Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Predicción de cristalización de perovskitas mediante aprendizaje automático

Tesis de Licenciatura en Ciencias de la Computación

María Belén Ticona Oquendo

Director: Diego Onna
Codirector: Pablo Turjanski

Buenos Aires, Argentina
Julio 2023

PREDICCIÓN DE CRISTALIZACIÓN DE PEROVSKITAS MEDIANTE APRENDIZAJE AUTOMÁTICO

La ciencia de los materiales es un área de estudio interdisciplinar en el que tradicionalmente se descubren materiales mediante la prueba y error experimental en el laboratorio. Descubrir un nuevo material suele llevar varios años de investigación, así como también una considerable cantidad de recursos e inversiones. Actualmente, uno de los materiales más investigados son las *perovskitas* por ser una promesa en el desarrollo de paneles solares. Si bien existen múltiples maneras de sintetizarlas, se destaca en particular la técnica de cristalización debido a que permite obtener una caracterización detallada de su composición y estructura química, entre otros motivos.

Sin embargo, debido a la sensibilidad de este proceso químico, es difícil conocer en qué condiciones experimentales se produce la cristalización. De allí que el uso del tradicional mecanismo de prueba y error experimental suela generar datos en los que mayoritariamente no cristaliza la *perovskita*. En este contexto, existen trabajos que han demostrado que es posible desarrollar modelos de predicción de cristalización de *perovskitas* empleando técnicas de aprendizaje automático, usando datos ya recolectados sobre experimentaciones de síntesis de cristales. No obstante, desde un punto de vista metodológico, estos modelos se han realizado sin considerar las características que conlleva usar datos experimentales, como por ejemplo, la cantidad acotada de datos, el desbalance entre experimentaciones en donde efectivamente cristaliza, los sesgos en la recolección de los datos, entre otros.

En este trabajo realizamos un estudio de la sensibilidad de la evaluación de modelos predictivos frente a errores en la recolección y muestreo de datos experimentales. Para ello estudiamos cómo se ven afectadas las tradicionales métricas de evaluación -mediante matrices de confusión sintéticas- al usar conjunto de datos chicos y desbalanceados. Consideramos una evaluación basada en una representación bidimensional de métricas y definimos categorías de *performance* esperadas de los sistemas predictivos finales para poder valorar mejor su utilidad. Por último, aplicamos este mecanismo para estudiar en particular modelos de ensambles, pudiendo analizar qué estimadores generan un ensamble superador a los modelos individuales de un único estimador. Además, al estudiar ensambles heterogéneos concluimos que combinar modelos especializados por clase permite generar ensambles superadores con mejor desempeño que otros tipos de ensambles, de modo que la diversidad de modelos combinados mejora el ensamble final.

Palabras claves: perovskita, cristalización, aprendizaje automátco, desbalance, ensambles.

PREDICTION OF PEROVSKITE CRYSTALLIZATION THROUGH MACHINE LEARNING

Material science is an interdisciplinary field that generally discovers new materials through trial and error in the lab. Discovering new materials usually takes several years of research, as well as a considerable amount of resources and investments. Currently, one of the most relevant materials are perovskites, as they hold great promise for the development of solar cells. The crystallization mechanism is the most relevant method to synthesize perovskites as it allows a detailed (compositional and chemical) characterization of the material structure, among the different methods.

However, because of the sensitivity of the crystallization process, experimental conditions that produce pervoskite crystals are hardly discovered. Hence,the experimental trial and error process usually results in imbalanced data (where most experiments do not produce crystals).

In this context, studies have shown that it is possible to develop classification models that predict perovskite crystallization using data collected in laboratory experiments. However, from a methodological point of view, these models have been developed without considering the characteristics of using experimental data (such as the limited data available, class imbalances, biases in data collection, among others).

In this work, we studied model evaluation sensitivity to errors in data collection and experimental data sampling. Therefore, we analyzed how traditional classification metrics behave, using synthetic confusion matrices, when small and imbalance datasets are used. We proposed an alternative evaluation method, based on a bidimensional representation of two complementary metrics, which makes it easier to distinguish model performance. Finally, we applied this method to study ensamble models for small datasets, concluding that the combination of specialized models in different classes produces high-performance models.

