a lo obtenido en el resto, mostrando un claro sesgo en las predicciones del modelo hacia las clases mayoritarias.

En la tabla 4.11 se muestran los resultados obtenidos en los grupos de test del clasificador Estocástico al ser entrenado con las distintas técnicas de sobremuestreo. La figura 4.16 muestra las matrices de confusión resultantes.

Tabla 4.11: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Estocástico con modelo XGB usando técnicas de sobremuestreo.

Método	Precision_M	Recall_M	F1-score_M	min Recall
ADASYN	0.89 ± 0.01	0.86 ± 0.01	0.87 ± 0.01	0.63 ± 0.02
BSM	0.90 ± 0.01	0.86 ± 0.01	0.88 ± 0.01	0.61 ± 0.03
ROS	0.91 ± 0.01	0.86 ± 0.01	0.88 ± 0.01	0.61 ± 0.03
SMOTE	0.89 ± 0.01	0.86 ± 0.01	0.87 ± 0.01	0.63 ± 0.03
SVM	0.91 ± 0.01	0.86 ± 0.01	0.88 ± 0.00	0.60 ± 0.02
MDO	0.92 ± 0.00	0.85 ± 0.01	0.88 ± 0.00	0.60 ± 0.03

Nuevamente se da el caso de altos valores de Precision_M (en promedio 0.90), pero el Recall_M en todas los casos es mayor al obtenido con BRF, siendo 0.85 el valor más bajo obtenido, correspondiente a la técnica MDO. De todas formas, el mínimo Recall obtenido por clase es notablemente más bajo en cada caso, con un promedio entre las distintas técnicas de 0.61.

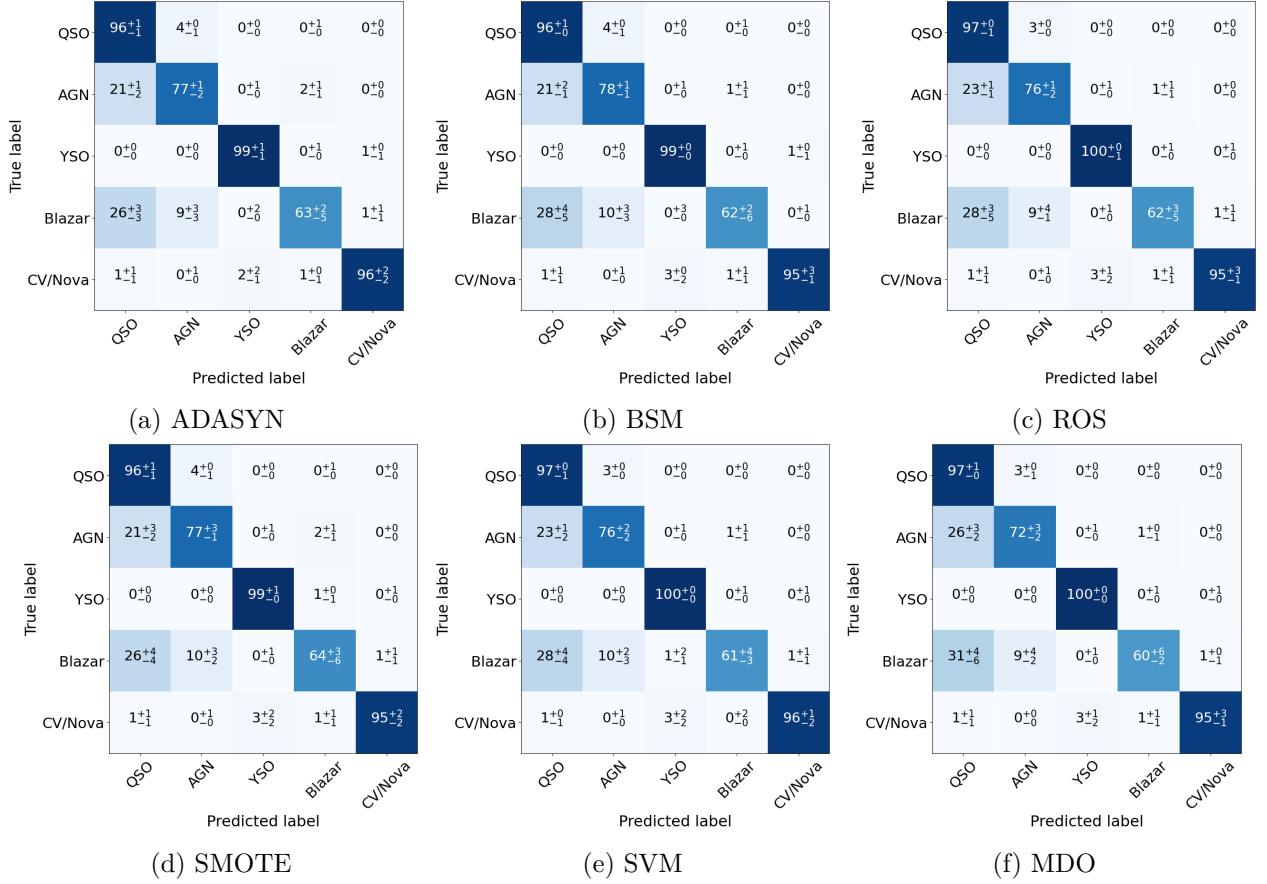


Figura 4.16: Matrices de confusión del clasificador Estocástico con distintas técnicas de sobremuestreo

En la figura 4.16 se observa que las clases de mayor acierto corresponden a la mayoritaria (QSO), la intermedia (YSO) y la minoritaria (CV/Nova). Situación idéntica a lo obtenido con XGBoost sin técnicas de balance, con la sola diferencia que en la clase AGN aumentó la tasa de aciertos.

En la tabla 4.12 se muestran los resultados obtenidos en los grupos de test del clasificador Transiente de la capa inferior del clasificador de curvas de luz, al ser entrenado con las distintas técnicas de sobremuestreo.

Tabla 4.12: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Transiente con modelo XGB usando técnicas de sobremuestreo.

Método	Precision _M	Recall _M	F1-score _M	min Recall
ADASYN	0.66±0.10	0.56±0.08	0.58±0.08	0.20±0.13
BSM	0.67±0.13	0.52±0.06	0.55±0.08	0.10±0.10
ROS	0.68±0.10	0.53±0.05	0.56±0.06	0.13±0.08
SMOTE	0.65±0.08	0.55±0.06	0.58±0.06	0.20±0.08
SVM	0.63±0.13	0.48±0.06	0.50±0.08	0.06±0.11
MDO	0.74±0.14	0.50±0.05	0.55±0.07	0.09±0.07

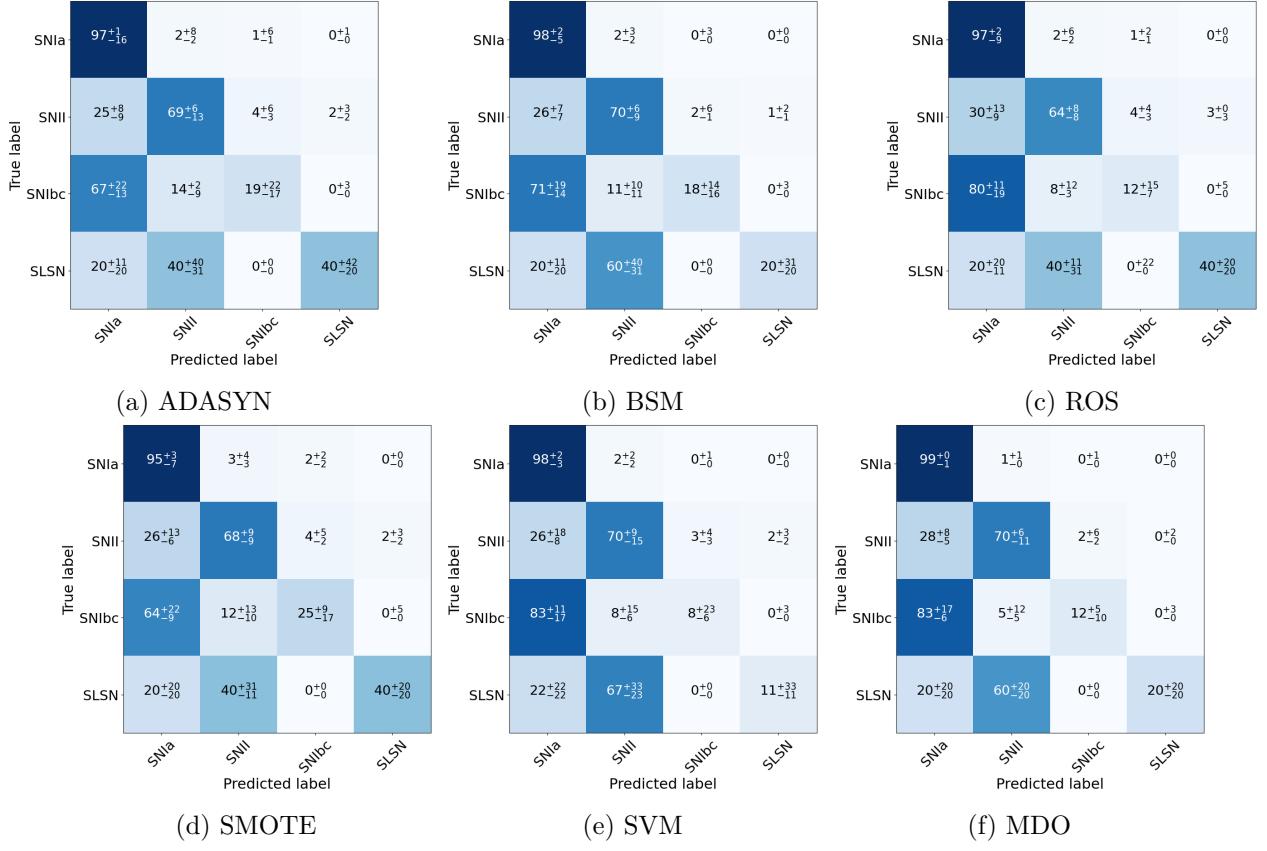


Figura 4.17: Matrices de confusión del clasificador Transiente con distintas técnicas de sobremuestreo.

El clasificador Transiente es el que más evidencia el efecto del desbalance, principalmente en el mínimo Recall por clase obtenido, donde el máximo valor fue de 0.20 para las técnicas ADADSYN y SMOTE, existiendo casos de 0.06 y 0.09 de mínimo Recall para las técnicas SVM y MDO respectivamente.

Los escenarios ilustrados por las matrices de confusión en la figura 4.17 para las técnicas que se obtuvieron mejor mínimo Recall son bastante similares a los obtenidos en la figura 4.12.a para Sampling con Recall_M como métrica de evaluación, donde se obtuvo un sesgo hacia las clases mayoritarias, obteniendo así un bajo desempeño en el resto de las clases. Para las otras técnicas de sobremuestreo este sesgo fue el mismo y de mayor magnitud.

A nivel general utilizar solamente técnicas de sobremuestreo para preprocesar los datos de entrenamiento ayudo a aumentar en baja medida el desempeño del modelo XGBoost en las clases con las que se obtuvieron peores resultados al momento de entrenar este mismo modelo sin técnica de balance alguna. Pero de igual forma, esta mejora no es suficiente, ya que siguen siendo las clases mayoritarias las que proporcionalmente se llevan gran parte de las predicciones del modelo.

Realizar *oversampling* en los datos de entrenamiento puede ser una mala estrategia, ya que se está buscando nivelar hacia arriba la cantidad de instancias por clase. Al ser tan grande el desbalance, implicaría en el caso de generación de instancias sintéticas, introducir demasiado

ruido en grupo de entrenamiento, ya que para ciertas clases es necesario aumentar su proporción más de 10 veces, finalizando con clases que tendrían más participación de instancias sintéticas que originales.

4.5. Entrenamiento con técnicas de Submuestreo y Sobremuestreo

Debido a las grandes diferencias en la cantidad de instancias entre las clases mayoritarias y minoritarias, es que se decidió no realizar pruebas de técnicas de submuestreo, debido a que implicaría la pérdida de mucha información que pudiese ser valiosa para el entrenamiento de un modelo predictivo. Se optó por priorizar la experimentación con combinaciones de técnicas de submuestreo y sobremuestreo de manera que ni se aumente ni se reduzca en cantidades drásticas la data.

En esta sección se presentaran los mejores resultados obtenidos al combinar distintas técnicas de Submuestreo y Sobremuestreo con el modelo XGBoost. Cabe mencionar que los parámetros de cada una de estas técnicas fueron seleccionados en el loop interior del procedimiento de Nested Cross-Validation de forma de obtener el mejor desempeño posible.

Las técnicas de *Submuestreo* utilizadas y sus parámetros a seleccionar fueron:

- All-KNN [19]
 - *strategy, n_neighbors, kind_sel*
- Condensed Nearest Neighbour (CNN) [17]
 - *strategy, n_neighbors*
- Edited Nearest Neighbour (ENN) [18]
 - *strategy, n_neighbors, kind_sel*
- Instance Hardness Threshold (IHT) [21]
 - *strategy*
- Neighbourhood Cleaning Rule (NCR) [20]
 - *strategy, n_neighbors, kind_sel*
- Near Miss (NM) [16]
 - *strategy, version*
- One Sided Selection (OSS) [31]
 - *strategy, n_neighbors*
- Repeated Nearest Neighbour (RENN) [19]

- *strategy*, *n_neighbors*, *kind_sel*
- Random Undersampling (RUS)
 - *strategy*
- Tomek Links (TL) [32]
 - *strategy*

La explicación de cada uno de estos parámetros se encuentra en la sección 8.1.2.3 del anexo.

Cada una de estas técnicas se combinó con las técnicas de *Sobremuestreo* utilizadas en la sección anterior.

Dadas las múltiples combinaciones realizadas para preprocessar los datos de entrenamiento con XGBoost para los distintos clasificadores, a continuación, se presentaran los resultados que cumplieron por la condición de filtrado propuesta en la metodología, esto es que el mínimo de recall entre sus clases supere el valor mínimo establecido.

Para establecer el límite para cada clasificador se utilizaron dos criterios. El primero es que el mínimo recall sea similar al valor mínimo entre el obtenido con BRF y el obtenido por alguna de las dos mejores combinaciones de método-métrica realizadas con XGBoost. El segundo criterio establece un valor mínimo de recall tal que sean como máximo 10 combinaciones de técnicas de muestreo las que cumplan con la condición. Se opta por el criterio que filtre la mayor cantidad de combinaciones posibles.

Los valores mínimos establecidos para cada clasificador son:

- min Recall clasificador Periódico: 0.7
- min Recall clasificador Estocástico: 0.7
- min Recall clasificador Transiente: 0.4

En la tabla 4.13 se muestran los resultados de las combinaciones de métodos de submuestreo y sobremuestreo que igualan o superan el mínimo valor de recall por clase establecido para el clasificador Periódico. La figura 4.18 muestra las correspondientes matrices de confusión.

Los mejores resultados fueron obtenidos con las técnicas de submuestreo *Condensed Nearest Neighbour* (CNN) y *Instance Hardness Threshold* (IHT). La primera utiliza técnicas de clustering y la segunda entrena modelos predictivos estándar para luego obtener la probabilidad de clasificar correctamente cierta instancia y seleccionar cuales mantener en base a este resultado (las de menor dificultad).

Tabla 4.13: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Periódico con modelo XGB usando combinaciones de técnicas de balance de datos con mejores resultados.

Método Under	Método Over	Precision_M	Recall_M	F1-score_M	min Recall
CNN	ROS	0.61 ± 0.01	0.84 ± 0.01	0.64 ± 0.01	0.72 ± 0.02
CNN	ADASYN	0.59 ± 0.01	0.83 ± 0.01	0.62 ± 0.01	0.71 ± 0.02
CNN	SMOTE	0.59 ± 0.01	0.83 ± 0.01	0.61 ± 0.01	0.70 ± 0.03
CNN	BSM	0.60 ± 0.01	0.83 ± 0.01	0.63 ± 0.01	0.70 ± 0.02
IHT	ROS	0.57 ± 0.00	0.83 ± 0.01	0.60 ± 0.01	0.73 ± 0.03
IHT	SMOTE	0.57 ± 0.00	0.82 ± 0.01	0.59 ± 0.00	0.73 ± 0.03
IHT	ADASYN	0.57 ± 0.00	0.82 ± 0.01	0.59 ± 0.01	0.73 ± 0.03
IHT	BSM	0.57 ± 0.00	0.82 ± 0.01	0.59 ± 0.00	0.71 ± 0.03

Los resultados obtenidos en la tabla 4.13 para las mejores combinaciones al entrenar el clasificador Periódico son similares a los obtenidos con BRF, pero en comparación con XGBoost sin técnicas de balance, se obtuvo mejor recall en su versión macro y mínimo por clase y valores más bajos en el resto de las métricas, indicando una mejora en cuanto al desbalance.

Las matrices de confusión de la figura 4.18 son bastante similares entre sí, presentando las mayores diferencias en la tasa de aciertos de la clase mayoritaria RRL, siendo las combinaciones con CNN las de mejor desempeño en esta clase (y las de mayor Recall_M). Comparando estas combinaciones con lo obtenido con BRF en la figura 4.1.b, se observa que solo en la clase Periodic-Other se obtuvo un significativo aumento de aciertos.

En cuanto a la combinación de técnicas de sobremuestreo con CNN para la reducción de datos, se decide que la técnica con ROS es la mejor para implementarse con XGBoost para el clasificador Periódico. Esta técnica presenta la menor confusión al predecir instancias de la clase E como perteneciente a la Periodic-Other (19 % del total de la clase E), ya que otras técnicas la mencionada confusión fue del 22 %.

En la tabla 4.14 se presentan los resultados obtenidos para las combinaciones de métodos de muestreo que hayan igualado o superado el valor mínimo de recall por clase establecido para el clasificador estocástico. La figura 4.19 muestra las correspondientes matrices de confusión.

Las combinaciones de técnicas de balanceo de datos de mayor minimo Recall se dieron para las combinaciones de todas las técnicas de sobremuestreo con los métodos de submuestreo ALL-KNN (similar a CNN) e IHT.

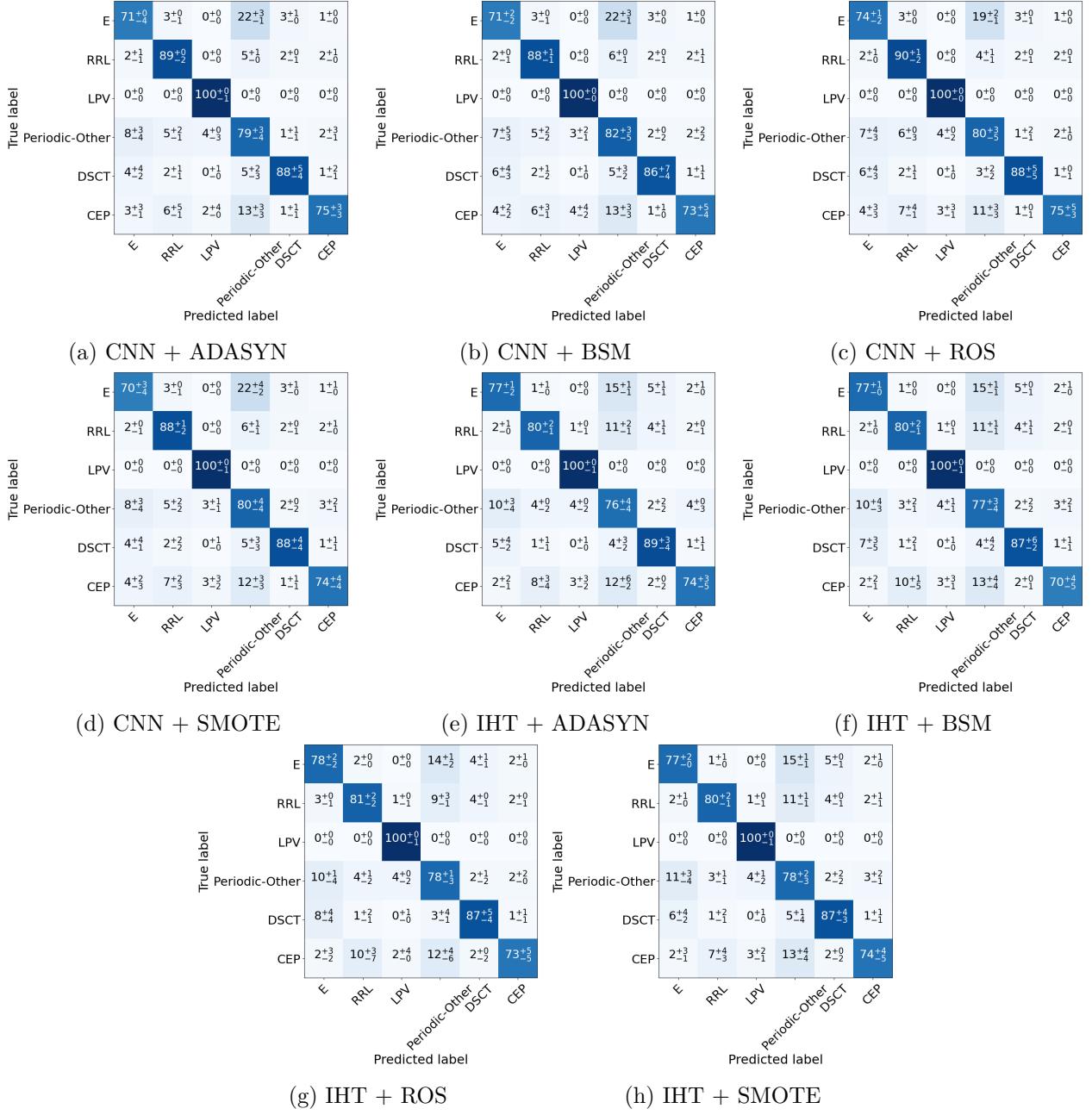


Figura 4.18: Matrices de confusión del clasificador Periódico con distintas combinaciones de técnicas de submuestreo y sobremuestreo.