Keywords: perovskite, crystallization, machine learning, imbalance, ensamble.

AGRADECIMIENTOS

Esta tesis culmina el largo camino de mi vida estudiantil en la UBA. Quiero agradecer en primer lugar a mi mamá y papá por todos sus esfuerzos para permitirme explorar cuanto quise durante mis años en la escuela y la universidad. Especialmente a mamá, por pensar siempre en cómo brindarme una mejor educación con lo que tenía a su alcance y por presentarme lo que serían mis puertas a la UBA. A mi familia.

A mis docentes de la secundaria que canalizaron mi motivación por explorar la ciencia, por incentivar mi curiosidad y permitirme que me cuestione qué y cómo quiero aprender en mi futuro. A Manu y a Yoshi.

A toda la gente que conocí durante mi estadía, llena de interrogantes, en química en Exactas. A Javi y Damián. Las cosas pasan por algo y todo eso tenía que pasar para encontrar mi lugar en Exactas. A Her y Pau por escucharme.

A toda la gente de los Profesorados de Exactas por hacerme entender mejor mi experiencia educativa personal y darme un norte cuando más lo necesitaba. A Mica y Sandra, a mis compas de cursada.

A aquellos espacios donde recibí una educación informal invaluable. A Puerta18, sus coordis y todes les niñes que me regalaron sus sonrisas, dudas e interrogantes. A todo el equipo TEDx por embarcarse en esa loca idea que hicimos realidad. A la UNA por permitirme explorar otros mundos artísticos, por ayudarme a encontrarme en mi búsqueda personal.

Al DC por ser mi lugar en Exactas.

A la comunidad de divulgación del DC por dejarme hacer lo que tanto disfruto durante toda la carrera. A Christian, Perla, Vir, Pablo, Augusto, Dani, Guilla, a todes les Divus y colaboradores.

A toda la comunidad docente del DC por hacerme crecer en esta hermosa tarea de enseñar. A David, Esteban, Chapa y todes mis colegas docentes.

A todas las Mujeres del DC, disidencias y aliades por su inspiración, acompañamiento y aliento. No estaría terminando esta carrera si no fuera por ustedes. A Vir, Lety, Dani y Caro por ser pilar fundacional para las que vinieron y vendrán.

A mis compañeres codepers y toda la comunidad de la ComCom. A Tobi, Mar, Gian, Rozen y Ale, por tantas risas, quejas y por compartir esta convicción de querer mejorar el DC desde el lado estudiantil.

A toda la comunidad mexicana en la UNAM por brindarme la posibilidad de vivir y encontrarme en un mundo paralelo. A toda la comunidad de Proyecto Nutria por enseñarme a confiar en mí y soñar en grande. A mi familia mexicana por adoptarme como una hija más, a Eduardo y Álica.

A todas las personas que conocí en *Seattle* por hacer que tenga una experiencia inolvidable. A Kristina y Brian, a Jela y Lalo. A Lupe, Rafa y José por ser una inspiración para la comunidad latina.

A todas las personas que no pude englobar en estos agradecimientos. A mis compañeres de cursada, a mis amigues de la secundaria y a quienes me han acompañado estos años. Gracias por permitirme ser parte de sus vidas y por hacer de la mía una alegría.

Por último, a Diego y Pablo por dirigir esta tesis, por toda su paciencia y predisposición. A Laura y Sara por su guía. A quienes no me conocen pero me han guiado.

A todas las personas que hacen de la educación pública una realidad.

Índice general

GLOSARIO

catión ión que posee una carga positiva y procede de un elemento electropositivo.

coordinación número de vecinos que están en contacto directo con un átomo o ion en particular en una red o estructura cristalina.

cristalización proceso físico por el cual se forma un sólido cristalino (comúnmente de un líquido o una disolución) en el que los iones, átomos o moléculas están altamente organizados, estableciendo enlaces fuertes y formando una red cristalina.

defectos cristalinos perturbación en la periodicidad de la red de un sólido cristalino.

energía de banda prohibida es la cantidad de energía necesaria para excitar un electrón, liberándolo de su estado de enlace y posibilitando su participación en la conducción electrónica.

estructura cristalina forma sólida de cómo se ordenan y empaquetan los átomos, moléculas, o iones, en donde los patrones de repetición que se extienden en las tres dimensiones del espacio.

halógeno elemento químico no metálico que forma sales minerales al unirse directamente con un metal. Los elementos halógenos son el flúor, el cloro, el bromo, el yodo y el ástato.

1. INTRODUCCIÓN

La ciencia de los materiales es una de las áreas interdisciplinarias que más avances tecnológicos ha permitido. Esto puede observarse tanto en la investigación aeroespacial por el descubrimiento de nuevos combustibles y materiales livianos, como en el desarrollo de baterías y nuevos componentes electrónicos, entre tantos otros.

Uno de los principales desafíos que enfrenta el área es el descubrimiento de nuevos materiales, tradicionalmente logrado mediante la continua prueba y error experimental hasta encontrar una propiedad deseada. Esta metodología de investigación requiere tanto de tiempo como de recursos suficientes para poder explorar el gran abanico de condiciones de reacción[1] posibles. Por otro lado, cuanto más sofisticadas se vuelven las síntesis y técnicas de caracterización, mayor es la complejidad del sistema a analizar y la relación entre las múltiples variables que lo describen. De allí que comprender cuál es la relación entre los reactivos de partida, las condiciones de experimentación y las propiedades del material resultante, demanda nuevos enfoques de análisis y metodologías de estudio.

Desde esta perspectiva, en los últimos años el uso del aprendizaje automático ha tomado gran importancia en el área de materiales como una herramienta para poder extraer conocimiento y relaciones implícitas a partir de los datos (incluso entre experimentaciones normalmente consideradas fallidas[2] en el laboratorio). No obstante, el costo y la dificultad de la obtención de los datos es una de las principales limitaciones a considerar, desafío común en las distintas ciencias experimentales.

Esta tesis de licenciatura se centra en cómo utilizar los datos obtenidos de experimentaciones químicas llevadas a cabo en el laboratorio, con el fin de predecir el resultado de una futura experimentación mediante técnicas de aprendizaje automático. En particular, nos centraremos en experimentaciones que sintetizan perovskita, un compuesto químico muy estudiado desde un enfoque interdisciplinario entre el aprendizaje automático y las ciencias de los materiales. De esta forma, se busca desarrollar modelos que guíen la síntesis experimental en el laboratorio, reduciendo la cantidad de pruebas fallidas, minimizando costos y aportando a la comprensión del sistema químico en estudio.

1.1. Problema químico: la cristalización de perovskita

Dentro del área de materiales las perovskitas no solamente son uno de los materiales más estudiados, sino que además, su investigación ha tenido un crecimiento exponencial en la última década desde que se descubrió su potencial en las celdas fotovoltaicas. Su variedad de composiciones posibles, su sencilla condición de síntesis y sus interesantes propiedades, hacen que las perovskitas sean utilizadas en múltiples aplicaciones. Por ejemplo, en el desarrollo de nuevas tecnologías de iluminación LED, en la creación de lásers, en la actividad industrial por su capacidad catalítica y, recientemente, en el desarrollo de paneles solares [3].

[1] En química, se conoce como condiciones de reacción a los factores físico-químicos que describen el entorno en donde ocurre una reacción química. Típicamente, se hace referencia a las concentraciones de los reactivos, temperatura, presión, entre otros.

[2] Se considera una experimentación fallida a aquella en la cual no se obtiene mayoritariamente el producto de síntesis esperado. Por lo general, se desestiman las condiciones de reacción de una experimentación fallida dado que sus resultados no suelen ser de interés.

Para tomar una dimensión del crecimiento de su investigación, consideremos que la eficiencia de las celdas fotovoltaicas de perovskita -en cuanto a su capacidad de conversión de energía solar a eléctrica-, ha crecido en los últimos 9 años tanto como lo han logrado las celdas de silicio en los últimos 40 años [4]. Para ilustrar esta tendencia Tao *et al* en 2021 consideraron la cantidad de trabajos científicos publicados sobre las perovskitas en la plataforma *Web of Science*[3]. En la figura 1.1 mostramos parte de sus resultados, en donde se observa cómo la cantidad de trabajos con las palabras claves «*machine learning*» y «*perovskite*» («aprendizaje automático» y «perovskita» en inglés) ha superado la cantidad de trabajos con las palabras claves «*machine learning*» y «*materials*» («aprendizaje automático» y «materiales» en inglés) [2]. Se entiende que la aplicación de esta rama de la inteligencia artificial al servicio del descubrimiento de materiales ha tenido un gran impacto en el acelerado crecimiento de la investigación de perovskitas, abriendo las puertas hacia una mejor comprensión del por qué de las características peculiares que posee este versátil material.