Tabla 4.14: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Estocástico con modelo XGB usando combinaciones de técnicas de balance de datos con mejores resultados.

Método Under	Método Over	Precision_M	Recall_M	F1-score_M	min Recall
ALL-KNN	SMOTE	0.75 ± 0.01	0.88 ± 0.01	0.79 ± 0.01	0.72 ± 0.02
ALL-KNN	BSM	0.75 ± 0.01	0.88 ± 0.01	0.78 ± 0.01	0.72 ± 0.03
ALL-KNN	SVM	0.74 ± 0.01	0.87 ± 0.01	0.78 ± 0.01	0.72 ± 0.02
ALL-KNN	ADASYN	0.75 ± 0.01	0.88 ± 0.01	0.78 ± 0.01	0.72 ± 0.03
ALL-KNN	ROS	0.75 ± 0.01	0.87 ± 0.01	0.79 ± 0.01	0.70 ± 0.03
IHT	ADASYN	0.74 ± 0.01	0.88 ± 0.01	0.77 ± 0.01	0.72 ± 0.02
IHT	SMOTE	0.74 ± 0.01	0.87 ± 0.01	0.77 ± 0.00	0.72 ± 0.02
IHT	SVM	0.73 ± 0.01	0.87 ± 0.01	0.77 ± 0.01	0.71 ± 0.02
IHT	BSM	0.73 ± 0.00	0.87 ± 0.01	0.76 ± 0.00	0.71 ± 0.02
IHT	ROS	0.74 ± 0.01	0.87 ± 0.01	0.77 ± 0.01	0.71 ± 0.02

Al comparar los resultados de XGBoost entrenado con datos equilibrados mediante las combinaciones de submuestreo y sobremuestreo con lo obtenido con el mismo modelo sin técnicas de balanceo, se repite la mejora en el Recall (macro y mínimo por clase) y baja en el resto de las métricas, indicando una mejora en lo que respecta al desbalance. Al comparar con los resultados de BRF en la tabla 4.2 para el clasificador Estocástico, se obtuvo un desempeño prácticamente igual para Recall_M y mínimo Recall, y un peor resultado en el resto de las métricas con respecto a lo obtenido al implementar la combinación de técnicas de sampleo.

Analizando las respectivas matrices de confusión de la figura 4.19 se observan resultados muy similares en cuanto a las predicciones realizadas. Las mayores diferencias se presentan en la confusión de instancias QSO como AGN (ambas clases mayoritarias), donde las combinaciones con ALL-KNN presentan mayor sesgo hacia QSO y las combinaciones con IHT hacia AGN.

Las mayores diferencias de las predicciones de XGBoost con combinación de técnicas de muestreo, con lo obtenido con BRF en la tabla 4.1.c, se encuentran en la tasa de aciertos de las clases mayoritarias y las consecuentes confusiones entre éstas. BRF obtuvo un mejor desempeño en la clase mayoritaria QSO de al menos un 10% en su tasa de aciertos, pero al mismo tiempo obtuvo menores aciertos en AGN.

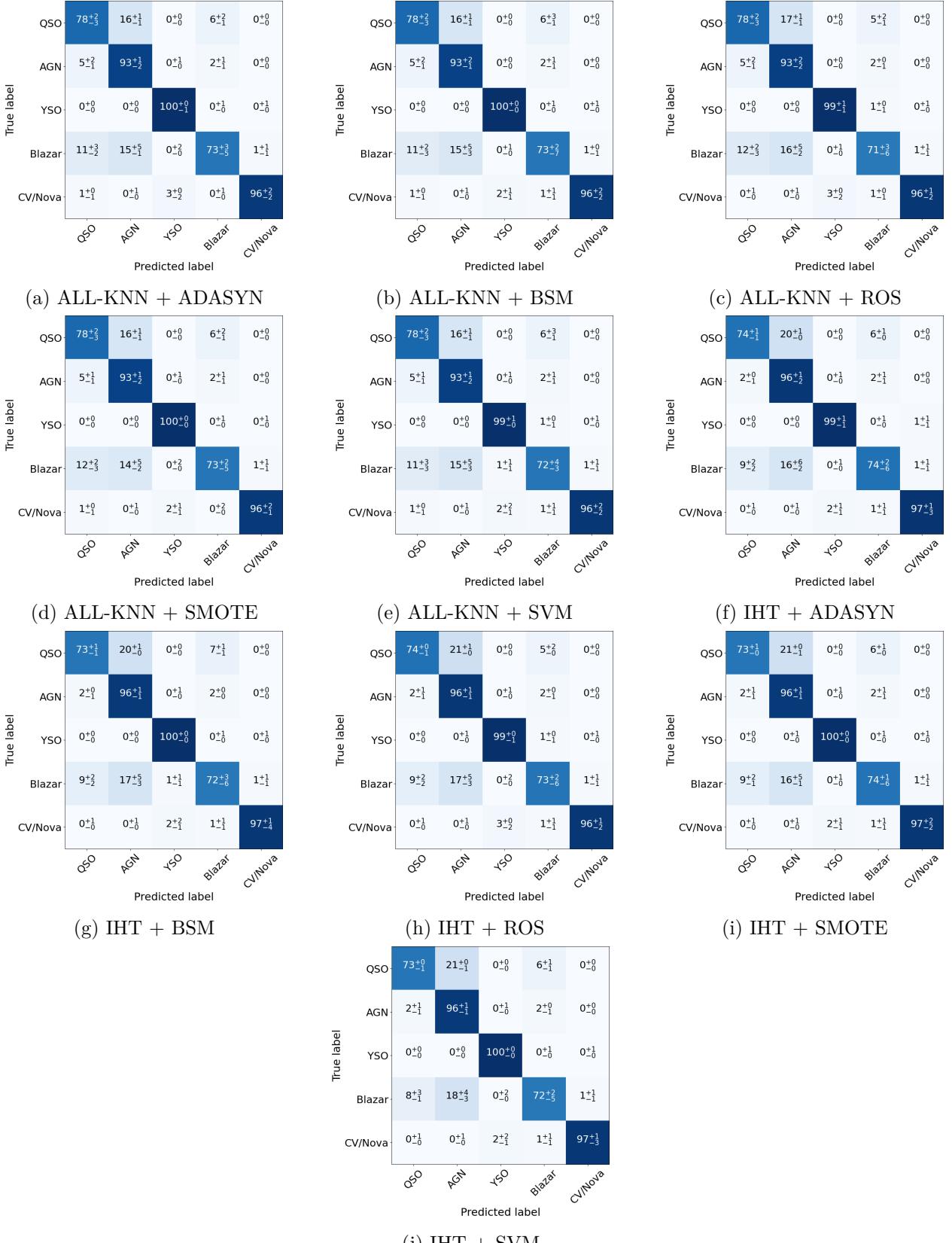


Figura 4.19: Matrices de confusión del clasificador Estocástico con distintas combinaciones de técnicas de sobremuestreo y submuestreo.

Debido a que QSO tiene más de 5 veces la cantidad de instancias que AGN, es que se prioriza esta ultima clase al decidir por un mejor modelo, en este caso XGBoost supera al modelo BRF. En particular para las combinaciones ALL-KNN con SMOTE, BSM y ADASYN e IHT con ADASYN para las cuales se obtuvieron los mejores resultados en Recall_M . Pero como se mencionó anteriormente, los valores obtenidos en las métricas no fueron mejores que los de BRF, por lo que no es posible demostrar diferencias significativas.

En la tabla 4.15 se presentan los resultados de las combinaciones de técnicas de muestreo que igualan o superen el mínimo valor de recall por clase establecido para el clasificador Transiente. La figura 4.20 muestra las correspondientes matrices de confusión.

Tabla 4.15: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Transiente con modelo XGB usando combinaciones de técnicas de balance de datos con mejores resultados.

Método Under	Método Over	Precision_M	Recall_M	F1-score_M	min Recall
IHT	SMOTE	0.52 ± 0.05	0.62 ± 0.04	0.53 ± 0.05	0.42 ± 0.13
IHT	ADASYN	0.54 ± 0.08	0.61 ± 0.05	0.53 ± 0.05	0.43 ± 0.08
NM	ADASYN	0.48 ± 0.05	0.59 ± 0.05	0.48 ± 0.05	0.42 ± 0.12

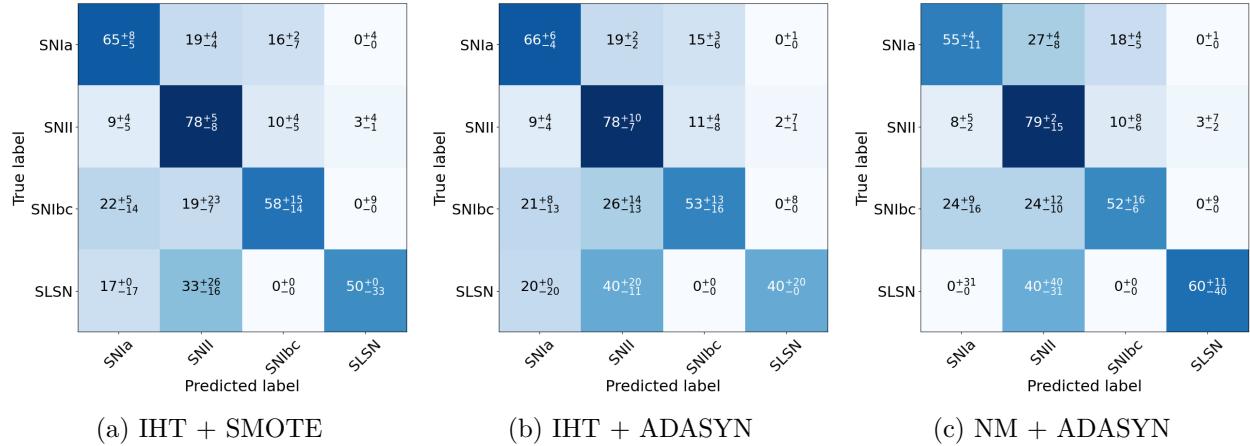


Figura 4.20: Matrices de confusión del clasificador Transiente con distintas combinaciones de técnicas de sobremuestreo y submuestreo.

Pocas combinaciones obtuvieron resultados aceptables (que hayan superado el mínimo recall establecido). Todos los resultados de la tabla 4.15 son evidentemente más bajos a lo obtenido con BRF, pero en comparación a XGBoost sin técnicas de balance se obtuvo un mejor Recall_M y mínimo, por lo que es probable que haya habido una mejora en cuanto al desbalance.

Las matrices de confusión de la figura 4.20 muestran una distribución bastante equitativa en la tasa de aciertos entre las distintas supernovas a clasificar, donde las principales diferencias se encuentran en la clase minoritaria SLSN, donde en su valor de mediana la combinación de NM con ADASYN obtuvo mayor tasa de aciertos.

Comparando los modelos analizados anteriormente, se tiene una clara mejora con respecto a XGBoost sin técnicas de balance, ya que disminuyó el sesgo hacia la clase mayoritaria SNIa. Pero con respecto a BRF el desempeño es peor, principalmente, porque este último modelo obtuvo muy buena tasa de aciertos en la clase minoritaria SLSN.

Se destaca que hubo una apreciable mejora en comparación a lo obtenido con anterioridad al utilizar Sampling (ROS+RUS) con XGBoost en el clasificador Transiente, esto muestra que una mejor selección de instancias a remover y generar instancias de manera sintética, supera la simple selección aleatoria para reducir y aumentar la cantidad de datos en este caso, donde en algunas clases hay muy pocas instancias.

De la experimentación con el modelo XGBoost utilizando combinaciones de técnicas de aumento y reducción de datos, se consideró que los clasificadores Periódico y Estocástico obtuvieron mejoras en el desempeño con respecto al modelo BRF. Sin embargo en cuanto a las métricas de evaluación, esta mejora se ve reflejada solo en el clasificador Periódico.

En el capítulo 5 se evaluará si la diferencia obtenida en las métricas tradicionales es estadísticamente significativa para la combinación CNN+ROS en el clasificador Periódico usando XGBoost y BRF.

4.6. Entrenamiento con técnicas híbridas

Las últimas técnicas de balance a nivel de datos implementadas y utilizadas para entrenar XGBoost corresponden a las clasificadas como híbridas, que combinan técnicas de sobremuestreo y submuestreo. Las técnicas híbridas utilizadas fueron:

- SPIDER [22]
- SOUP [23]

La tabla 4.17 muestra los resultados del clasificador Periódico con el modelo XGBoost entrenado con datos preprocesados con las técnicas híbridas implementadas. La figura 4.21 muestra las matrices de confusión resultantes.

Tabla 4.17: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Periódico con modelo XGB usando técnicas híbridas de balance de datos.

Método Híbrido	Precision_M	Recall_M	F1-score_M	min Recall
SOUP	0.72 ± 0.01	0.80 ± 0.02	0.75 ± 0.01	0.58 ± 0.05
SPIDER	0.73 ± 0.04	0.79 ± 0.02	0.75 ± 0.02	0.56 ± 0.01

En la tabla 4.17 se observa que tanto al utilizar SOUP como SPIDER, se obtuvieron mínimos Recall por clase por debajo del valor mínimo aceptable establecido (0.7) para el análisis de combinaciones de técnicas de sobremuestreo y submuestreo. Además, en ambos casos el Recall_M fue menor al obtenido por BRF en el clasificador Periódico.

Al comparar los valores obtenidos de Recall con BRF, nos generamos la idea preliminar de que las técnicas híbridas implementadas no son suficientes para tratar el desbalance con el modelo XGBoost para el clasificador de curvas periódicas.

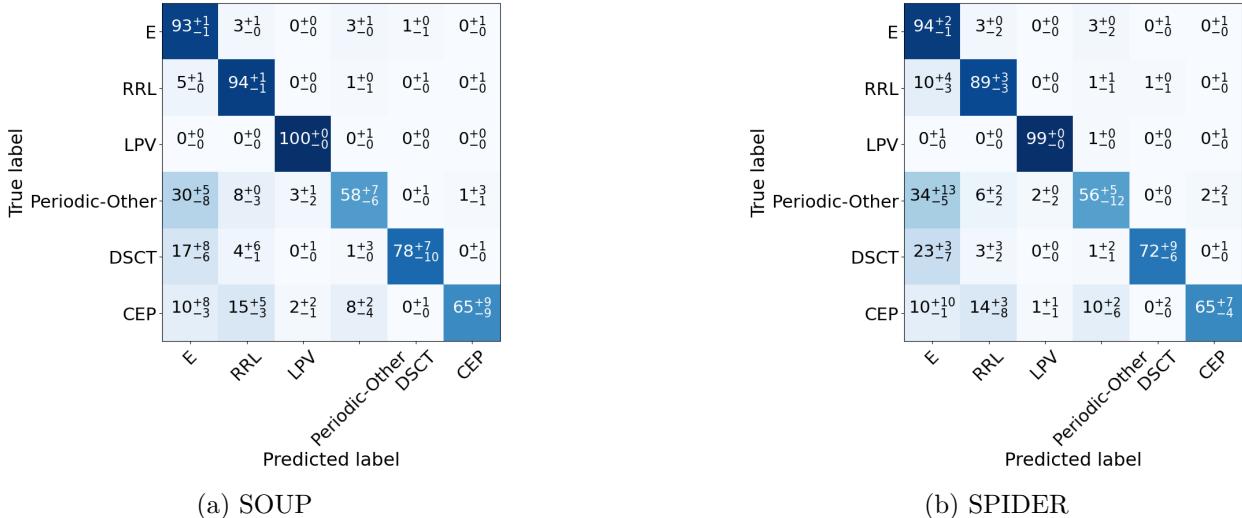


Figura 4.21: Matrices de confusión del clasificador Periódico con distintas técnicas híbridas de balanceo de datos.

En la figura 4.21 se observa en ambos casos un sesgo hacia las clases mayoritarias, obteniendo una tasa de aciertos notablemente más alta en estas clases que en las minoritarias. Además, se aprecia ver una gran confusión de las tres clases minoritarias hacia la clase mayoritaria E.

Sin embargo, al comparar los resultados de XGBoost con datos preprocesados con las técnicas híbridas, con respecto a lo obtenido en la tabla 4.3, además la matriz de confusión de la figura 4.3.b para XGBoost sin técnica alguna de balanceo, se evidencia una clara mejora y disminución del efecto del desbalance. De todas formas, no es suficiente para considerar su desempeño como comparable con lo obtenido con BRF para el clasificador Periódico.

La tabla 4.18 muestra los resultados del clasificador Estocástico utilizando XGBoost con técnicas híbridas. La figura 4.22 muestra las correspondientes matrices de confusión resultantes.

Tabla 4.18: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Estocástico con modelo XGB usando técnicas híbridas de balance de datos.

Método Híbrido	Precision _M	Recall _M	F1-score _M	min Recall
SOUP	0.88±0.01	0.86±0.01	0.87±0.01	0.60±0.03
SPIDER	0.87±0.01	0.87±0.01	0.87±0.01	0.64±0.03

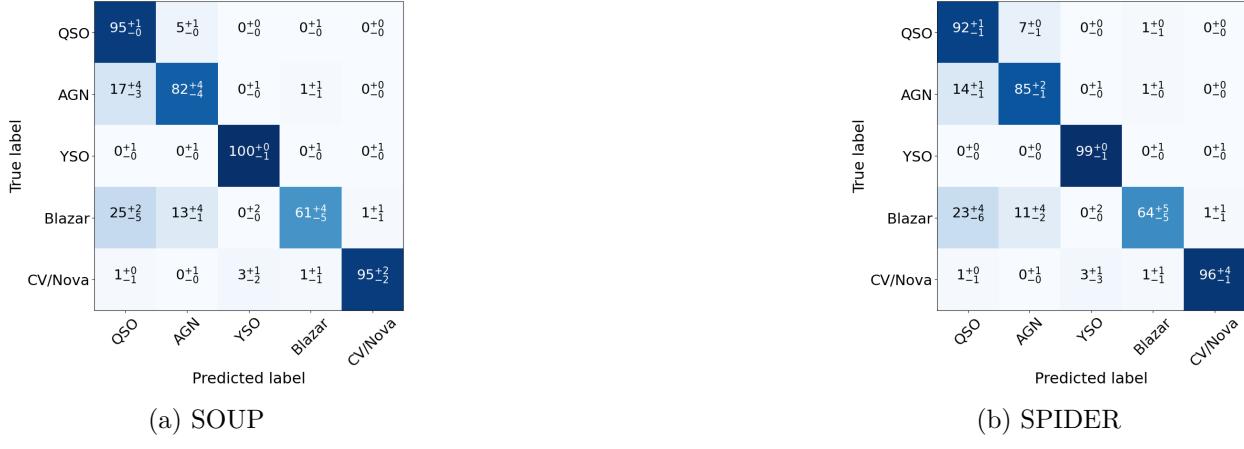


Figura 4.22: Matrices de confusión del clasificador Estocástico con distintas técnicas híbridas de balanceo de datos.

Nuevamente el mínimo Recall por clase obtenido es considerablemente menor al mínimo aceptable (0.7) para ambas técnicas utilizadas. Pero al comparar los resultados con los del caso de XGBoost sin técnicas de balance, se observa en las matrices de confusión de la figura 4.22 que aumentó significativamente la tasa de aciertos para la clase AGN, la cual junto con Blazar suelen ser las de peor desempeño en este clasificador. Sin embargo, debido al bajo desempeño en Blazar es que se considera mejor el modelo de BRF.

La tabla 4.19 muestra los resultados de utilizar las técnicas híbridas implementadas para preprocesar los datos de entrenamiento para XGBoost con el clasificador Transiente. En la figura 4.23 se presentan las matrices de confusión resultantes del clasificador Transiente con las técnicas híbridas.

Tabla 4.19: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Transiente con modelo XGB usando técnicas híbridas de balance de datos.

Método Híbrido	Precision _M	Recall _M	F1-score _M	min Recall
SOUP	0.68±0.09	0.54±0.03	0.57±0.04	0.15±0.05
SPIDER	0.61±0.10	0.53±0.04	0.54±0.05	0.16±0.07

Se presentan nuevamente los mismos indicios de desbalance, con un mínimo Recall por debajo al aceptable y Recall_M menor al obtenido con anterioridad, aunque levemente más alto a los resultados obtenidos con XGBoost sin técnicas de balance.

En las matrices de la figura 4.23 se observan claros sesgos hacia las clases mayoritarias SNIa y SNII. Además, en comparación a lo obtenido en la figura 4.3.d para el mismo modelo predictivo pero sin preprocesamiento de datos, se ven leves mejoras en cuanto a los aciertos en las clases minoritarias.

	SNIa	SNII	SNIbc	SLSN
SNIa	95 ⁺⁵ ₋₅	4 ⁺³ ₋₁	0 ⁺³ ₋₀	0 ⁺⁰ ₋₀
SNII	22 ⁺¹¹ ₋₄	73 ⁺⁵ ₋₈	3 ⁺² ₋₂	2 ⁺⁴ ₋₂
SNIbc	67 ⁺¹⁴ ₋₁₃	16 ⁺¹⁷ ₋₁₄	16 ⁺⁶ ₋₈	0 ⁺⁵ ₋₀
SLSN	18 ⁺¹⁸ ₋₁₈	45 ⁺¹⁰ ₋₁₉	0 ⁺⁰ ₋₀	36 ⁺¹⁰ ₋₁₈

(a) SOUP

	SNIa	SNII	SNIbc	SLSN
SNIa	89 ⁺⁶ ₋₄	9 ⁺³ ₋₃	2 ⁺⁴ ₋₂	0 ⁺¹ ₋₀
SNII	17 ⁺⁷ ₋₄	78 ⁺⁷ ₋₈	4 ⁺⁵ ₋₂	1 ⁺³ ₋₁
SNIbc	56 ⁺³⁰ ₋₁₂	23 ⁺¹⁶ ₋₁₁	21 ⁺²⁸ ₋₁₅	0 ⁺⁶ ₋₀
SLSN	20 ⁺²⁰ ₋₂₀	60 ⁺²⁰ ₋₃₁	0 ⁺⁰ ₋₀	20 ⁺³¹ ₋₁₁

(b) SPIDER

Figura 4.23: Matrices de confusión del clasificador Transiente con distintas técnicas híbridas de balanceo de datos.

A nivel general, para cada clasificador del nivel inferior usando XGBoost se obtuvo una mejora de desempeño al implementar las técnicas híbridas con respecto al escenario sin técnica alguna. Los clasificadores Periódico y Estocástico evidencian las mayores mejoras. Sin embargo en ambos casos no se obtuvo un mínimo Recall para aceptar preliminarmente las técnicas, por lo que los resultados para cada clasificador son peores a los obtenidos con la combinación de técnicas de sampleo y a lo obtenido con BRF.

4.7. Entrenamiento con combinación de técnicas de balance a nivel de datos y Cost Sensitive Learning

Debido a los buenos resultados obtenidos con XGBoost al ser entrenado con datos preprocesados con algunas combinaciones de técnicas de submuestreo y sobremuestreo y la técnica de *Cost Sensitive Learning* denominada Weight en la sección 4.3, es que se tomó la decisión de combinar ambas técnicas de distinto enfoque, una a nivel de datos y otro a nivel del algoritmo, para luego analizar si esta combinación es beneficiosa en comparación a cuando se implementa un solo tipo de técnica.

Los valores ponderadores de la función de pérdida para cada clase fueron seleccionados en base a la ecuación 4.1. Las combinaciones de técnicas de submuestreo y sobremuestreo a utilizar son las que en la sección 4.5 superaron el mínimo recall, incluyendo además la técnica *Random Undersampling* con las combinaciones de técnicas de aumento de datos implementadas.

El criterio de mínimo Recall por clase será nuevamente utilizado, ocupando los mismos valores por clasificador (Periódico: 0.7; Estocástico: 0.7; Transiente: 0.4).

Los mejores resultados obtenidos de las combinaciones de técnicas de balance para preprocesar los datos de entrenamiento de XGBoost ocupando la técnica Weight para el clasificador Periódico son los que se muestran la tabla 4.20, en donde fue la técnica de reducción de datos IHT con sus combinaciones con las técnicas de aumento ADASYN, ROS y SMOTE con las

que se obtuvo resultados que superen el mínimo Recall por clase establecido. En la figura 4.24 se muestran las correspondientes matrices de confusión.

Tabla 4.20: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Periódico con modelo XGB ocupando combinaciones de técnicas de balance de datos y técnica Weight.

Método Under	Método Over	Precision _M	Recall _M	F1-score _M	min Recall
IHT	ADASYN	0.56±0.00	0.82±0.01	0.56±0.00	0.72±0.01
IHT	ROS	0.56±0.00	0.83±0.01	0.58±0.00	0.74±0.01
IHT	SMOTE	0.56±0.00	0.83±0.01	0.56±0.01	0.72±0.01

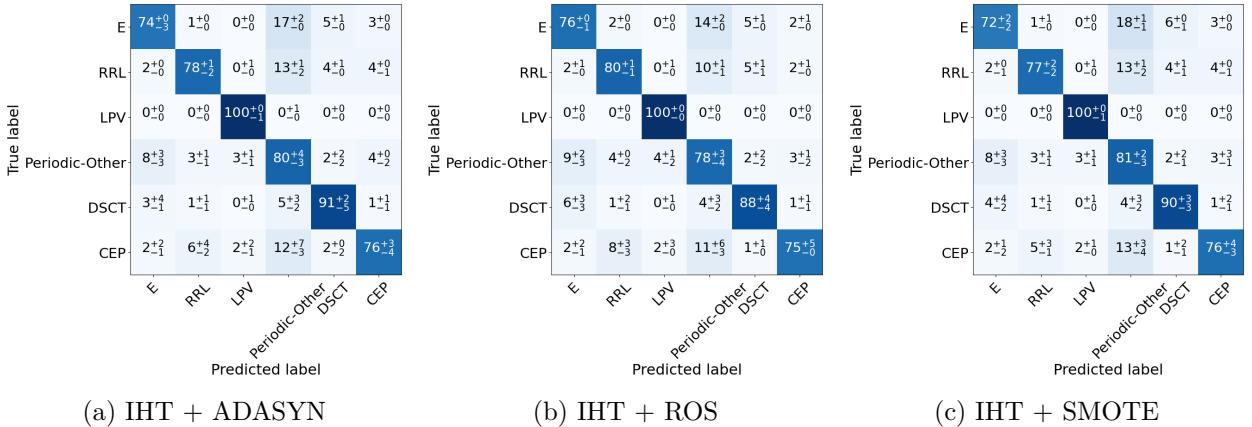


Figura 4.24: Matrices de confusión del clasificador Periódico ocupando XGBoost entrenado con técnicas de balance de datos y técnica *Cost Sensitive Learning*.

En la tabla 4.20 los valores obtenidos en las métricas son menores a lo obtenido con solo utilizar Weight, y son similares a los resultados de XGBoost solo con preprocesamiento de datos usando las mismas combinaciones de técnicas.

En las matrices de confusión no se evidencia la presencia de un sesgo hacia las clases mayoritarias, ya que las minoritarias tienen igual o mayor tasa de aciertos. En comparación con lo obtenido en la figura 4.7.a al usar solo la técnica Weight, se obtuvo una notable disminución en la cantidad de predicciones correctas en las clases mayoritarias E y RRL, pero a cambio, se obtuvo un aumento del 7% y 4% en la tasa de aciertos de las clases Periodic-Other y DSCT, respectivamente.

Comparando estos resultados con las matrices de confusión obtenidas para las mismas combinaciones de técnicas de sampleo en la figura 4.18, se observa que el incorporar la técnica *Cost Sensitive Learning* al entrenamiento resultó en una tasa de aciertos menor en las clases mayoritarias pero mayor en las minoritarias, pero en estas últimas con diferencias de menor porcentaje.

La implementación de ambos tipos de técnicas en el clasificador Periódico usando XGBoost ha demostrado favorecer la clasificación de las clases minoritarias sobre las mayoritarias.

El problema es que las diferencias positivas con respecto a los otros escenarios (con solo un tipo de técnica) constan de poco porcentaje, en cambio las negativas, principalmente la menor tasa de aciertos en las clases mayoritarias, son mucho más significativas.

Ya que en el presente problema las clases minoritarias no son tanto más relevantes que las mayoritarias, es que se concluye que no hay beneficio al implementar la combinación de técnicas a nivel de datos y algoritmo en conjunto con XGBoost para el clasificador Periódico, en comparación con estas mismas técnicas por separado.

La tabla 4.21 muestra los mejores resultados para el clasificador Estocástico para las combinaciones de técnicas de aumento y reducción de datos en conjunto con la técnica Weight. Las respectivas matrices de confusión se muestran en la figura 4.25.

Tabla 4.21: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Estocástico con modelo XGB ocupando combinaciones de técnicas de balance de datos y técnica Weight.

Método Under	Método Over	Precision_M	Recall_M	F1-score_M	min Recall
ALL-KNN	ADASYN	0.78 ± 0.01	0.89 ± 0.01	0.82 ± 0.01	0.73 ± 0.03
ALL-KNN	ROS	0.80 ± 0.01	0.88 ± 0.01	0.83 ± 0.01	0.70 ± 0.02
ALL-KNN	SMOTE	0.79 ± 0.01	0.89 ± 0.01	0.83 ± 0.01	0.73 ± 0.02
RUS	ADASYN	0.80 ± 0.01	0.89 ± 0.01	0.83 ± 0.01	0.71 ± 0.02
RUS	ROS	0.82 ± 0.01	0.89 ± 0.01	0.84 ± 0.01	0.70 ± 0.03
RUS	SMOTE	0.80 ± 0.01	0.89 ± 0.01	0.83 ± 0.01	0.72 ± 0.03

Los valores obtenidos para las distintas combinaciones aceptadas son bastante similares entre sí y con lo obtenido con Sampling y Weight en la tabla 4.6, en particular la combinación RUS+ROS (Sampling) con Weight en el que se obtuvo valores casi idénticos a los obtenidos al utilizar solo Weight con XGBoost.

Al comparar con el desempeño obtenido en la tabla 4.14 para las combinaciones de solo técnicas de muestreo, observa un aumento de Precision_M para cada caso al implementar Weight, pero los valores de Recall_M se mantuvieron.

En la figura 4.25 se muestran las matrices de confusión con escenarios muy similares a los obtenidos anteriormente. En comparación a Sampling se obtuvo una leve disminución en los aciertos de la clase QSO, pero a cambio se dió una menor confusión de instancias de la clase Blazar, resultando en un aumento en los aciertos de esta misma.

En relación a las matrices de confusión obtenidas con la combinación de solo técnicas de muestreo, al implementar Weight, se ve una clara disminución en predicciones de falsos AGN, lo cual si bien implica una reducción en la tasa de aciertos de esta clase, se trata de un porcentaje bastante bajo en todos los casos. Sin embargo, produce una mayor tendencia a la clase mayoritaria QSO, aumentando sustancialmente sus aciertos sin provocar menos clasificaciones correctas en las otras clases.

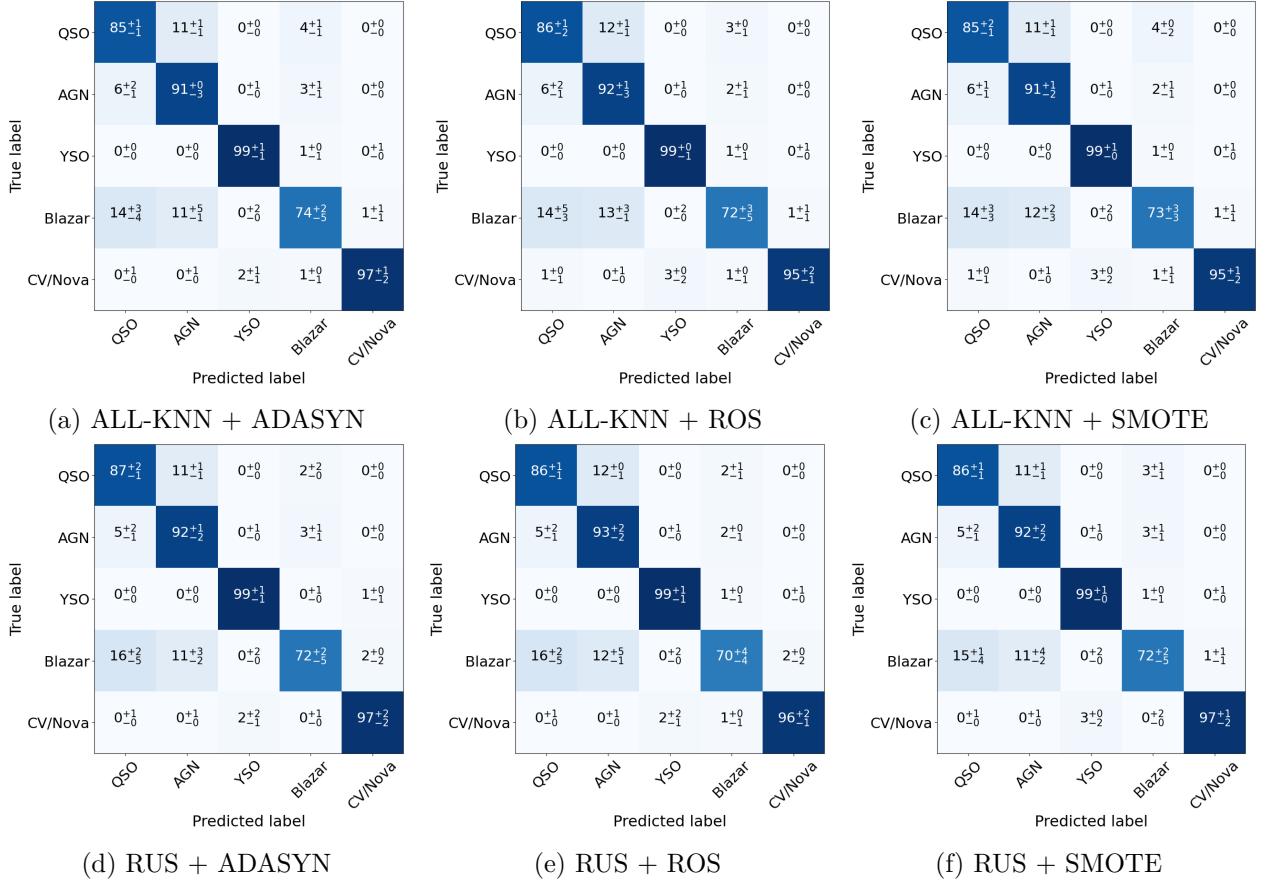


Figura 4.25: Matrices de confusión del clasificador Estocástico ocupando XGBoost entrenado con técnicas de balance de datos y técnica *Cost Sensitive Learning*.

Comparando XGBoost con solo Weight como técnica de balance y al implementar adicionalmente las técnicas a nivel de datos, la única diferencia que se advierte es en la tasa de aciertos de QSO y AGN. La combinación de técnicas obtuvo más aciertos para AGN y menos para QSO, caso contrario a lo que ocurre cuando solo utilizar Weight. Debido a que AGN tiene notablemente menos instancias que la clase QSO es que se estima como beneficioso el implementar ambos tipos de técnicas a la vez.

Los resultados en Recall_M obtenidos con XGBoost entrenado con técnicas de balanceo fueron muy similares a los de BRF en la tabla 4.2, con mejoras mínimas en todas las combinaciones, menos con ALL-KNN+ROS, por lo que más adelante se evaluará si estas diferencias son estadísticamente significativas.

Al comparar las matrices de confusión de la figura 4.25 con lo obtenido con BRF en la figura 4.1.c, las principales diferencias con las combinaciones de mayor Recall_M es que estas obtuvieron mayores tasas de aciertos en la clase AGN.

La tabla 4.22 muestra para el clasificador Transiente, los mejores resultados obtenidos con XGBoost entrenado con datos preprocesados con técnicas de balance e implementada la técnica Weight. La figura 4.26 muestra las respectivas matrices.

Tabla 4.22: Desempeño en métricas de Precision, Recall, F1-score y min Recall obtenidas para el clasificador Transiente con modelo XGB ocupando combinaciones de técnicas de balance de datos y técnica Weight.

Método Under	Método Over	Precision _M	Recall _M	F1-score _M	min Recall
IHT	ADASYN	0.51±0.05	0.61±0.05	0.49±0.04	0.45±0.15
IHT	SMOTE	0.50±0.04	0.61±0.08	0.49±0.06	0.43±0.14
RUS	ADASYN	0.54±0.04	0.63±0.06	0.55±0.04	0.45±0.15
RUS	SMOTE	0.54±0.05	0.64±0.06	0.56±0.05	0.42±0.15

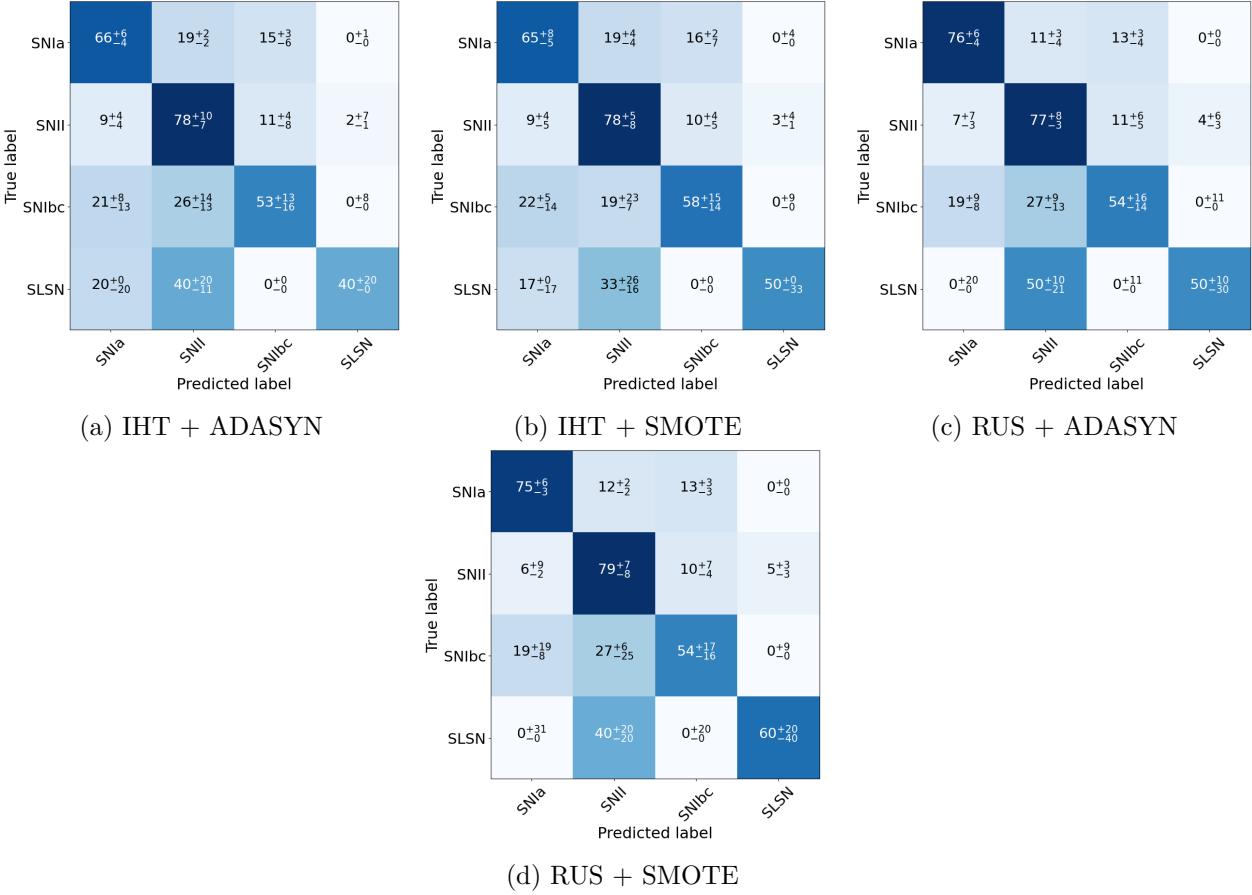


Figura 4.26: Matrices de confusión del clasificador Transiente ocupando XGBoost entrenado con técnicas de balance de datos y técnica Weight.