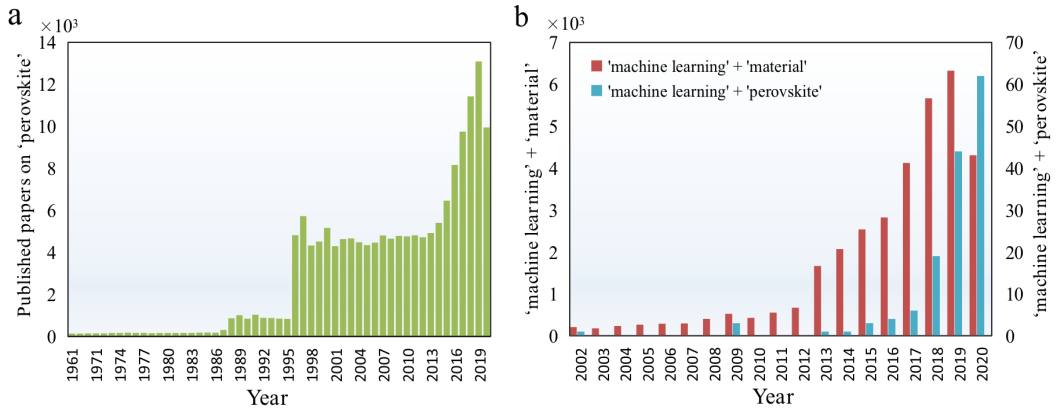

Fig. 1.1: Crecimiento de las publicaciones académicas sobre perovskitas según la cantidad trabajos científicos disponibles a través de la plataforma *Web of Science* (extraído de Tao et al [2]). (a) Cantidad de trabajos que incluyen la palabra clave «*perovskite*» desde 1960 hasta 2020. (b) Comparación de cantidad de trabajos que incluyen las palabras claves «*perovskite*» y «*machine learning*», versus aquellos que incluyen las palabras «*materials*» y «*machine learning*».

En las ciencias de los materiales, las perovskitas son un grupo de compuestos químicos con una estructura cristalina representada como ABX_3, donde A y B son cationes de una y dos cargas positivas respectivamente, y X es un halógeno o ion de oxígeno pequeño (ver figura 1.2). En particular, se dice que se trata de una perovskita híbrida orgánica-inorgánica (conocida como HOIP por sus siglas en inglés, *"hybrid organic–inorganic perovskite"*) cuando se cumple que A es un compuesto orgánico, B un compuesto inorgánico (un metal como Pb^{2+} o Sn^{2+}) y X es un halógeno.

[3] *Web of Science* es una plataforma paga que permite el acceso a múltiples bases de datos que proveen datos de citas y referencias a *journals* académicos, *proceedings* de congresos y conferencias, entre otros.

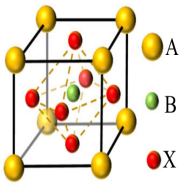

Fig. 1.2: Estructura cristalina cúbica simple de la perovskita (extraído de Tao et al [2]).

Existen múltiples formas de preparar perovskitas, se pueden sintetizar como *films* delgados, microcubos, laminillas o como cristales de diverso tipo, entre otros. Dentro de las formas obtenibles mediante cristalización, se destacan los monocristales dado que permiten hacer una detallada caracterización por técnicas de cristalografía. Esta clase de técnicas es clave puesto que las propiedades físicas de la perovskita -aquellas que permiten que tenga tantas posibles aplicaciones-, responden a los defectos cristalinos de su estructura. Además, los monocristales de perovskita ha mostrado incluso realzar ciertas propiedades relevantes del material, destacándolas de las otras formas posibles de síntesis [3].