En la tabla 4.22 se observan valores bajos para las métricas tradicionales, pero en comparación con lo obtenido en la tabla 4.15 con solo técnicas de muestreo, las combinaciones con RUS podrían llegar a considerarse superiores debido a su mayor Recall_M.

Estos resultados en comparación al desempeño obtenido con XGBoost usando solo Weight en la tabla 4.7 es clara la disminución de desempeño al agregar el preprocesamiento de datos, esto principalmente por lo obtenido en el Recall en sus versiones macro y mínimo por clase.

Analizando las matrices de confusión en la figura 4.26 para las combinaciones con RUS como

método de reducción de datos, ya que éstas obtuvieron el máximo Recall_M , se observa una distribución desigual en los aciertos de las clases, presentando las mayoritarias valores significativamente más altos al de las minoritarias.

A pesar del sesgo hacia las clases mayoritarias obtenido con XGBoost al utilizar la combinación de técnicas de balance de datos, el desempeño de la combinación de RUS con SMOTE podría considerarse el mejor obtenido con estas combinaciones, ya que principalmente corresponde a la combinación con mayor tasa de aciertos en la clase minoritaria SLSN.

Sin embargo al analizar el escenario sin técnicas de muestreo, es decir, el modelo XGBoost con solo Weight, hubo una baja de desempeño en las clases minoritarias. Por esta razón que no considera beneficioso incorporar técnicas a nivel de datos al modelo XGBoost junto con la técnica Weight.

4.8. Entrenamiento con Focal Loss Cross-Entropy como función de pérdida

Como se mencionó anteriormente, una forma de abordar el aprendizaje de datos desbalanceados al nivel del algoritmo, es por medio de implementar la función de pérdida denominada Focal Loss Cross-Entropy [24].

Para poder implementar una nueva función de pérdida en el algoritmo XGBoost es necesario agregar de manera manual tanto el gradiente de la nueva función, como también su Hessiano, ambas con respecto al puntaje dado por el algoritmo para cada una de las clases a clasificar.

Para derivar la función objetivo con respecto a distintos puntajes (uno por cada clase), se ocupará la siguiente notación para la función de Focal Loss Cross-Entropy:

$$L(y, s) = - \sum_{i=1}^C y_i (1 - s_i)^\gamma \log(s_i), \quad (4.2)$$

donde C es el número total de clases, y es la etiqueta de la instancia a evaluar denotada de la forma *One Hot Encoded*, lo cual es un vector de largo C compuesto de ceros y un 1 en la posición de la clase real. Por ejemplo, si $C = 5$, entonces $y' = 3 \Rightarrow y = (0, 0, 1, 0, 0)^T$, y s es un vector de largo C que contiene las probabilidades de pertenencia para cada clase. La que la ecuación 4.2 es equivalente a:

$$L(y, s) = -(1 - s_t)^\gamma \log(s_t), \quad (4.3)$$

donde s_t es la probabilidad obtenida para la clase verdadera.

Para obtener las probabilidades s de pertenencia a cada clase en problemas multiclase, XGBoost utiliza la función de activación *softmax*, la cual recibe los puntajes z obtenidos por el algoritmo. La función *softmax* se define como sigue:

$$s_i = \frac{e^{z_i}}{\sum_{k=1}^C z_k} \quad (4.4)$$

Para obtener el gradiente y el Hessiano de la ecuación 4.2 con respecto a los puntajes z , se requiere obtener primero el gradiente de la función *softmax*. Este cálculo aparece en el desarrollo de la subsección 8.2.1 en el anexo, siendo el gradiente:

$$\frac{\partial s_i}{\partial z_j} = s_i \cdot (\mathbb{1}_{i=j} - s_j), \quad (4.5)$$

donde el operador $\mathbb{1}_{i=j}$ es igual a 1 cuando se cumple la condición de que $i = j$, en caso contrario es igual a 0. Ocupando la ecuación 4.5 junto con la regla de la cadena, se obtiene el gradiente de la función Focal Loss Cross-Entropy cuyo detalle se presenta en el desarrollo en la subsección 8.2.2 en el anexo, y su resultado es:

$$\frac{\partial L}{\partial z_j} = y_j \left[\gamma \log(s_j) (1 - s_j)^{\gamma-1} s_j - (1 - s_j)^\gamma \right] + s_j \sum_{i=1}^C y_i \left[(1 - s_i)^\gamma - \gamma \log(s_i) (1 - s_i)^{\gamma-1} s_i \right] \quad (4.6)$$

Una vez obtenido el gradiente, se procede a derivar esta expresión una vez más con respecto a z_j . El detalle de este cálculo se presenta en la subsección 8.2.3 del anexo, y el resultado final es:

$$\begin{aligned} \frac{\partial^2 L}{\partial z_j^2} &= y_j \gamma (1 - s_j)^\gamma s_j \left[\log(s_j) (1 - (\gamma - 1)(1 - s_j)^{-1} s_j) + 2 \right] (1 - s_j(1 - s_j)^{-1}) \\ &\quad + s_j (1 - s_j) \sum_{i=1}^C y_i \left[(1 - s_i)^\gamma - \gamma \log(s_i) (1 - s_i)^{\gamma-1} s_i \right] \\ &\quad + s_j^2 \gamma \sum_{i=1}^C y_i (1 - s_i)^{\gamma-1} s_i \left[\log(s_i) (1 - (\gamma - 1)(1 - s_i)^{-1} s_i) + 2 \right] \end{aligned} \quad (4.7)$$

El calculo de las ecuaciones 4.6 y 4.7 se entrega en forma de función al algoritmo XGBoost para así implementar la función de Focal Loss Cross Entropy.

Para entrenar los clasificadores, se utilizó el procedimiento de Nested Cross-Validation, seleccionando en los loops interiores los mismos hiperparámetros de XGBoost mencionados anteriormente, incluyendo en estos además el parámetro γ de la ecuación 4.2.

Ya que se está experimentando con una nueva función de pérdida, se aprovechará además de implementar la técnica Weight ya utilizada en secciones anteriores, ocupando la ecuación 4.1 para calcular las constantes ponderadoras de la función de pérdida según la clase a evaluar.

La tabla 4.23 muestra los resultados obtenidos para los clasificadores entrenados del nivel superior e inferior del clasificador de curvas de luz. se implementó por si sola la nueva función de pérdida y también cuando se utilizaron ponderaciones acorde a la clase evaluada (Weight). Además se muestran los resultados de la unión de ambos niveles del clasificador, esta unión es denominada LCC.

Tabla 4.23: Desempeño en métricas de Precision, Recall, F1-score y minRecall obtenidas para la unión de la capa superior e inferior del clasificador de curvas de luz ocupando la función Focal Loss Cross-Entropy con modelo XGB.

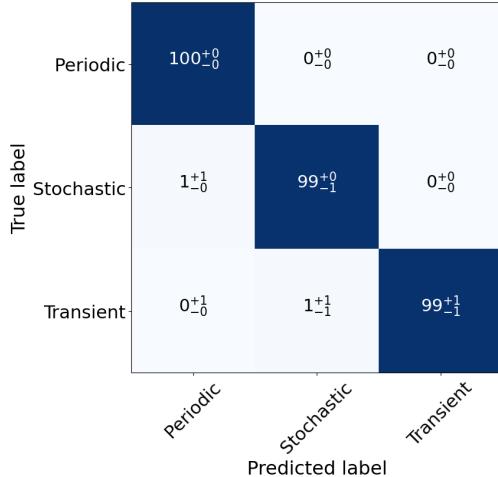
Clasificador	Ponderado	Precision_M	Recall_M	F1-score_M	min Recall
Hierarchical	Si	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
	No	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.98 ± 0.00
Periódico	Si	0.73 ± 0.02	0.82 ± 0.01	0.77 ± 0.02	0.58 ± 0.02
	No	0.90 ± 0.01	0.74 ± 0.01	0.80 ± 0.01	0.32 ± 0.03
Estocástico	Si	0.86 ± 0.01	0.88 ± 0.01	0.87 ± 0.01	0.67 ± 0.03
	No	0.92 ± 0.01	0.85 ± 0.01	0.88 ± 0.01	0.59 ± 0.03
Transiente	Si	0.57 ± 0.06	0.64 ± 0.06	0.59 ± 0.06	0.42 ± 0.08
	No	0.67 ± 0.10	0.50 ± 0.05	0.53 ± 0.06	0.08 ± 0.08
LCC	Si	0.71 ± 0.02	0.77 ± 0.02	0.73 ± 0.02	0.41 ± 0.09
	No	0.83 ± 0.02	0.68 ± 0.01	0.73 ± 0.02	0.06 ± 0.07

Para los resultados de los clasificadores individuales del nivel inferior y su unión con junto al nivel superior (LCC) se obtuvieron valores de Precision_M notablemente más altos y valores de Recall_M más bajos al no implementar Weight en la función de pérdida, en comparación a cuando si fue incluida la ponderación.

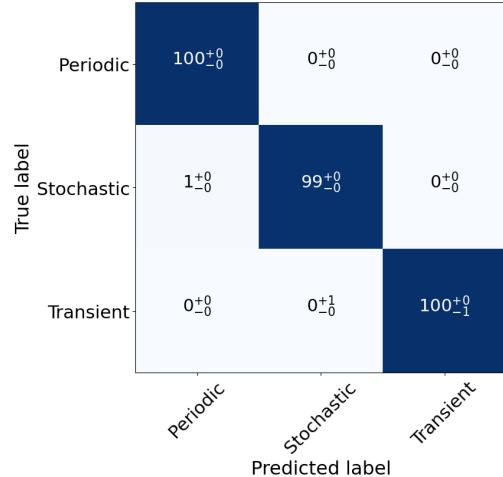
Tal como se ha mencionado anteriormente, esto es un claro signo de desbalance en las predicciones del modelo. Además, los resultados de mínimo Recall por clase de la tabla 4.23 indican proporciones bastante desiguales en la tasa de aciertos para los clasificadores entrenados con solo la función FLCE.

Las figuras 4.27, 4.28 muestran las correspondientes matrices de confusión resultantes de los clasificadores con el modelo XGBoost entrenado con la función de pérdida *Focal Loss Cross Entropy*. Además las matrices de confusión de las figuras 4.31 y 4.32 corresponden a la salida obtenida para el nivel inferior del clasificador de curvas de luz.

Tal como se esperaba de la poca diferencia en los valores de las métricas para el clasificador del nivel superior, las predicciones fueron bastante similares entre los dos casos de FLCE a comparar, obteniendo además errores mínimos en la clasificación, tal como se ha dado en cada entrenamiento del modelo XGBoost con distintas técnicas de balance y sin éstas.

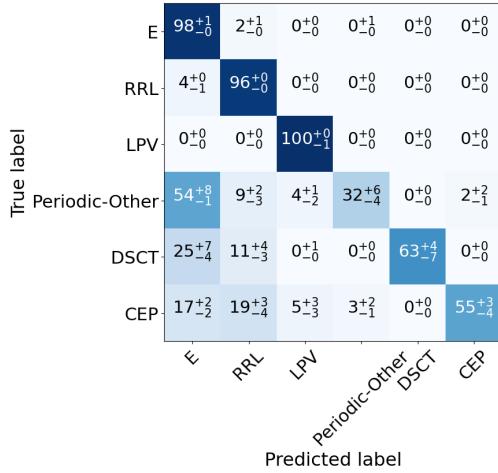


(a) Sin ponderación

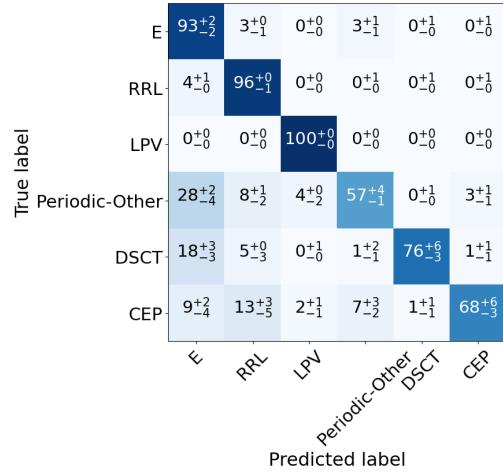


(b) Con ponderación

Figura 4.27: Matrices de confusión para el clasificador jerárquico con XG-Boost entrenado con la función de pérdida *Focal Loss Cross Entropy*.



(a) Sin ponderación



(b) Con ponderación

Figura 4.28: Matrices de confusión para el clasificador Periódico con XG-Boost entrenado con la función de pérdida *Focal Loss Cross Entropy*.

Se observa que en la figura 4.28.a para el clasificador Periódico sin utilizar Weight hay un claro sesgo hacia las clases mayoritarias, en donde la clase E y RRL se llevan gran parte de las predicciones. Comparando con la matriz de confusión obtenida con XGBoost sin técnicas de balance y con Cross-Entropy como función de pérdida, en la figura 4.3.b mostrada, se aprecia una situación prácticamente idéntica.

Las predicciones realizadas en el conjunto de test cuando se incluyó la ponderación en la función de pérdida fueron bastante más acertadas para las clases de minoritarias Periodic-Other, DSCT y CEP. Aun así, el sesgo hacia las clases de mayoritarias es alto, caso que no ocurrió cuando se utilizó la función de pérdida Cross-Entropy con Weight.

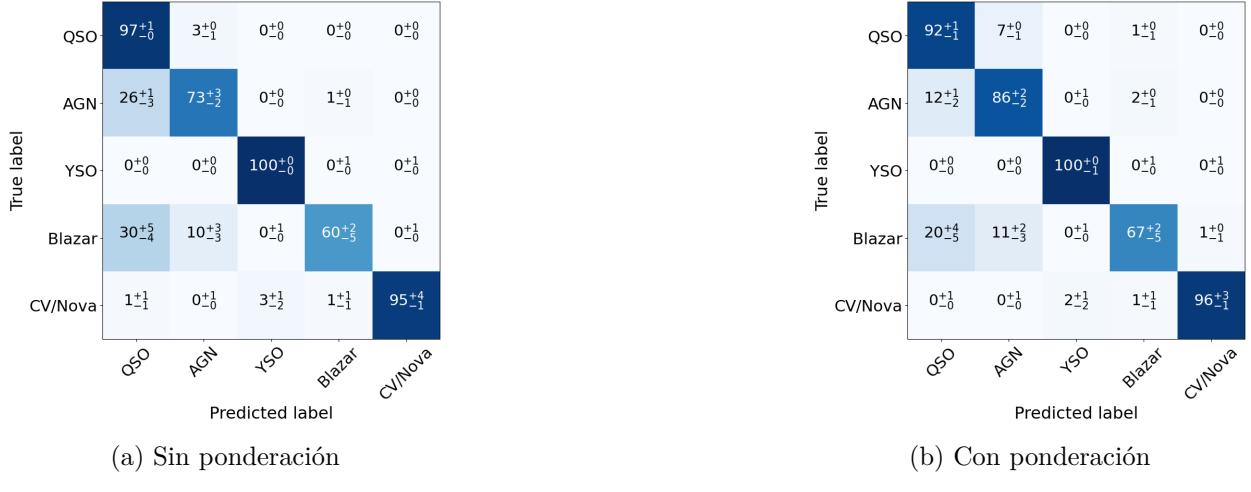


Figura 4.29: Matrices de confusión para el clasificador Estocástico con XG-Boost entrenado con la función de pérdida *Focal Loss Cross Entropy*.

Las predicciones obtenidas con en el clasificador Estocástico son similares a las del clasificador Periódico, en donde el utilizar la función FLCE en el entrenamiento de XGBoost sin ponderar no se obtiene ningún cambio significativo con respecto a cuando se utilizó Cross-Entropy sin técnicas de balance. Al comparar los resultados obtenidos con la técnica Weight, FLCE presenta menor tasa de aciertos en las clases minoritarias, siendo Blazar la clase de mayor dificultad de aprendizaje.

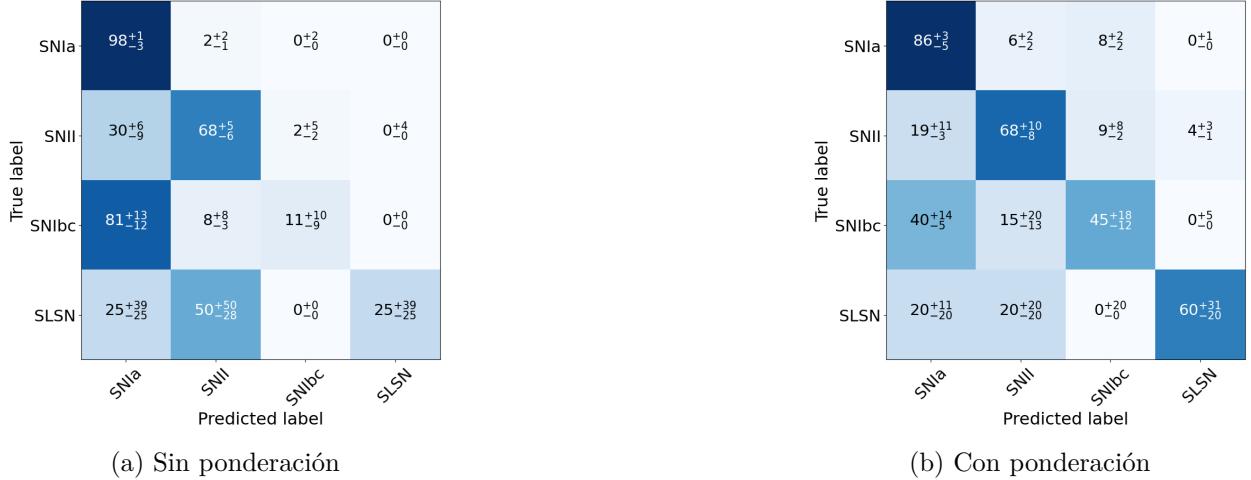


Figura 4.30: Matrices de confusión para el clasificador Transiente con XG-Boost entrenado con la función de pérdida *Focal Loss Cross Entropy*.

Al implementar FLCE en el entrenamiento de XGBoost para el clasificador Transiente se obtuvieron resultados que, en ambos casos, con y sin implementar Weight, fueron peores a los obtenidos al utilizar Cross-Entropy.

Por un lado, al no utilizar valores ponderadores en las funciones de pérdida, las predicciones con FLCE presentaron una mayor confusión al predecir instancias como falsos SNIa en comparación con CE. Por otro lado, la confusión entre las clases SNII y SLSN disminuyó,

aumentando la tasa de aciertos en la clase minoritaria SLSN.

Al ponderar la función de pérdida hubo algunos casos donde la función Focal Loss Cross-Entropy obtuvo un 10 % más en la tasa de aciertos de la clase mayoritaria SNIa con respecto a CE, pero con 20 % menos en la clase SNIbc que es la segunda de menor proporción. De igual forma los resultados en la clase minoritaria SLSN son muy similares, esto teniendo en cuenta que los percentiles mayores y menores son iguales, y que si bien, con FLCE se obtuvo un 10 % menos de aciertos, ese porcentaje corresponde a solo dos instancias.