En este trabajo se estudia en particular la cristalización de HOIP de ioduro de plomo, composición de perovskita considerada de referencia al ser la primera de esta familia de materiales en mostrar un gran potencial en la conversión energética eficiente [5]. En particular, se estudian HOIPs de esquema híbrido orgánico-inorgánico, en el que se usa plomo como componente inorgánico y donde se varía el componente orgánico entre distintas aminas orgánicas (también denominas organoaminas).

El problema químico abordado consiste en conocer en qué condiciones experimentales se produce la cristalización de HOIP de ioduro de plomo a partir de estudiar patrones en datos experimentales de síntesis de este material. Para ello, no solo se usan datos obtenidos en el laboratorio, sino que también se hace uso de una descripción fisico-químico del componente orgánico, la organoamina.

En resumen, en esta tesis se emplean métodos estadísticos y de aprendizaje automático para conocer en qué condiciones se espera que cristalice la HOIP de ioduro de plomo, así como también para analizar qué propiedades físico-químicas de la amina orgánica usada como reactivo resultan relevantes para los modelos a desarrollar.

1.2. Estado del arte

A lo largo de los 40 años de trayectoria que lleva la investigación en perovskitas, hubo un hito que produjo que las HOIPs sean hoy en día uno de los materiales más estudiados: el descubrimiento de su potencial en el área de la energía solar fotovoltaica.

En 2009 Kojima *et al.* lograron crear las primeras celdas fotovoltaicas de perovskitas (PSC) empleando HOIP de plomo, abriendo lugar a las perovskitas en el área de los materiales fotovoltaicos [7]. Actualmente, el rendimiento de conversión energética más alto alcanzado en PSC es del 25.2 %, cercano al 29.8 %, mejor factor logrado en materiales fotovoltaicos emergentes en general (según el Laboratorio Nacional Estadounidense de Energías Renovables [6]). Si consideramos que en 2009 el factor de conversión inicial conseguido era del 3.8 %, el crecimiento de la eficiencia de las HOIPs como material fotovoltaico resulta muy prometedor para el área, de allí que se investigue qué características tienen las HOIPs de plomo en particular.

Si bien se sabe que el plomo juega un factor clave en la alta performance de la PSC, se estudia reemplazarlo por otros compuestos dado que las celdas fotovoltaicas de estas

perovskitas se degradan fácilmente con la exposición al aire, luz y calor, resultando en productos carcinógenos como el ioduro de plomo (PbI_2). De allí también que se busque sintetizar HOIPs de plomo de mayor estabilidad química y que se degraden a compuestos amenos para el medio ambiente y la salud. Esta es una de las tantas razones por la que se investiga la síntesis de otras composiciones posibles de HOIPs, por ejemplo explorando composiciones que usen organoaminas comúnmente no utilizadas como componente orgánico, entre otros [8].

En este sentido, la aplicación del aprendizaje automático ha posibilitado la creación de modelos que aceleran el descubrimiento de interesantes composiciones estables de HOIPs. Por un lado, existen líneas de investigación que analiza la estabilidad teórica de las HOIPs usando métodos de modelado mecánico-cuántico. De esta forma, mediante la generación de datos sintéticos por métodos de simulación computacional, se crean modelos de predicción de estabilidad por aprendizaje automático. Por ejemplo, Lu *et al.* armaron un *dataset* de energía de banda prohibida (*bandgap energy* en inglés) a partir de la cual construyeron un modelo basado en la técnica Gradiente de Potenciación (*Gradient Boosting* en inglés). Esto no solo les permitió explorar más de 5 mil composiciones posibles de HOIPs, sino que además pudieron encontrar seis candidatas estables a temperatura ambiente con potencial aplicación en celdas solares [2].

Por otro lado, otras líneas de investigación utilizan datos experimentales para generar modelos de predicción de síntesis de HOIPs. En este marco, grupos de investigación con cierta capacidad económica generan datos experimentales de forma sistemática mediante una experimentación automatizada asistida por robótica. Este mecanismo se denomina experimentación de alto rendimiento (*high-throughput experimentation* en inglés), metodología que permite capturar las variables experimentales, reduciendo el sesgo introducido por la manipulación manual [9]. A través de la experimentación de alto rendimiento, en los últimos años, han surgido múltiples *datasets* provenientes de proyectos de investigación de diferentes instituciones [10–13]. Sin embargo, como su tamaño suele ser reducido (del orden de miles) los métodos de aprendizaje automático suelen estar limitados por la poca disponibilidad de datos.