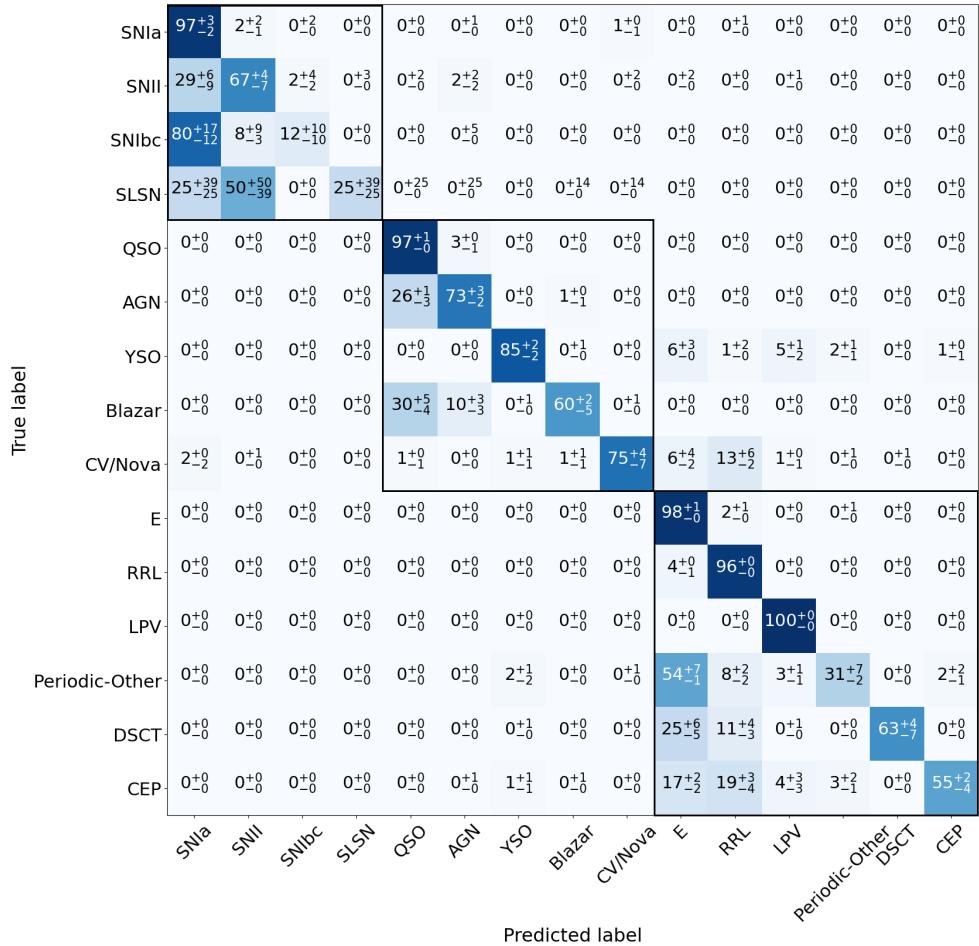


Figura 4.31: Matriz de confusión para la unión de capas del clasificador de curvas de luz ocupando XGBoost entrenado con la función de pérdida *Focal Loss Cross Entropy* sin ponderar.

	SNIA	SNII	SNIbc	SLSN	QSO	AGN	YSO	Blazar	CV/Nova	E	RRL	LPV	Periodic-Other	DSCT	CEP
True label	85 ⁺⁵ ₋₃	7 ⁺¹ ₋₃	8 ⁺² ₋₂	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
SNIA	85 ⁺⁵ ₋₃	7 ⁺¹ ₋₃	8 ⁺² ₋₂	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
SNII	18 ⁺¹² ₋₂	67 ⁺¹⁰ ₋₉	9 ⁺⁸ ₋₂	4 ⁺³ ₋₁	0 ⁺⁰ ₋₀	1 ⁺² ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀				
SNIbc	41 ⁺¹⁵ ₋₅	12 ⁺²⁴ ₋₁₀	47 ⁺¹⁷ ₋₁₃	0 ⁺⁶ ₋₀	0 ⁺⁰ ₋₀	0 ⁺³ ₋₀	0 ⁺³ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
SLSN	20 ⁺¹¹ ₋₂₀	20 ⁺²⁰ ₋₂₀	0 ⁺¹¹ ₋₀	60 ⁺³¹ ₋₃₁	0 ⁺⁰ ₋₀	0 ⁺¹¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺²⁰ ₋₀	0 ⁺¹¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
QSO	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	92 ⁺¹ ₋₁	7 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
AGN	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	12 ⁺¹ ₋₂	86 ⁺² ₋₂	0 ⁺⁰ ₋₀	2 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
YSO	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	90 ⁺¹ ₋₃	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	1 ⁺³ ₋₁	1 ⁺⁰ ₋₁	4 ⁺¹ ₋₂	3 ⁺² ₋₁	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₁
Blazar	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	20 ⁺⁴ ₋₅	11 ⁺² ₋₃	0 ⁺¹ ₋₀	67 ⁺⁷ ₋₅	1 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
CV/Nova	2 ⁺¹ ₋₁	0 ⁺¹ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺¹ ₋₀	1 ⁺¹ ₋₁	1 ⁺¹ ₋₁	80 ⁺⁶ ₋₅	4 ⁺² ₋₂	8 ⁺⁵ ₋₂	1 ⁺¹ ₋₁	0 ⁺² ₋₀	1 ⁺¹ ₋₁	1 ⁺⁰ ₋₁
E	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	93 ⁺¹ ₋₂	3 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀	3 ⁺¹ ₋₁	0 ⁺¹ ₋₀	0 ⁺¹ ₋₀
RRL	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	4 ⁺¹ ₋₀	96 ⁺¹ ₋₁	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀
LPV	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	100 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
Periodic-Other	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	3 ⁺³ ₋₁	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₁	27 ⁺³ ₋₃	8 ⁺¹ ₋₂	2 ⁺¹ ₋₁	56 ⁺⁴ ₋₂	0 ⁺¹ ₋₀	3 ⁺¹ ₋₁
DSCT	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	18 ⁺³ ₋₃	5 ⁺⁰ ₋₃	0 ⁺¹ ₋₀	1 ⁺² ₋₂	76 ⁺⁶ ₋₃	1 ⁺⁰ ₋₁
CEP	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	2 ⁺⁴ ₋₁	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	9 ⁺² ₋₄	13 ⁺³ ₋₅	1 ⁺¹ ₋₁	6 ⁺⁴ ₋₁	1 ⁺¹ ₋₁	68 ⁺⁴ ₋₅

Figura 4.32: Matriz de confusión para la unión de capas del clasificador de curvas de luz ocupando XGBoost entrenado con la función de pérdida *Focal Loss Cross Entropy* con ponderar.

En todo el nivel inferior del clasificador de curvas de luz, es clara la utilidad de implementar la técnica *Cost Sensitive Learning* con la función de pérdida Focal Loss Cross-Entropy para enfrentar el efecto del desbalance de datos. En las matrices de confusión de las figuras 4.31 y 4.32 se observa que las magnitudes de las confusiones hacia la clase mayoritaria SNIA entre las transientes, QSO para las estocásticas y E para las periódicas, disminuye notablemente cuando se utiliza Weight.

Implementar Focal Loss Cross-Entropy como función de pérdida para XGBoost no evidenció ningún beneficio por sobre lo ya obtenido al utilizar Cross-Entropy, es más, el sesgo hacia las clases mayoritarias aumentó, lo cual fue contrario a lo esperado.

Una posible razón de porque FLCE no fue de ayuda en el presente escenario que las probabilidades de pertenencia que asigna el modelo predictivo se tienen que distribuir hasta entre 6 clases a la vez. Por lo que es esperable que no sean muchas las curvas de luz que tengan una probabilidad alta (0.6) de pertenecer a su verdadera clase, que es cuando se espera que FLCE influya más.

Es posible que a buena parte de las curvas de luz se les esté asignando probabilidades simila-

res entre las distintas clases, en particular a las más conflictivas como las minoritarias, por lo que la función FLCE estaría afectando en una magnitud similar a varias de estas instancias, sin poder distinguir suficientemente entre instancias de fácil o difícil clasificación.

Un fenómeno interesante es que a mayor cantidad de clases a clasificar, peor fue el desempeño de FLCE (utilizando Weight) en comparación con lo obtenido con CE. Para el clasificador Periódico en que se clasifican 6 clases se evidenció la mayor disminución de la tasa de aciertos en las clases minoritarias, luego en el clasificador Estocástico con 5 clases en menor medida, y para el clasificador Transiente con 4 clases un poco más que el Estocástico, pero hay que tener en cuenta que es la clase jerárquica con mayor desbalance. En el clasificador del nivel superior, en el que se busca distinguir entre 3 clases no se obtuvieron diferencias significativas de desempeño.

Es posible que implementar la función Focal Loss Cross-Entropy sea más útil en los problemas de clasificación binaria, donde la probabilidad se reparte entre dos clases.

Capítulo 5

Análisis Estadístico de Resultados

Todos los resultados expuestos en el capítulo 4 corresponden a medidas de tendencia central, como los promedios (caso de las métricas) o a las medianas (valores en la matriz de confusión) de los resultados obtenidos con los conjuntos de test para las 10 veces que se entrena cada modelo (10 K-fold).

Cada valor medio tiene cierta de desviación, por lo que las comparaciones directas entre valores no son del todo concluyentes. Por ello, es que se propuso en el capítulo 3 incorporar test estadísticos para encontrar diferencias cuantitativas en los desempeños de los modelos a comparar.

A continuación se evalúan las diferencias de desempeño entre lo obtenido con BRF y XGBoost en conjunto con alguna técnica de balance cuyos resultados se hayan estimados como similares o mejores al modelo de Balanced Random Forest.

En la prueba con las primeras técnicas de balance y distintas métricas de evaluación en el entrenamiento, se concluyó que utilizando Recall_M se obtuvieron mejores resultados. Por lo que se procede a evaluar si la diferencia de desempeño de XGBoost para dicha métrica y técnicas de balance Weight y Sampling es significativa en comparación con BRF.

La tabla 5.1 muestra los resultados del test de normalidad (Gaussianidad) para la diferencia de desempeño entre XGBoost con técnicas de balance de datos (Weight y Sampling) y BRF con las métricas tradicionales.

Se obtuvieron para casi todas las métricas tradicionales, *p-values* mayores al nivel de confianza establecido ($\alpha = 0.05$) para todos los clasificadores de la capa inferior (y su combinación). Se acepta así la hipótesis nula que indica que las diferencias se distribuyen normalmente, por lo que realizar el test de *t-student* es válido.

Tabla 5.1: Valores *p-value* obtenidos del test de normalidad para la diferencia de desempeño entre XGBoost con técnicas de balance y BRF, para los clasificadores individuales de ambos niveles del clasificador de curvas de luz y su unión (LCC).

Clasificador	Métrica	Precision_M	Recall_M	F1 score_M
Periódico	Weight	0.444	0.390	0.570
	Sampling	0.685	0.081	0.911
Estocástico	Weight	0.955	0.719	0.987
	Sampling	0.005	0.115	0.098
Transiente	Weight	0.273	0.755	0.145
	Sampling	0.737	0.230	0.652
LCC	Weight	0.827	0.709	0.744
	Sampling	0.528	0.955	0.872

Las diferencias de Precision_M en el clasificador Estocástico usando Sampling fue el único caso en que se rechazó la hipótesis nula, aceptando la hipótesis alternativa de que los datos no se distribuyen normalmente, por tanto, para esta diferencia se realiza el *Permutation Test*.

Las tablas 5.2 y 5.7 muestran los resultados del test *t-student* y del *Permutation test* respectivamente.

Tabla 5.2: Valores *p-value* obtenidos del test de *t-student* para la diferencia de desempeño entre XGBoost con técnicas de balance y BRF, para los clasificadores individuales de ambos niveles del clasificador de curvas de luz y su unión (LCC).

Clasificador	Métrica	Precision_M	Recall_M	F1 score_M
Periódico	Weight	2.9e-8	3.7e-9	3.7e-9
	Sampling	1.8e-11	8.9e-11	5.2e-12
Estocástico	Weight	1.2e-5	1.0e-4	3.7e-6
	Sampling	X	4.9e-4	2.9e-9
Transiente	Weight	0.012	0.218	0.003
	Sampling	0.006	0.003	0.016
LCC	Weight	6.4e-9	1.6e-5	2.2e-9
	Sampling	1.9e-7	0.052	9.3e-8

Tabla 5.3: Valores *p-value* obtenidos del *Permutation test* para la diferencia de desempeño entre XGBoost con técnicas de balance y BRF, para los clasificadores individuales de ambos niveles del clasificador de curvas de luz y su unión (LCC).

Clasificador	Métrica	Precision_M	Recall_M	F1 score_M
Periódico	Weight	0.002	0.002	0.002
	Sampling	0.002	0.002	0.002
Estocástico	Weight	0.002	0.002	0.002
	Sampling	0.002	0.002	0.002
Transiente	Weight	0.023	0.203	0.008
	Sampling	0.010	0.002	0.002
LCC	Weight	0.002	0.002	0.002
	Sampling	0.002	0.058	0.002

En las tablas 5.2 y 5.7 se puede ver que para casi todas las métricas y clasificadores se tuvieron valores de probabilidad menores a 0.05, indicando que las diferencias si son significativas y por tanto, XGBoost supera a BRF.

Sin embargo para el clasificador Transiente, además de haber obtenido *p-values* cercanos al nivel de significancia en Precision_M y F1 score_M , para Recall_M se confirma que BRF supera a XGBoost con la técnica Sampling. Para las diferencias de BRF con XGBoost Weight se obtuvo una probabilidad del test de 0.218, aceptando así la hipótesis nula de que la diferencia no es significativa para Recall_M .

Para la unión de niveles del clasificador (LCC), la única diferencia de resultados con el modelo de BRF que no se estimó como cuantitativa fue la de XGBoost con Sampling en Recall_M .

De esta forma, se confirma parcialmente la idea preliminar mencionada, que XGBoost con técnicas de balanceo y Recall_M como métrica de evaluación supera el desempeño obtenido con BRF. Solo en los clasificadores Periódico y Estocástico las diferencias de todas las métricas se estimaron como estadísticamente cuantitativas. Para la métrica Recall_M en el clasificador Transiente la diferencia no fue significativa, y como se ha mencionado con anterioridad, está es la de mayor importancia en escenarios con desbalance de datos.

De la experimentación utilizando combinaciones de técnicas de aumento y reducción de datos, en cuanto a las métricas de evaluación, solo se vio reflejada una mejora en el clasificador Periódico con XGBoost, en particular para combinación de técnicas CNN y ROS.

Por tanto, se procede a evaluar si la diferencia obtenida en las métricas tradicionales es estadísticamente significativa con respecto a BRF en el clasificador Periódico. En la tabla 5.4 se muestra el resultado del test de normalidad para la diferencia de desempeño con el modelo BRF en el clasificador Periódico.

Tabla 5.4: Valores *p-value* obtenidos del test de normalidad para la diferencia de desempeño entre XGBoost con CNN+ROS y BRF para el clasificador Periódico.

Clasificador	Combinación	Precision_M	Recall_M	F1 score_M
Periódico	CNN+ROS	0.696	0.867	0.903

Los valores de probabilidad obtenidos con el test de normalidad fueron en cada caso superiores al nivel de significancia establecido, confirmando así que los datos se distribuyen de manera normal, cumpliendo así la condición para utilizar el test de *t-student*. En la tabla 5.5 se muestran los resultados de dicho test, para verificar la presencia de una diferencia significativa en el desempeño de los modelos a comparar.

Tabla 5.5: Valores *p-value* obtenidos del test de *t-student* para la diferencia de desempeño entre XGBoost con CNN+ROS y BRF para el clasificador Periódico.

Clasificador	Combinación	Precision_M	Recall_M	F1 score_M
Periódico	CNN+ROS	9.1e-8	1.3e-5	9.4e-8

Como se puede ver en la tabla 5.5, para cada métrica en comparación, se obtuvieron *p-values* menores a 0.05, confirmando que la diferencia en desempeño es significativa. El desempeño del clasificador Periódico utilizando XGBoost junto a la combinación CNN+ROS para el preprocesamiento de los datos de entrenamiento supera los resultados obtenidos con *Balanced Random Forest*.

En la experimentación con técnicas de aumento y reducción de datos en conjunto con *Cost Sensitive Learning*, solo para el clasificador Estocástico se consideró beneficioso utilizar estas técnicas en conjunto con el modelo XGBoost.

Se procede a evaluar si el desempeño del modelo XGBoost en conjunto con la técnica Weight y entrenado con las combinaciones de Recall_M más alto de la tabla 4.21 es significativamente mayor a lo obtenido en el clasificador Estocástico con BRF. Primero se realiza el test de normalidad para la diferencia en las métricas tradicionales.

Tabla 5.6: Valores *p-value* obtenidos del *normality test* para la diferencia de desempeño entre modelos de XGBoost Weight y BRF para el clasificador Estocástico.

Clasificador	Combinación	Precision_M	Recall_M	F1 score_M
Estocástico	ALL-KNN+ADASYN	0.477	0.264	0.877
	ALL-KNN+SMOTE	0.910	0.666	0.700
	RUS+ADASYN	0.238	0.742	0.141
	RUS+ROS	0.186	0.926	0.190
	RUS+SMOTE	0.531	0.349	0.547

En base a los resultados de la tabla 5.6 se confirma que las diferencias evaluadas distribuyen de manera normal, debido a que en cada caso se obtuvo un *p-value* mayor a 0.05, por lo que

se procede a realizar el test de *t-student*.

Tabla 5.7: Valores *p-value* obtenidos con el *Permutation test* para la diferencia de desempeño entre modelos de XGBoost Weight y BRF para el clasificador Estocástico.

Clasificador	Combinación	Precision _M	Recall _M	F1 score _M
Estocástico	ALL-KNN+ADASYN	7.1e-4	1.4e-4	0.002
	ALL-KNN+SMOTE	9.9e-5	1.0e-4	2.6e-4
	RUS+ADASYN	7.2e-6	1.4e-5	9.9e-6
	RUS+ROS	3.3e-8	3.7e-7	6.2e-8
	RUS+SMOTE	3.9e-6	4.0e-6	1.3e-5

Ya que para cada combinación de técnicas de muestreo junto con la técnica de *Cost Sensitive Learning* en el modelo XGBoost los *p-values* son menores al nivel de significancia, se confirma que la diferencia de desempeño de XGBoost con respecto al modelo BRF es significativa, siendo superior el primero.

Finalmente, debido a que la técnica *Cost Sensitive Learning* fue la de mejor desempeño en comparación a otras en conjunto con XGBoost para todos los clasificadores, y significativamente superior al desempeño obtenido por BRF excepto en el clasificador Transiente, es que se decide por esta técnica como la mejor para hacer frente a los efectos del desbalance de datos al momento de entrenar el algoritmo XGBoost.

Capítulo 6

Evaluación de mejores modelos con datos nuevos de ALeRCE

En esta sección se evalua tanto el desempeño de los clasificadores ocupando el modelo *Balanced Random Forest* como también usando los modelos entrenados de XGBoost con las técnicas de balance (o combinación de éstas) que se consideren mejores. La evaluación se realiza utilizando curvas de luz nuevas de ALeRCE, las cuales en el presente trabajo no han sido utilizadas para el entrenamiento de modelos ni tampoco para su testeo.

Las técnicas de balance consideradas como mejores son aquellas que en conjunto con modelo XGBoost han obtenido un mejor desempeño las métricas de evaluación y tasas de verdaderos positivos en sus matrices de confusión (teniendo en cuenta el desbalance).

Las técnicas a evaluar junto al modelo XGBoost en el test final con datos nuevos de ALeRCE son las siguientes para cada clasificador:

- Hierarchical
 - Weight (*Cost-Sensitive Learning*)
 - Sampling (ROS + RUS)
 - Sin técnicas
- Periódico
 - Weight (*Cost-Sensitive Learning*)
 - Sampling (RUS + ROS)
 - CNN + ROS
- Estocástico
 - Weight (*Cost-Sensitive Learning*)
 - Sampling (ROS + RUS)
 - Weight junto a ALL-KNN + ADASYN
 - Weight junto a ALL-KNN SMOTE
 - Weight junto a RUS + ADASYN

- Weight junto a RUS + ROS
- Weight junto a RUS + SMOTE
- Transiente
 - Weight (*Cost-Sensitive Learning*)

La tabla 6.1 muestra los resultados de las técnicas evaluadas con XGBoost y del modelo de BRF para el nivel superior del clasificador de curvas de luz con los datos del nuevo conjunto de test, de esta se observa una clara baja de desempeño en cada caso en comparación a lo obtenido en las curvas de luz antiguas de los conjuntos de test. La mayor diferencia fue la obtenida por BRF, en especial en los valores de Precision_M y F1 score_M .

Tabla 6.1: Desempeño en métricas de Precision, Recall, F1 score y min Recall obtenidas para el clasificador jerárquico usando las curvas de luz nuevas de ALeRCE.

Modelo	Técnica	Precision_M	Recall_M	F1 score_M	min Recall_M
BRF	None	0.83 ± 0.00	0.95 ± 0.00	0.88 ± 0.00	0.92 ± 0.00
XGB	None	0.98 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.94 ± 0.00
XGB	Weight	0.97 ± 0.01	0.97 ± 0.00	0.97 ± 0.00	0.95 ± 0.00
XGB	Sampling	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.95 ± 0.00

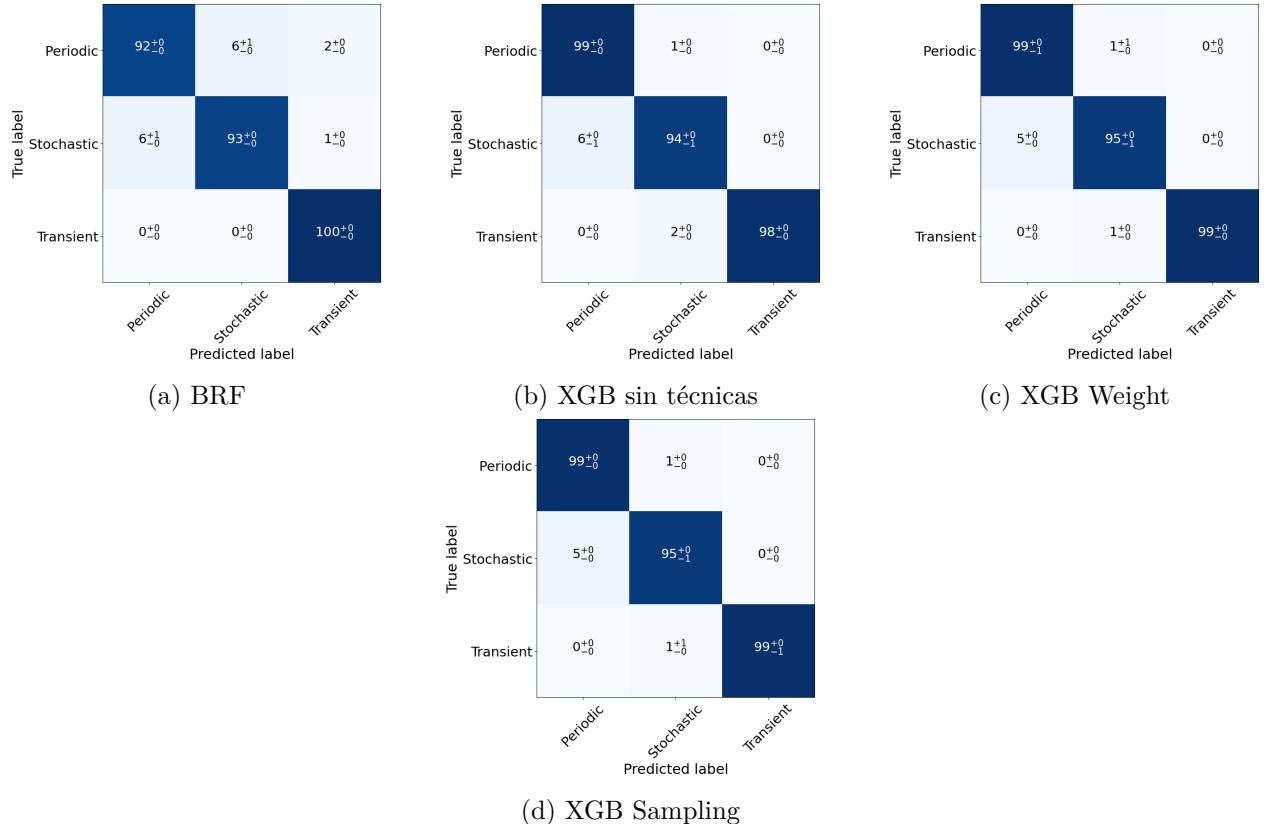


Figura 6.1: Matrices de confusión del nivel superior del clasificador de curvas de luz obtenidas del test con las curvas de luz nuevas de ALeRCE.

Los resultados obtenidos con XGBoost fueron bastante similares para el clasificador jerárquico y superiores a lo obtenido con BRF. Esta superioridad de resultados se puede observar además en las matrices de confusión mostradas en la figura 6.1, en donde en comparación, destaca la mayor cantidad de curvas de luz periódicas predichas como falsas curvas estocásticas.

Al igual que para el nivel superior del clasificador de curvas de luz, los resultados mostrados en la tabla 6.2 para el clasificador Periódico son muy inferiores a los obtenidos anteriormente en cada caso. De los valores obtenidos, se puede ver que el mejor resultado para Recall_M fue el de XGBoost al ser entrenado utilizando la técnica Weight (0.68), seguido por BRF (0.64), pero ambos resultados fueron menores a los obtenidos en los conjuntos de test anteriores (0.86 y 0.82 respectivamente).

Tabla 6.2: Desempeño en métricas de Precision, Recall, F1 score y min Recall obtenidas para el clasificador Periódico, usando las curvas de luz nuevas de ALeRCE.

Modelo	Técnica	Precision_M	Recall_M	F1 score_M	min Recall
BRF	None	0.51 ± 0.00	0.64 ± 0.00	0.43 ± 0.00	0.40 ± 0.01
XGB	Weight	0.53 ± 0.01	0.68 ± 0.01	0.54 ± 0.02	0.46 ± 0.04
XGB	Sampling	0.70 ± 0.00	0.59 ± 0.01	0.63 ± 0.00	0.23 ± 0.01
XGB	CNN+ROS	0.52 ± 0.01	0.62 ± 0.01	0.47 ± 0.01	0.35 ± 0.02

De las matrices de confusión mostradas en la figura 6.2 se puede observar que para BRF y XGBoost con las técnicas CNN y ROS se presenta una alta confusión en la predicción de curvas pertenecientes a la clase intermedia Periodic-Other, afectando principalmente a la correcta clasificación de las clases mayoritarias. El caso de XGBoost con Sampling presenta grandes confusiones de las curvas de las clases minoritarias como falsas E, provocando así el menor valor de Recall_M y mínimo Recall por clase.

Por otro lado, la matriz de confusión obtenida para XGBoost con Weight muestra que las predicciones erróneas fueron mayormente hacia la clase E y luego hacia la clase Periodic-Other, siendo las clases minoritarias las más afectadas.

Si bien los mejores resultados anteriormente fueron obtenidos con las técnicas Weight y Sampling, con matrices de confusión similares (figuras 4.7.a y 4.8.a respectivamente), en este último test los resultados de Sampling muestra fuerte sesgo hacia la clase mayoritaria, demostrando que Weight fue mejor para superar el sesgo hacia las clases mayoritarias en el entrenamiento del clasificador Periódico.

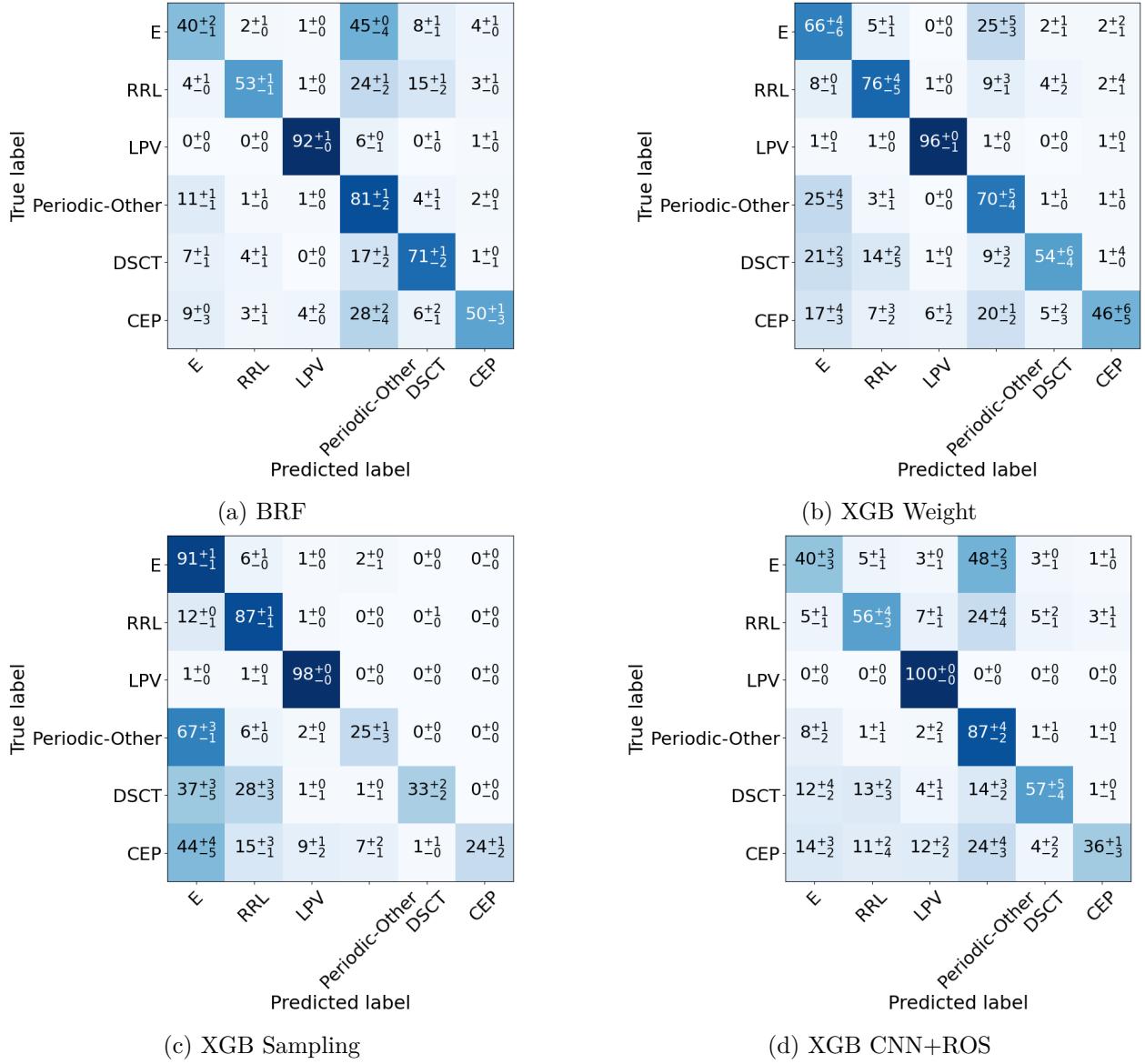


Figura 6.2: Matrices de confusión del clasificador Periódico obtenidas del test con las curvas de luz nuevas de ALeRCE.

En la tabla 6.3 se muestran los resultados obtenidos para el clasificador Estocástico para el test con nuevas curvas de luz. Los valores obtenidos son bastante similares entre sí en cuanto al Recall en su versión macro, pero en cuanto al mínimo por clase, es XGBoost con la técnica Weight la que obtuvo un mayor valor.

Tabla 6.3: Desempeño en métricas de Precision, Recall, F1 score y min Recall obtenidas para el clasificador Estocástico, usando las curvas de luz nuevas de ALeRCE.

Modelo	Técnica	Precision_M	Recall_M	F1 score_M	min Recall
BRF	None	0.64 ± 0.01	0.75 ± 0.00	0.69 ± 0.00	0.32 ± 0.01
XGB	Weight	0.66 ± 0.01	0.77 ± 0.00	0.70 ± 0.01	0.37 ± 0.02
XGB	Sampling	0.72 ± 0.01	0.75 ± 0.00	0.73 ± 0.00	0.25 ± 0.01
XGB	Weight, ALLKNN+ADASYN	0.66 ± 0.01	0.76 ± 0.00	0.70 ± 0.01	0.32 ± 0.02
XGB	Weight, ALLKNN+SMOTE	0.66 ± 0.01	0.76 ± 0.00	0.70 ± 0.01	0.32 ± 0.03
XGB	Weight, RUS+ADASYN	0.67 ± 0.01	0.77 ± 0.00	0.70 ± 0.01	0.31 ± 0.02
XGB	Weight, RUS+ROS	0.68 ± 0.01	0.75 ± 0.00	0.71 ± 0.01	0.27 ± 0.01
XGB	Weight, RUS+SMOTE	0.66 ± 0.01	0.76 ± 0.00	0.70 ± 0.01	0.31 ± 0.021

Si bien tanto para BRF como también para XGBoost con distintas técnicas de balance de datos se obtuvieron valores muy por debajo a lo obtenido anteriormente para el clasificador Estocástico, estos siguen siendo mejores a los obtenidos con el resto de los clasificadores del nivel inferior del clasificador de curvas de luz (Periódico y Transiente).

Analizando las matrices de confusión de la figura 6.3 obtenidas para el test final con nuevas curvas de luz, estas son bastante similares entre si en cuanto a la tasa de aciertos para cada una de las clases estocásticas. La mayores diferencia se presenta en la cantidad de falsas predicciones de las curvas de la clase Blazar como pertenecientes a las clases mayoritarias QSO y AGN, donde es XGBoost entrenado con la técnica Weight el caso en que se obtuvo una menor confusión entre clases.

Los resultados del test final para al clasificador Transiente usando BRF y XGBoost con la técnica Weight son los mostrados en la tabla 6.4. Los valores obtenidos con XGBoost fueron más altos que los de BRF en las métricas Precision_M y F1-score_M , pero más bajos en Recall_M y mínimo Recall por clase, indicando así un mayor sesgo hacia las clases mayoritarias por parte de XGBoost.

Tabla 6.4: Desempeño en métricas de Precision, Recall, F1 score y min Recall obtenidas para el clasificador Transiente, usando las curvas de luz nuevas de ALeRCE.

Modelo	Técnica	Precision_M	Recall_M	F1 score_M	min Recall
BRF	X	0.50 ± 0.01	0.62 ± 0.02	0.50 ± 0.02	0.47 ± 0.02
XGB	Weight	0.58 ± 0.03	0.60 ± 0.02	0.57 ± 0.02	0.37 ± 0.08

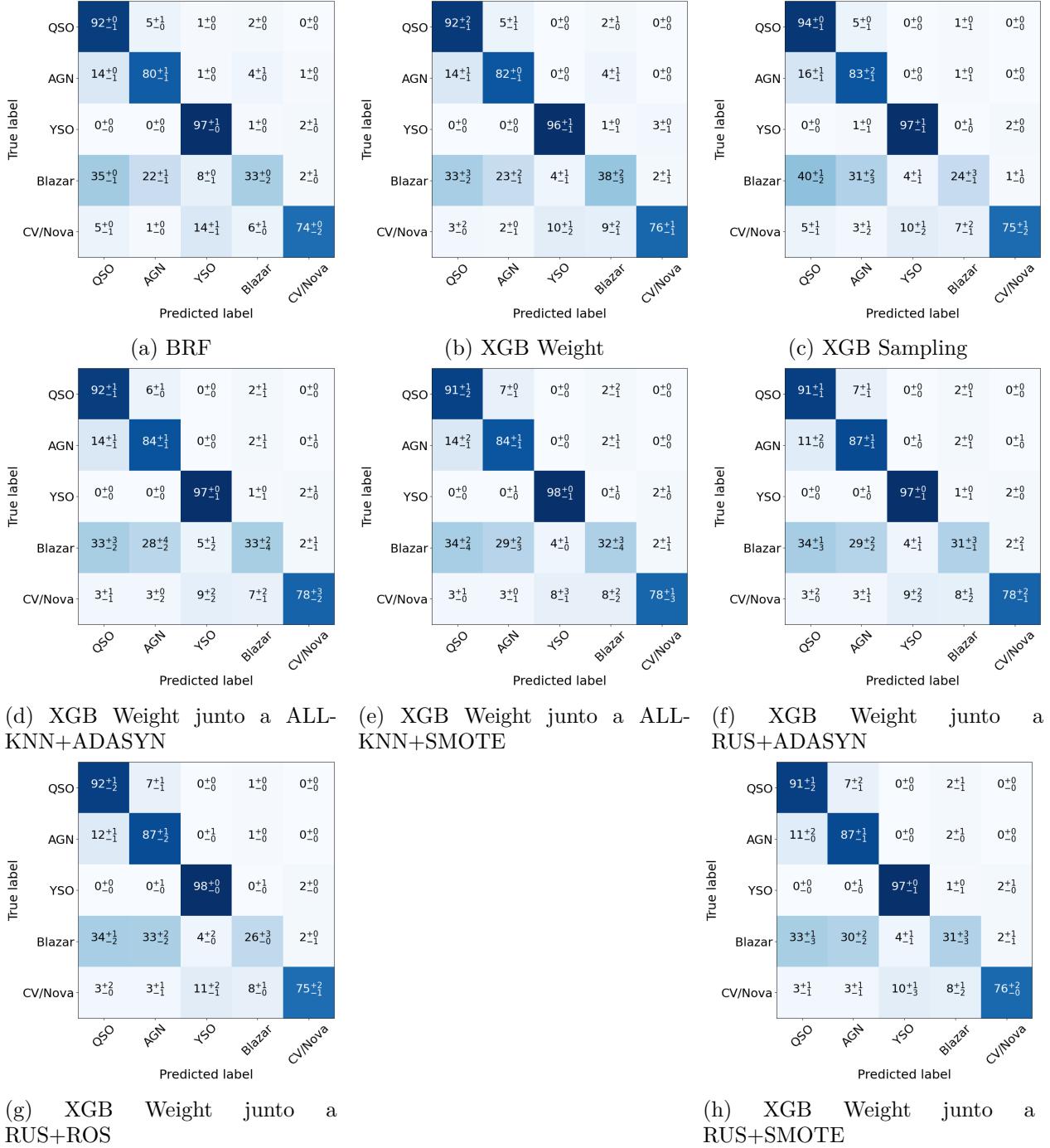
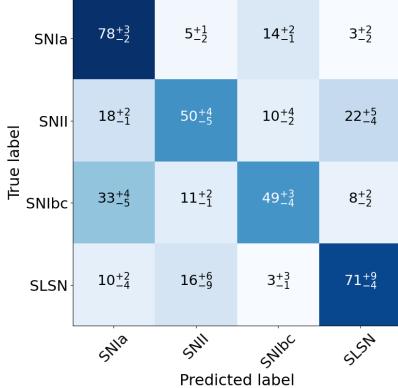
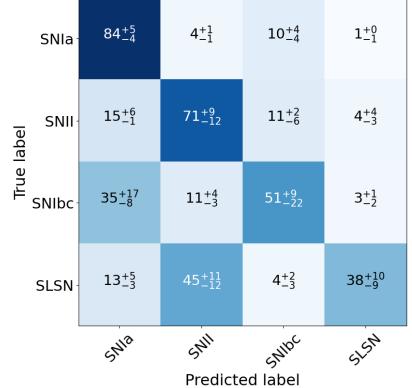


Figura 6.3: Matrices de confusión del clasificador Estocástico obtenidas del test con las curvas de luz nuevas de ALeRCE.

La matriz de confusión mostrada en la figura 6.4.a obtenida para BRF es bastante similar a la obtenida con los conjuntos de test anteriores en la figura 4.1.d para el clasificador Transiente, destacando solo la reducción de aciertos a la clase minoritaria SLSN. Pero para XGBoost la reducción de aciertos en las clases SNIbc y SLSN en la matriz de la figura 6.4.b fue mayor aun, resultando en un peor desempeño teniendo en cuenta el sesgo en las predicciones hacia las clases mayoritarias.



(a) BRF



(b) XGB Weight

Figura 6.4: Matrices de confusión del clasificador Transiente obtenidas del test con las curvas de luz nuevas de ALeRCE.

Finalmente, en la tabla 6.5 se muestran los resultados obtenidos en la unión de niveles del clasificador de curvas de luz para el test final con curvas de luz nuevas de ALeRCE. En casa una de las métricas a evaluar XGBoost supero a BRF, pero con diferencias casi mínimas en los valores de Recall macro y minimo por clase.

Tabla 6.5: Desempeño en métricas de Precision, Recall, F1 score y min Recall obtenidas para la unión de niveles del clasificador de curvas de luz nuevas de ALeRCE.

Modelo	Técnica	Precision _M	Recall _M	F1 score _M	min Recall
BRF	X	0.45±0.00	0.61±0.00	0.45±0.00	0.28±0.01
XGB	Weight	0.55±0.01	0.63±0.00	0.56±0.01	0.30±0.04

En la figura 6.5 se muestran las matrices de confusión obtenidas para BRF y XGBoost Weight para la unión de niveles del clasificador de curvas de luz en el test final. Se observa que con este test ha cambiado la situación obtenida con las curvas de luz anteriores, en la que BRF superó a XGBoost en la tasa de aciertos de solo las clases SLSN y Blazar. En esta ocasión BRF obtuvo mayor éxito principalmente al clasificar las clases minoritarias de los tres tipos de curvas de luz (periódicas, estocásticas y transientes).