En este trabajo se emplean datos obtenidos de una plataforma de síntesis automática de perovskita, la cual permite realizar experimentaciones de alto rendimiento para múltiples condiciones experimentales y distintos reactivos. En 2020 Zhi Li *et al.* presentaron la plataforma «RAPID, investigación y descubrimiento de perovskita acelerada por robótica», con la cual han generado pruebas de concepto de cómo el aprendizaje automático y la experimentación de alto rendimiento pueden combinarse para el estudio de la cristalización de HOIPs [14]. En su trabajo mostraron cómo el uso de propiedades físico-químicas, en combinación con datos de condiciones experimentales provistos por su plataforma, permite generar modelos de aprendizaje automáticos más confiables que aquellos que usan exclusivamente datos experimentales [15].

1.3. Objetivos y estructura del trabajo

El objetivo general de esta tesis es hacer una contribución al descubrimiento de nuevos materiales, haciendo foco en la predicción de la cristalización de HOIPs de ioduro de plomo desde un punto de vista computacional. En particular, nos enfocaremos en los desafíos que conlleva usar datos experimentales para el desarrollo de sistemas predictivos (como por ejemplo, sesgos y errores en la recolección de datos, disponibilidad, entre otros).

Los objetivos específicos (**OE**) del trabajo son los siguientes:

OE1 Recolectar y procesar datos experimentales correspondientes a la síntesis de HOIPs

OE2 Analizar características de los datos para definir cómo abordar el desarrollo de sistemas predictivos químicos basados en aprendizaje automático

OE3 Proponer modelos de predicción de la cristalización de HOIPs considerando la caracterización de los datos

En cuanto a la estructura del trabajo, en el capítulo Métodos se realiza una introducción teórica a los métodos estadísticos y de aprendizaje automático utilizados.

En el capítulo Fuente de Datos se realiza una descripción del *dataset* empleado en esta tesis. Para ello, se realiza un análisis exploratorio de los datos, analizando las características a ser consideradas para desarrollar los modelos de aprendizaje automático.

En el capítulo Métricas de Evaluación y Errores se efectúa un análisis de los errores que afectan a los modelos propuestos (con el fin de comprender su impacto en las métricas usadas), así como también se definen y justifican las métricas de *performance* a usar en este trabajo.

En el capítulo Desarrollo y Selección de Modelos se exploran distintas algoritmos de aprendizaje y técnicas de ensamble para desarrollar modelos que permitan clasificar experimentos de cristalización de HOIPs de ioduro de plomo.

Por último, en el capítulo Conclusiones y Trabajo, se presentan los resultados generales de esta tesis, así como los aprendizaje realizados, desafíos encontrados e ideas para el trabajo futuro de esta línea de trabajo.

Cabe mencionar que el código e implementación de los métodos usados en esta tesis se encuentran disponibles para su consulta (véase sección A del Apéndice).

2. MÉTODOS DE APRENDIZAJE AUTOMÁTICO

En este capítulo describimos los métodos del aprendizaje automático usados con el objetivo de estudiar la cristalización de la perovskita. Dentro de esta disciplina nos abocamos al aprendizaje supervisado, área que se caracteriza por aproximar una función objetivo desconocida mediante una serie de datos de entrada-salida. En particular, en tareas de clasificación los datos de entrada son instancias representadas por un vector de atributos, mientras que los datos de salida consisten en etiquetas de clasificación.

En el contexto de esta tesis, cada instancia de entrada consta tanto de atributos experimentales como de atributos físico-químicos que, en su conjunto, describen y definen una experimentación específica. En cambio, las etiquetas de salida son la categoría del producto de síntesis obtenido de la experimentación de entrada, habiendo cuatro posibles categorías: (1) sin cristal, (2) polvo fino, (3) pequeños cristales o (4) monocristales.

2.1. Métodos de clasificación supervisada

2.1.1. Estimadores de base

- **K-Vecinos Más Cercanos (KNN)**

K-Vecinos Más Cercanos [16] es un algoritmo de clasificación que usa instancias específicas para hacer sus predicciones, sin necesidad de formular una abstracción propiamente dicha. Su concepto es de los más simples e intuitivos: dada una instancia de evaluación, se computa su distancia (o similaridad) con respecto a todas las instancias de entrenamiento, considerando las k-más cercanas para determinar la etiqueta predicha según cual sea la mayoritaria en su vecindad de entrenamiento.

Una de las particularidades que tiene este algoritmo es que, en un conjunto de entrenamiento desbalanceado, su etiquetado suele verse afectado por la prevalencia de clases mayoritarias. De allí la relevancia de poder considerar distintas maneras de determinar la etiqueta de predicción dada la vecindad de una instancia (por ejemplo, definiendo con mayor criterio la vecindad de una instancia).

- **Árboles de Decisión (DT)**

Los árboles de decisión [17] son modelos de predicción que realizan divisiones en el espacio de los atributos asignando una etiqueta a cada subdivisión final resultante. Se puede identificar dos fases principales en el armado del árbol: la construcción y la poda.

La construcción se realiza de forma recursiva partiendo el conjunto de entrenamiento de modo que la mayor parte de las instancias de una misma clase queden juntas en la misma subdivisión. Dichas particiones se hacen en base a los atributos disponibles, siguiendo un criterio de optimización local que permite diferenciar a las clases a clasificar. De esta forma, cada subdivisión final es considerada una hoja del árbol, la cual tiene asignada una etiqueta de clasificación final.

Por otro lado, la poda consiste en la remoción de hojas o subdivisiones, pudiéndose llevar a cabo antes o después del armado del árbol. En ambos casos, se escoge un criterio

de poda en pos de evitar el sobreajuste sobre las instancias de entrenamiento, mejorando la capacidad de generalización del árbol.

En la figura 2.1 se encuentra a modo de ejemplo una representación del árbol de decisión generado para una clasificación binaria de dos atributos, en donde se aprecian las subdivisiones del espacio de atributos como hojas del árbol. Una vez generado el árbol de decisión, la clasificación de nuevas instancias se basa en determinar la subdivisión correspondiente a la instancia de entrada y su etiqueta asociada (la más frecuente entre las instancias de entrenamiento cubiertas por esa subdivisión). De allí que ante la presencia de un problema de desbalance de clases, se dificulte la distinción entre las minoritarias y las mayoritarias. Por ejemplo, puede que la construcción del árbol termine sin que las ramas alcancen a reconocer la clase minoritaria, o que en caso de hacerlo se pierdan en la poda para evitar el sobreajuste. De allí la relevancia de explorar distintos métodos basados en árboles que estén adaptados a *datasets* desbalanceados.

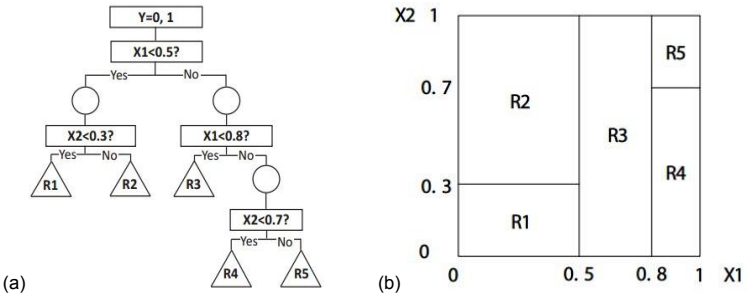

Fig. 2.1: Representación de un ejemplar de árbol de decisión para una clasificación binaria (donde X1 y X2 son los atributos, Y es la variable binaria a predecir y R1-R4 representan las subdivisiones del espacio de atributos). (a) Árbol de decisión de la clasificación. (b) Espacio correspondiente de atributos subdividido. Figuras extraídas de Song & Lu [20]