En cuanto al rango, el mínimo porcentaje de aciertos obtenido con BRF fue del 29% para Blazar y el máximo de 92% para QSO, por otro lado, para XGBoost el mínimo fue de 34% y el máximo de 97% para las clases Blazar y LPV. Pero en general el desempeño fue muy similar entre los dos algoritmos tal como se muestra en la tabla 6.6 para los valores de promedio, mediana y desviación estándar de los porcentajes de aciertos entre las distintas clases a clasificar.

Tabla 6.6: Promedio, mediana y desviación estándar (std) del porcentaje de aciertos del test con nuevas curvas de ALeRCE para la unión de niveles del clasificador de curvas de luz.

Método	Métrica	Promedio	Mediana	Std
BRF	None	62 %	66 %	18 %
XGB	Weight	64 %	65 %	19 %

Por lo que a pesar de que los resultados de la tabla 6.5 indican un desempeño superior para XGBoost entrenado con la técnica Weight, este en general es bastante similar al obtenido con BRF.

En base a los resultados obtenidos con XGBoost y la técnica *Cost Sensitive Learning* (Weight) en este test final es que se confirma que esta corresponde a la mejor técnica de balance de datos para utilizar en el entrenamiento de XGBoost, pues en cada caso supero al resto de las técnicas en cuanto a sus valores de Recall_M , mínimo Recall por clase y tasa de aciertos en la predicción de las distintas clases.

	SNIa	SNI	SNIbc	SLSN	QSO	AGN	YSO	Blazar	CV/Nova	E	RRL	LPV	Periodic-Other	DSCT	CEP
True label	78^{+2}_{-3}	5^{+1}_{-2}	14^{+2}_{-1}	3^{+2}_{-2}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
SNIa	78^{+2}_{-3}	5^{+1}_{-2}	14^{+2}_{-1}	3^{+2}_{-2}	0^{+0}_{-0}	1^{+1}_{-0}	0^{+1}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
SNI	18^{+2}_{-1}	49^{+5}_{-5}	10^{+4}_{-2}	21^{+5}_{-3}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
SNIbc	33^{+4}_{-5}	11^{+2}_{-1}	49^{+3}_{-4}	8^{+2}_{-2}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
SLSN	9^{+3}_{-4}	14^{+7}_{-7}	2^{+2}_{-2}	66^{+8}_{-4}	3^{+1}_{-0}	5^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
QSO	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	92^{+0}_{-1}	5^{+1}_{-0}	1^{+0}_{-1}	2^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}
AGN	0^{+1}_{-0}	1^{+0}_{-1}	1^{+0}_{-1}	0^{+1}_{-0}	13^{+0}_{-0}	79^{+1}_{-0}	1^{+0}_{-0}	3^{+1}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	2^{+0}_{-1}	0^{+0}_{-0}	0^{+0}_{-0}
YSO	0^{+0}_{-0}	0^{+0}_{-0}	1^{+0}_{-1}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	69^{+1}_{-1}	0^{+0}_{-0}	1^{+0}_{-0}	2^{+0}_{-0}	0^{+0}_{-0}	3^{+1}_{-0}	20^{+1}_{-1}	1^{+0}_{-0}	3^{+0}_{-1}
Blazar	1^{+0}_{-0}	0^{+1}_{-0}	1^{+1}_{-0}	1^{+0}_{-0}	35^{+1}_{-1}	22^{+2}_{-0}	6^{+1}_{-0}	29^{+1}_{-1}	1^{+1}_{-0}	0^{+0}_{-0}	1^{+0}_{-1}	1^{+0}_{-0}	2^{+0}_{-1}	0^{+0}_{-0}	0^{+0}_{-0}
CV/Nova	8^{+2}_{-2}	1^{+0}_{-0}	2^{+1}_{-1}	0^{+1}_{-0}	5^{+0}_{-0}	0^{+1}_{-0}	6^{+1}_{-1}	5^{+1}_{-2}	48^{+1}_{-2}	3^{+2}_{-1}	3^{+0}_{-0}	3^{+1}_{-0}	7^{+2}_{-0}	6^{+1}_{-2}	3^{+0}_{-1}
E	1^{+0}_{-1}	0^{+0}_{-0}	1^{+0}_{-1}	0^{+0}_{-0}	2^{+1}_{-0}	0^{+0}_{-0}	5^{+0}_{-1}	1^{+1}_{-0}	1^{+0}_{-0}	38^{+3}_{-2}	2^{+1}_{-0}	0^{+1}_{-0}	37^{+0}_{-3}	8^{+1}_{-0}	4^{+0}_{-0}
RRL	2^{+0}_{-1}	0^{+0}_{-0}	0^{+1}_{-0}	0^{+0}_{-0}	4^{+0}_{-0}	0^{+1}_{-0}	5^{+1}_{-0}	1^{+0}_{-0}	2^{+1}_{-1}	3^{+0}_{-0}	52^{+1}_{-1}	1^{+0}_{-0}	14^{+1}_{-1}	13^{+1}_{-1}	3^{+1}_{-1}
LPV	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	10^{+1}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	87^{+1}_{-0}	2^{+0}_{-0}	0^{+0}_{-0}	1^{+0}_{-0}	0^{+0}_{-0}
Periodic-Other	0^{+1}_{-0}	0^{+0}_{-0}	0^{+1}_{-0}	0^{+0}_{-0}	3^{+0}_{-0}	0^{+1}_{-0}	8^{+0}_{-1}	2^{+1}_{-0}	0^{+1}_{-0}	10^{+1}_{-1}	1^{+1}_{-0}	1^{+0}_{-0}	69^{+2}_{-2}	3^{+1}_{-0}	2^{+0}_{-1}
DSCT	1^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	1^{+0}_{-0}	0^{+0}_{-0}	2^{+1}_{-1}	0^{+0}_{-0}	1^{+1}_{-0}	7^{+2}_{-1}	4^{+1}_{-1}	0^{+0}_{-0}	13^{+1}_{-1}	70^{+2}_{-1}	1^{+0}_{-1}
CEP	1^{+1}_{-1}	1^{+0}_{-1}	1^{+1}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	0^{+0}_{-0}	6^{+2}_{-2}	1^{+0}_{-1}	0^{+1}_{-0}	7^{+2}_{-1}	3^{+1}_{-1}	4^{+0}_{-2}	21^{+3}_{-3}	6^{+2}_{-1}	48^{+2}_{-2}

(a) BRF

(b) XGB Weight

Figura 6.5: Matrices de confusión para la unión de niveles del clasificador de curvas de luz obtenidas del test con las curvas de luz nuevas de ALeRCE.

Capítulo 7

Conclusiones

En el presente trabajo, se realizó una investigación sobre el problema del desbalance en los datos y el estado del arte en técnicas para su tratamiento se implementaron en conjunto con XGBoost 20 técnicas distintas con enfoque en el preprocesamiento de datos o en el algoritmo, y se probaron además combinaciones de estas que puedan ser complementarias.

Se exploró a fondo el potencial de XGBoost como algoritmo predictivo del clasificador de curvas de luz, el cual en [4] fue entrenado con sus hiperparámetros por defecto e incorporando técnicas de muestreo básicas como *Random Oversampling* y *Random Undersampling*, logrando resultados prometedores pero con claros sesgos hacia las clases mayoritarias a raíz del gran desbalance en los datos.

Mediante la selección de 8 hiperparámetros de XGBoost se obtuvo una considerable mejora de desempeño en comparación a lo obtenido por ALeRCE para este mismo modelo. Sin embargo la optimización de hiperparámetros demostró no ser suficiente para hacer frente al efecto del desbalance.

Al incorporar técnicas de balance a XGBoost se obtuvo una clara mejora en el desempeño del modelo en el escenario desbalanceado, aumentando en porcentajes significativos las tasas de aciertos de las clases minoritarias para cada uno de los clasificadores individuales del clasificador de curvas de luz.

Se probaron distintas métricas para evaluar el desempeño en la clasificación de múltiples clases para así seleccionar el mejor modelo predictivo. A partir del análisis de resultados se concluyó que Recall_M es la mejor métrica para evaluar el desempeño de un modelo entrenado en un escenario desbalanceado, en especial cuando son múltiples las clases a clasificar.

Por otro lado, Precision_M no es una métrica confiable en escenarios desbalanceados con múltiples clases. De todas formas, se comprobó que analizar las diferencias de Precision_M y Recall_M entre dos modelos resulta útil para identificar cambios en la magnitud del sesgo hacia las clases mayoritarias. Un aumento en Precision_M con una disminución de Recall_M es signo de que las predicciones realizadas hacia las clases mayoritarias han aumentado.

Al experimentar con técnicas de muestreo para preprocesar los conjuntos de entrenamiento utilizados con XGBoost, se obtuvieron mejores resultados al combinar técnicas de *oversam-*

pling y *undersampling*. Utilizar solo aumento de datos implica introducir demasiado ruido en los datos de entrenamiento en el caso de generación de instancias sintéticas, y por otro lado la reducción de datos implica la pérdida de información valiosa de las clases mayoritarias.

Si bien en las pruebas con combinaciones de técnicas de aumento y reducción de datos fueron varias las que demostraron un buen desempeño, fueron más las combinaciones descartadas por no cumplir con el mínimo Recall por clase establecido para considerar como aceptable un resultado.

Utilizar técnicas combinadas de aumento y reducción de datos junto con la técnica Weight provocó para varias combinaciones de técnicas de preprocessamiento una disminución de desempeño del modelo XGBoost. Esto debido a que para utilizar Weight se calculan los valores ponderadores de la función de pérdida en base a la distribución desbalanceada inicial de los datos. Por lo tanto al balancear esta distribución los ponderadores previamente calculados provocarían una sobre-distorsión en los valores de pérdida calculados durante el entrenamiento, provocando ajustes incorrectos en la estructura del modelo XGBoost.

Se logró implementar la función de pérdida *Focal Loss Cross-Entropy* para el entrenamiento de XGBoost para la clasificación de múltiples clases. Esto por medio de calcular el gradiente y el Hessiano de esta nueva función de pérdida y incorporando los resultados en su algoritmo.

Utilizar la función de pérdida *Focal Loss Cross-Entropy* para el entrenamiento de XGBoost resultó en desempeños inferiores a los obtenidos con la función de pérdida *Cross-Entropy*, ocurriendo lo contrario a lo que esperado, es decir un aumento del sesgo hacia las clases mayoritarias.

Varias técnicas de balance a nivel de datos en conjunto con XGBoost obtuvieron buenos resultados en uno o dos clasificadores del nivel inferior del clasificador de curvas de luz. Presentando entre si diferencias en el punto de operación del modelo entrenado, aumentando el acierto sobre ciertas clases en desmedro de la correcta clasificación de otras, lo cual podría ser analizado a futuro con el fin de medir el impacto de los puntos de operación desde una perspectiva astronómica.

La técnica que demostró un buen desempeño en cada clasificador y su unión, fue la de *Cost Sensitive Learning*, la cual además, por tener un alcance a nivel del algoritmo, resulta ser más eficiente en comparación con las técnicas de balance por muestreo.

Los resultados de XGBoost en conjunto con la técnica *Cost Sensitive Learning*, en la unión de niveles del clasificador de curvas de luz fueron de 0.67, 0.79 y 0.70 para Precision_M , Recall_M y F1-score_M respectivamente. Lo cual, en su contraparte, con el modelo BRF se obtuvieron valores de 0.57, 0.76 y 0.60 para las mismas métricas, respectivamente.

En el nivel superior del clasificador, XGBoost es capaz de identificar de manera casi perfecta las tres clases superiores, presentando mínimas confusiones en las predicciones realizadas. En cuanto a la unión de niveles del clasificador, el desempeño de XGBoost con *Cost Sensitive Learning* superó o igualó al de BRF en la cantidad de predicciones acertadas en 13 de las 15 clases en total. Solo en las clases SLSN y Blazar su desempeño fue un 9% y 2% menor en

proporción al total de instancias por clase.

Se logró comparar el algoritmo XGBoost con el algoritmo *Balanced Random Forest* por medio del test *t-student* y del *Permutation test*, comprobando estadísticamente que para la unión de niveles del clasificador de curvas de luz y para los clasificadores Periódico y Estocástico las diferencias en Precision_M , Recall_M y F1-score_M entre el desempeño de XGBoost y BRF son significativas. Si bien para el clasificador Transiente se obtuvieron mayores valores con XGBoost, solo en Recall_M no se pudo demostrar una diferencia significativa.

En el test con nuevas curvas de luz de ALeRCE se probó el algoritmo BRF y los modelos entrenados de XGBoost con las técnicas de balance con las que obtuvo un mejor desempeño anteriormente. En cada uno de los casos a comparar se obtuvieron desempeños por debajo a lo obtenido con los conjuntos de test utilizados anteriormente.

De este último test se pudo confirmar que la mejor técnica para implementar en conjunto con XGBoost para tratar el desbalance de datos es la de *Cost Sensitive Learning (Weight)* debido a que para cada clasificador obtuvo los mejores resultados tanto en el Recall macro y mínimo por clase, como también en la tasa de aciertos. Pero comparando con BRF, este obtuvo resultados inferiores a los de XGBoost con Weight en todos los clasificadores menos el Transiente, pero en general, para la unión de niveles del clasificador de curvas de luz, los resultados obtenidos fueron bastante similares.

Habiendo demostrado la superioridad del modelo XGBoost por sobre *Balanced Random Forest* en el clasificador jerárquico, Periódico y Estocástico, y con resultados similares para el clasificador Transiente, es que se recomienda su implementación en conjunto con la técnica de *Cost Sensitive Learning* para ser utilizado por el *broker* ALeRCE en su clasificador de curvas de luz.

Bibliografía

- [1] Chen, T. and Guestrin, C. “XGBoost: A Scalable Tree Boosting System,” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* pp. 785-794, 2016.
- [2] Förster, F. *et al.* “The Automatic Learning for the Rapid Classification of Events (ALeRCE) Alert Broker,” *The Astronomical Journal*, vol. 161, no. 5, 2021.
- [3] ALeRCE [Online]. Available: <http://alerce.science/services/>. [Accessed: 11-Jun-2021].
- [4] Sanchez-Saéz, P. *et al.* “Alert Classification for the ALeRCE Broker System: The Light Curve Classifier,” *The Astronomical Journal*, vol. 161, no. 3, 2021.
- [5] Catelan, M. and Smith, H. “Pulsating stars”, John Wiley & Sons, 2014.
- [6] Chen, C. and Liaw, A., Breiman, L., “Using Random Forest to Learn Imbalanced Data,” University of California, Berkeley, 2004.
- [7] XGBoost. [Online]. Available: <https://www.kaggle.com/code/dansbecker/xgboost/notebook>. [Accessed: 24-Mar-2022].
- [8] Błaszczyński, J., Stefanowski, J., Yen, G., “Local Data Characteristics in Learning Classifiers from Imbalanced Data,” *Advances in Data Analysis with Computational Intelligence Methods*, pp. 51-85. Springer, 2018.
- [9] Liu, L. and Miao, H. “Identification of Different Types of Minority Class Examples in Imbalanced Data,” in *Hybrid Artificial Intelligent Systems: 7th International Conference, HAIS 2012*, pp. 139-150. Springer, 2012.
- [10] Haixiang, G. *et al.*, “Learning from class-imbalanced data: Review of methods and applications,” *Expert Systems With Applications* 73, pp.220-239. ELSEVIER, 2017.
- [11] 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.
- [12] 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.
- [13] Nguyen, H., Cooper, E., Kamei, K. “Borderline over-sampling for imbalanced data classification,” *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1), pp.4-21, 2009.
- [14] Haibo, H., Bai, Y., Garcia, E., Li, S. “ADASYN: Adaptive synthetic sampling approach for imbalanced learning,” *IEEE International Joint Conference on Neural Net-*

works (IEEE World Congress on Computational Intelligence), pp. 1322-1328, 2008.

- [15] Abdi, L., and Hashemi, S. “To combat multi-class imbalanced problems by means of over-sampling techniques”. *IEEE Transactions on Knowledge and Data Engineering* 28 (January 2016), 238–251.
- [16] Zhang, J. and 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.
- [17] Hart, P. “The condensed nearest neighbor rule,” *Information Theory, IEEE Transactions on*, vol. 14(3), pp. 515-516, 1968.
- [18] Wilson, D. “Properties of Nearest Neighbor Rules Using Edited Data,” in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 2 (3), pp. 408-421, 1972.
- [19] Tomek, I. “An Experiment with the Edited Nearest-Neighbor Rule,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6(6), pp. 448-452, June 1976.
- [20] Laurikkala, J. “Improving identification of difficult small classes by balancing class distribution,” *Proceedings of the 8th Conference on AI in Medicine in Europe: Artificial Intelligence Medicine*, pp. 63-66, Springer Berlin Heidelberg, 2001.
- [21] Smith, D., Michael, R., Martinez, T., Giraud-Carrier, C., “An instance level analysis of data complexity.” *Machine learning* 95.2 (2014): 225-256.
- [22] Wojciechowski, S., Wilk, S., and Stefanowski, J. “An algorithm for selective preprocessing of multi-class imbalanced data”. *Proceedings of the 10th International Conference on Computer Recognition Systems* (2017), pp. 238–247.
- [23] Lango, M., and Stefanowski, J. “SOUP-Bagging: a new approach for multi-class imbalanced data classification”. PP-RAI '19: Polskie Porozumienie na Rzecz Sztucznej Inteligencji (2019).
- [24] Lin, T.-Y., et al. “Focal loss for dense object detection,” *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.
- [25] Wang, C., Deng, C., Wang, S., “Imbalance-XGBoost: Leveraging Weighted and Focal Losses for Binary Label-Imbalanced Classification with XGBoost,” *Pattern Recognition Letters*, vol 136, pp. 190-197, 2020.
- [26] Fernández, A., García, S., Galar, M., Prati, R.C., Krawczyk, B., Herrera, F. “Data Intrinsic Characteristics. Learning from Imbalanced Data Sets,” pp. 253–277. Springer, Cham (2018).
- [27] Mortaz, E. “Imbalance accuracy metric for model selection in multi-class imbalance classification problems,” ELSEVIER, 2020.
- [28] “sklearn.model_selection.StratifiedShuffleSplit,” scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html. [Accessed: 10-Oct-2021].
- [29] JavierMolinaF (2022) Clasificador-de-curvas-de-luz-utilizando-modelo-XGBoost-y-tecnicas-de-balance-de-datos [Source code]. <https://github.com/JavierM23/Clasificador-de-curvas-de-luz-utilizando-modelo-XGBoost-y-tecnicas-de-balance-de-datos>

- [30] Bergstra, J., Yamins, D., Cox, D. D. (2013) “Making a Science of Model Search: Hyper-parameter Optimization in Hundreds of Dimensions for Vision Architectures”, *Proc. of the 30th International Conference on Machine Learning*, ICML, 2013.
- [31] Kubat, M., Matwin, S. “Addressing the curse of imbalanced training sets: one-sided selection”, *ICML*, vol. 97, pp. 179-186, 1997.
- [32] Tomek, I. “Two modifications of CNN,” in *Systems, Man, and Cybernetics, IEEE Transactions on*, vol. 6, pp 769-772, 1976.

Capítulo 8

Anexos

8.1. Matrices de confusión

8.1.1. Resultados obtenidos por ALeRCE en [4]

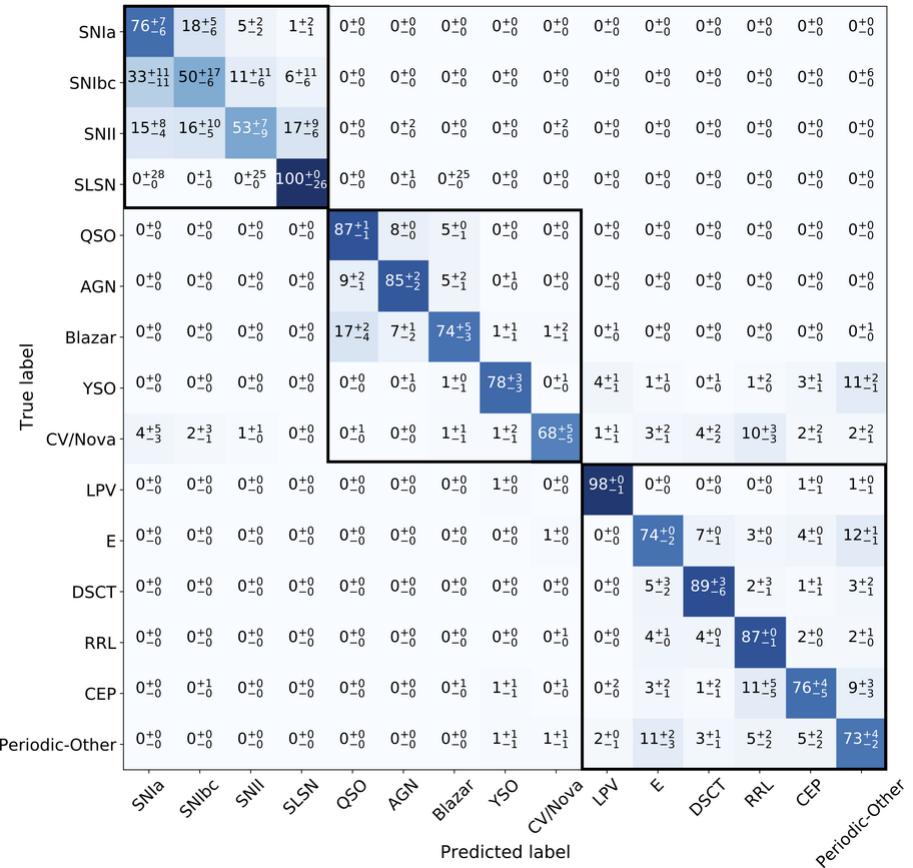


Figura 8.1: Matriz de confusión obtenida con BRF para la unión de capas del clasificador de curvas de luz.

	SNIa	SNIbc	SNII	SLSN	QSO	AGN	Blazar	YSO	CV/Nova	LPV	E	DSCT	RRL	CEP	Periodic-Other
True label	76 ⁺⁷ ₋₆	18 ⁺⁵ ₋₆	5 ⁺² ₋₂	1 ⁺² ₋₁	0 ⁺⁰ ₋₀										
SNIa	76 ⁺⁷ ₋₆	18 ⁺⁵ ₋₆	5 ⁺² ₋₂	1 ⁺² ₋₁	0 ⁺⁰ ₋₀										
SNIbc	33 ⁺¹¹ ₋₁₁	50 ⁺¹⁷ ₋₆	11 ⁺¹¹ ₋₆	6 ⁺¹¹ ₋₆	0 ⁺⁰ ₋₀	0 ⁺⁶ ₋₀									
SNII	15 ⁺⁸ ₋₄	16 ⁺¹⁰ ₋₅	53 ⁺⁷ ₋₉	17 ⁺⁹ ₋₆	0 ⁺⁰ ₋₀	0 ⁺² ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺² ₋₀	0 ⁺⁰ ₋₀					
SLSN	0 ⁺²⁸ ₋₀	0 ⁺¹ ₋₀	0 ⁺²⁵ ₋₀	100 ⁺⁰ ₋₂₆	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺²⁵ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀
QSO	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	87 ⁺¹ ₋₁	8 ⁺⁰ ₋₀	5 ⁺⁰ ₋₁	0 ⁺⁰ ₋₀							
AGN	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	9 ⁺² ₋₁	85 ⁺² ₋₂	5 ⁺² ₋₁	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀						
Blazar	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	17 ⁺⁴ ₋₄	7 ⁺¹ ₂	74 ⁺⁵ ₃	1 ⁺¹ ₁	1 ⁺² ₁	0 ⁺¹ ₀	0 ⁺⁰ ₀	0 ⁺⁰ ₀	0 ⁺⁰ ₀	0 ⁺⁰ ₀	0 ⁺¹ ₀
YSO	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₀	1 ⁺⁰ ₋₁	78 ⁺³ ₋₃	0 ⁺¹ ₀	4 ⁺¹ ₁	1 ⁺¹ ₀	0 ⁺¹ ₀	1 ⁺² ₀	3 ⁺¹ ₁	11 ⁺² ₋₁
CV/Nova	4 ⁺⁵ ₋₃	2 ⁺³ ₋₁	1 ⁺¹ ₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₀	0 ⁺⁰ ₋₀	1 ⁺¹ ₋₁	1 ⁺² ₁	68 ⁺⁵ ₋₅	1 ⁺¹ ₁	3 ⁺² ₁	4 ⁺² ₂	10 ⁺³ ₋₃	2 ⁺² ₁	2 ⁺² ₁
LPV	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	98 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₁	1 ⁺⁰ ₋₁
E	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	1 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	74 ⁺⁰ ₋₂	7 ⁺⁰ ₋₁	3 ⁺⁰ ₋₀	4 ⁺⁰ ₋₁	12 ⁺¹ ₋₁
DSCT	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	5 ⁺³ ₋₂	89 ⁺³ ₋₆	2 ⁺³ ₋₁	1 ⁺¹ ₋₁	3 ⁺² ₋₁
RRL	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₀	0 ⁺⁰ ₋₀	4 ⁺¹ ₀	4 ⁺⁰ ₋₁	87 ⁺⁰ ₋₁	2 ⁺⁰ ₋₀	2 ⁺¹ ₋₀
CEP	0 ⁺⁰ ₋₀	0 ⁺¹ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺¹ ₀	0 ⁺¹ ₀	0 ⁺⁰ ₋₀	0 ⁺² ₀	3 ⁺² ₁	1 ⁺² ₁	11 ⁺⁵ ₋₅	76 ⁺⁴ ₋₅	9 ⁺³ ₋₃
Periodic-Other	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	0 ⁺⁰ ₋₀	1 ⁺¹ ₋₁	1 ⁺¹ ₋₁	2 ⁺⁰ ₁	11 ⁺³ ₋₃	3 ⁺¹ ₁	5 ⁺² ₋₂	5 ⁺² ₂	73 ⁺⁴ ₋₂

Figura 8.2: Matriz de confusión obtenida con XGBoost para la unión de capas del clasificador de curvas de luz.

8.1.2. Glosario hiperparámetros

8.1.2.1. XGBoost

- *learning_rate*: Tasa de aprendizaje, A menor es su valor más conservativo se vuelve el proceso de aprendizaje, actualizando así en menor proporción el algoritmo, previniendo el sobreajuste.
- *max_depth*: Máxima profundidad permitida en los arboles de decisión. Entre mayor la profundidad, mayor es la complejidad del modelo resultante.
- *min_child_weight*: Mínima suma de pesos de instancias por hoja en el proceso de construcción de los arboles de decisión. Si son muy pocas las instancias que caen en una respectiva hoja, no se continua con la partición de esta misma.
- *gamma*: Mínimo valor de reducción en la pérdida requerido para continuar particionando cierta hoja de un árbol en construcción. A mayor *gamma* más conservador se vuelve el algoritmo.
- *reg_alpha*: Hiperparámetro regulador para el cálculo de los pesos en las hojas.
- *reg_lambda*: Hiperparámetro regulador para el cálculo de los pesos en las hojas.

- *colsample_bytree*: Proporción de columnas a utilizar para construir cada árbol de decisión.
- *subsample*: Proporción de instancias de entrenamiento a utilizar en el entrenamiento de cada árbol de decisión.

8.1.2.2. Técnicas de *Oversampling*

- *strategy*: Cantidad objetivo de instancias por clase a alcanzar en el proceso de aumento de datos.
- *k_neighbors*: Número de vecinos más cercanos a considerar para generar nuevas instancias sintéticas.
- *m_neighbors*: Número de vecinos más cercanos a usar para determinar si una instancia minoritaria es considerada como no segura.
- *kind*: Tipo de algoritmo de SMOTE a utilizar (*borderline-1*, *borderline-2*)

8.1.2.3. Técnicas de *Undersampling*

- *strategy*: Cantidad objetivo de instancias por clase a alcanzar en el proceso de aumento de datos.
- *n_neighbors*: Número de vecinos más cercanos a considerar en el procedimiento de KNN
- *kind_sel*: Estrategia a utilizar para remover instancias no seguras.
- *version*: Versión de *Near Miss* a realizar.

8.2. Cálculo de gradientes y Hessiano

8.2.1. Gradiente de la función *softmax*

$$s_i = \frac{e^{z_i}}{\sum_{k=1}^C e^{z_k}}$$

Como los valores de *softmax* son estrictamente positivos, se puede aplicar la técnica de derivada logarítmica:

$$\begin{aligned}
\frac{\partial}{\partial z_j} \log(s_i) &= \frac{1}{s_i} \frac{\partial s_i}{\partial z_j} \\
\Leftrightarrow \frac{\partial s_i}{\partial z_j} &= s_i \frac{\partial}{\partial z_j} \log(s_i) \\
\frac{\partial s_i}{\partial z_j} &= s_i \frac{\partial}{\partial z_j} \log\left(\frac{e^{z_i}}{\sum_{k=1}^C e^{z_k}}\right) \\
\frac{\partial s_i}{\partial z_j} &= s_i \frac{\partial}{\partial z_j} \left(z_i - \log\left(\sum_{k=1}^C e^{z_k}\right) \right) \\
\frac{\partial s_i}{\partial z_j} &= s_i \left(\frac{\partial z_i}{\partial z_j} - \frac{\partial}{\partial z_j} \log\left(\sum_{k=1}^C e^{z_k}\right) \right)
\end{aligned}$$

De donde:

$$\frac{\partial z_i}{\partial z_j} = \mathbb{1}_{i=j} := \begin{cases} 1, & \text{si } i = j \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{aligned}
\frac{\partial s_i}{\partial z_j} &= s_i \left(\mathbb{1}_{i=j} - \frac{e^{z_j}}{\sum_{k=1}^C e^{z_k}} \right) \\
\frac{\partial s_i}{\partial z_j} &= s_i \cdot (\mathbb{1}_{i=j} - s_j)
\end{aligned}$$

8.2.2. Gradiente de la función Focal Loss Cross-Entropy

$$\begin{aligned}
L(y, s) &= - \sum_{i=1}^C y_i (1 - s_i)^\gamma \log(s_i) \\
\frac{\partial L}{\partial z_j} &= - \frac{\partial}{\partial z_j} \sum_{i=1}^C y_i (1 - s_i)^\gamma \log(s_i) \\
&= - \sum_{i=1}^C y_i \left[(1 - s_i)^\gamma \frac{\partial}{\partial z_j} \log(s_i) + \log(s_i) \frac{\partial}{\partial z_j} (1 - s_i)^\gamma \right] \\
&= - \sum_{i=1}^C y_i \left[(1 - s_i)^\gamma \frac{1}{s_i} \frac{\partial s_i}{\partial z_j} - \gamma \log(s_i) (1 - s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_j} \right] \\
&= - \sum_{i=1}^C y_i \left[(1 - s_i)^\gamma (\mathbb{1}_{i=j} - s_j) - \gamma \log(s_i) (1 - s_i)^{\gamma-1} s_i (\mathbb{1}_{i=j} - s_j) \right] \\
&= \underbrace{- \sum_{i=1}^C y_i (1 - s_i)^\gamma (\mathbb{1}_{i=j} - s_j)}_{(1)} + \underbrace{\gamma \sum_{i=1}^C y_i \log(s_i) (1 - s_i)^{\gamma-1} s_i (\mathbb{1}_{i=j} - s_j)}_{(2)}
\end{aligned}$$

$$\begin{aligned}
(1) &\rightarrow - \sum_{i=1}^C y_i (1-s_i)^\gamma (\mathbb{1}_{i=j} - s_j) \\
&= - \sum_{i=1}^C y_i (1-s_i)^\gamma \mathbb{1}_{i=j} + s_j \sum_{i=1}^C y_i (1-s_i)^\gamma \\
&= - y_j (1-s_j)^\gamma + s_j \sum_{i=1}^C y_i (1-s_i)^\gamma \\
(2) &\rightarrow \sum_{i=1}^C y_i \log(s_i) (1-s_i)^{\gamma-1} s_i (\mathbb{1}_{i=j} - s_j) \\
&= \sum_{i=1}^C y_i \log(s_i) (1-s_i)^{\gamma-1} s_i \mathbb{1}_{i=j} - s_j \sum_{i=1}^C y_i \log(s_i) (1-s_i)^{\gamma-1} s_i \\
&= y_j \log(s_j) (1-s_j)^{\gamma-1} s_j - s_j \sum_{i=1}^C y_i \log(s_i) (1-s_i)^{\gamma-1} s_i \\
\frac{\partial L}{\partial z_j} &= - y_j (1-s_j)^\gamma + s_j \sum_{i=1}^C y_i (1-s_i)^\gamma + \gamma y_j \log(s_j) (1-s_j)^{\gamma-1} s_j - \gamma s_j \sum_{i=1}^C y_i \log(s_i) (1-s_i)^{\gamma-1} s_i \\
&= y_j \left[\gamma \log(s_j) (1-s_j)^{\gamma-1} s_j - (1-s_j)^\gamma \right] + s_j \sum_{i=1}^C y_i \left[(1-s_i)^\gamma - \gamma \log(s_i) (1-s_i)^{\gamma-1} s_i \right]
\end{aligned}$$

8.2.3. Hessiano de la función Focal Loss Cross-Entropy

$$\begin{aligned}
\frac{\partial L}{\partial z_j} &= y_j \left[\gamma \log(s_j) (1 - s_j)^{\gamma-1} s_j - (1 - s_j)^\gamma \right] + s_j \sum_{i=1}^C y_i \left[(1 - s_i)^\gamma - \gamma \log(s_i) (1 - s_i)^{\gamma-1} s_i \right] \\
\frac{\partial^2 L}{\partial z_j^2} &= \frac{\partial}{\partial z_j} \left(y_j \left[\gamma \log(s_j) (1 - s_j)^{\gamma-1} s_j - (1 - s_j)^\gamma \right] + s_j \sum_{i=1}^C y_i \left[(1 - s_i)^\gamma - \gamma \log(s_i) (1 - s_i)^{\gamma-1} s_i \right] \right) \\
&= y_j \underbrace{\frac{\partial}{\partial z_j} \left[\gamma \log(s_j) (1 - s_j)^{\gamma-1} s_j - (1 - s_j)^\gamma \right]}_{(1)} + \underbrace{s_j \sum_{i=1}^C y_i \left[(1 - s_i)^\gamma - \gamma \log(s_i) (1 - s_i)^{\gamma-1} s_i \right]}_{(2)} \\
(1) \rightarrow &\gamma \left(\log(s_j) (1 - s_j)^{\gamma-1} \frac{\partial s_j}{\partial z_j} + \log(s_j) s_j \frac{\partial}{\partial z_j} (1 - s_j)^{\gamma-1} + (1 - s_j)^{\gamma-1} s_j \frac{\partial}{\partial z_j} \log(s_j) + (1 - s_j)^{\gamma-1} \frac{\partial s_j}{\partial z_j} \right) \\
&= \gamma \left(\log(s_j) (1 - s_j)^{\gamma-1} \frac{\partial s_j}{\partial z_j} - (\gamma - 1) \log(s_j) (1 - s_j)^{\gamma-2} s_j \frac{\partial s_j}{\partial z_j} + (1 - s_j)^{\gamma-1} \frac{\partial s_j}{\partial z_j} + (1 - s_j)^{\gamma-1} \frac{\partial s_j}{\partial z_j} \right) \\
&= \gamma \left(\log(s_j) (1 - s_j)^{\gamma-1} \frac{\partial s_j}{\partial z_j} - (\gamma - 1) \log(s_j) (1 - s_j)^{\gamma-2} s_j \frac{\partial s_j}{\partial z_j} + 2(1 - s_j)^{\gamma-1} \frac{\partial s_j}{\partial z_j} \right) \\
&= \gamma (1 - s_j)^{\gamma-1} \frac{\partial s_j}{\partial z_j} \left[\log(s_j) (1 - (\gamma - 1)(1 - s_j)^{-1} s_j) + 2 \right] \quad \left(\frac{\partial s_j}{\partial z_j} = s_j \cdot (\mathbb{1}_{j=j} - s_j) = s_j \cdot (1 - s_j) \right) \\
&= \gamma (1 - s_j)^{\gamma-1} s_j (1 - s_j) \left[\log(s_j) (1 - (\gamma - 1)(1 - s_j)^{-1} s_j) + 2 \right] \\
&= \gamma (1 - s_j)^\gamma s_j \left[\log(s_j) (1 - (\gamma - 1)(1 - s_j)^{-1} s_j) + 2 \right] \\
(2) \rightarrow &\frac{\partial s_j}{\partial z_j} \sum_{i=1}^C y_i \left[(1 - s_i)^\gamma - \gamma \log(s_i) (1 - s_i)^{\gamma-1} s_i \right] \\
&+ s_j \sum_{i=1}^C y_i \left[-\gamma (1 - s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_j} - \gamma \frac{\partial}{\partial z_j} (\log(s_i) (1 - s_i)^{\gamma-1} s_i) \right] \\
&= s_j (1 - s_j) \sum_{i=1}^C y_i \left[(1 - s_i)^\gamma - \gamma \log(s_i) (1 - s_i)^{\gamma-1} s_i \right] \\
&- s_j \gamma \underbrace{\sum_{i=1}^C y_i \left[(1 - s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_j} + \frac{\partial}{\partial z_j} (\log(s_i) (1 - s_i)^{\gamma-1} s_i) \right]}_{(3)}
\end{aligned}$$

$$\begin{aligned}
(3) &\rightarrow \sum_{i=1}^C y_i \left[(1-s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_j} + \log(s_i)(1-s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_i} + \log(s_i)s_i \frac{\partial}{\partial z_j}(1-s_i)^{\gamma-1} + (1-s_i)^{\gamma-1}s_i \frac{\partial}{\partial z_j} \log(s_i) \right] \\
&= \sum_{i=1}^C y_i \left[(1-s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_j} + \log(s_i)(1-s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_i} - (\gamma-1)\log(s_i)(1-s_i)^{\gamma-2}s_i \frac{\partial s_i}{\partial z_j} + (1-s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_j} \right] \\
&= \sum_{i=1}^C y_i \left[\log(s_i)(1-s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_i} - (\gamma-1)\log(s_i)(1-s_i)^{\gamma-2}s_i \frac{\partial s_i}{\partial z_j} + 2(1-s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_j} \right] \\
&= \sum_{i=1}^C y_i (1-s_i)^{\gamma-1} \frac{\partial s_i}{\partial z_i} \left[\log(s_i)(1-(\gamma-1)(1-s_i)^{-1}s_i) + 2 \right] \\
&= \sum_{i=1}^C y_i (1-s_i)^{\gamma-1} s_i (\mathbb{1}_{i=j} - s_j) \left[\log(s_i)(1-(\gamma-1)(1-s_i)^{-1}s_i) + 2 \right] \\
&= y_j (1-s_j)^{\gamma-1} s_j \left[\log(s_j)(1-(\gamma-1)(1-s_j)^{-1}s_j) + 2 \right] \\
&\quad - s_j \sum_{i=1}^C y_i (1-s_i)^{\gamma-1} s_i \left[\log(s_i)(1-(\gamma-1)(1-s_i)^{-1}s_i) + 2 \right]
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 L}{\partial z_j^2} &= y_j \gamma (1-s_j)^\gamma s_j \left[\log(s_j)(1-(\gamma-1)(1-s_j)^{-1}s_j) + 2 \right] \\
&\quad + s_j (1-s_j) \sum_{i=1}^C y_i \left[(1-s_i)^\gamma - \gamma \log(s_i)(1-s_i)^{\gamma-1}s_i \right] \\
&\quad - y_j \gamma s_j (1-s_j)^{\gamma-1} s_j \left[\log(s_j)(1-(\gamma-1)(1-s_j)^{-1}s_j) + 2 \right] \\
&\quad + s_j^2 \gamma \sum_{i=1}^C y_i (1-s_i)^{\gamma-1} s_i \left[\log(s_i)(1-(\gamma-1)(1-s_i)^{-1}s_i) + 2 \right] \\
\frac{\partial^2 L}{\partial z_j^2} &= y_j \gamma (1-s_j)^\gamma s_j \left[\log(s_j)(1-(\gamma-1)(1-s_j)^{-1}s_j) + 2 \right] (1-s_j(1-s_j)^{-1}) \\
&\quad + s_j (1-s_j) \sum_{i=1}^C y_i \left[(1-s_i)^\gamma - \gamma \log(s_i)(1-s_i)^{\gamma-1}s_i \right] \\
&\quad + s_j^2 \gamma \sum_{i=1}^C y_i (1-s_i)^{\gamma-1} s_i \left[\log(s_i)(1-(\gamma-1)(1-s_i)^{-1}s_i) + 2 \right]
\end{aligned}$$