- **Máquina de Soporte Vectorial (SVM)**

La máquina de soporte vectorial [21] es un algoritmo que permite maximizar una función matemática con respecto a un conjunto de datos. En los problemas de clasificación se puede entender esta técnica mediante los siguientes conceptos: (1) el hiperplano de separación, (2) el hiperplano que maximiza el margen, (3) la adición de un margen suave y (4) la función kernel. Se conoce como hiperplano de separación a la linea recta en un espacio multidimensional que separa las instancias a clasificar. La técnica SVM se diferencia de otros clasificadores que también usan hiperplanos, por la forma en que selecciona el hiperplano de separación, dado que busca maximizar la distancia que existe entre el hiperplano y cualquier instancia de las clases a clasificar. La habilidad de predicción de SVM se basa en la selección de este hiperplano, bajo la suposición de su existencia. Dado que no todos los conjuntos de datos pueden ser separados linealmente, esta técnica considera cierto margen de error conocido como margen suave (*soft margin* en inglés), el cual le permite cierto error de clasificación parametrizable. En cuanto a la función kernel, esta consiste en una transformación matemática que se aplican a los datos para llevarlos a una mayor dimensionalidad, con el fin de poder separar las instancias linealmente en esta mayor dimensión.

2.1.2. Métodos de ensamble

Los métodos de ensamble [22] se caracterizan por entrenar un conjunto de modelos de predicción con el fin de combinarlos, creando así un estimador más robusto denominado *ensamble* (ver figura 2.2). Para ello, estos métodos emplean estimadores de base (también llamados *learners*) que resultan de haber sido entrenados en un conjunto de datos mediante algún algoritmo de aprendizaje (por ejemplo, KNN, SVM, DT, entre otros).

Los métodos de ensambles se pueden clasificar en *homogéneos* o *heterogéneos* según si combinan estimadores de base de un mismo algoritmo de aprendizaje o no. La combinación de los estimadores es una etapa clave en la cual se pueden aplicar distintas reglas de combinación, dando lugar diferentes algoritmos posibles. En general, lo que se busca es poder generar estimadores de base *diversos* y *precisos* según el problema en particular abordado.

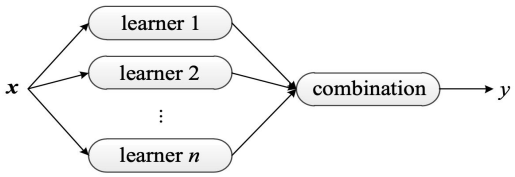

Fig. 2.2: Arquitectura general de un método de ensamble: las instancias de entrada x se usan para entrenar n estimadores de base (o *learners* en inglés) que posteriormente son combinados para generar las predicción de salida y. Extraído de *Ensemble Methods*, Zhi-Hua (2012) [22].

Dentro de los métodos de ensamble hay dos paradigmas de cómo generar los estimadores de base, pudiendo ser de forma secuencial o paralela, también conocidos como *boosting* o *bagging* respectivamente.

El enfoque ***boosting*** se caracteriza por explotar la dependencia entre los estimadores de base, mejorando la *performance* general mediante la disminución de los residuos. Secuencialmente, se los va entrenando y combinando de modo que los últimos estimadores se enfoquen en aprender de los errores cometidos por los anteriores. Para ello, se va ajustando la distribución de los datos de entrenamiento en cada iteración, focalizando en aquellos donde más error se ha cometido.

Por otro lado, en el esquema ***bagging*** se busca poder aprovechar la independencia de los estimadores de base, dado que el error se puede reducir considerablemente combinando estimadores independientes. Si bien técnicamente es difícil evitar la dependencia de los estimadores al ser creados de un mismo conjunto de entrenamiento, introduciendo azar en esta etapa se puede facilitar esta tarea. Además, este enfoque suele ser de interés al ser compatible con un procesamiento paralelo, pudiendo reducir los tiempos de entrenamiento, incluso a mayor volumen de datos.

En *bagging* se entrenan los estimadores a partir de distintos subconjuntos de datos originados de un mismo conjunto total de desarrollo. Para ello se emplea la técnica de muestreo denominada *bootstrap*, la cual consiste en realizar un muestreo con repetición de modo que el conjunto resultante puede repetir múltiples o nula cantidad de veces a cada instancia. Una vez entrenados los estimadores, se realiza una agregación de sus predicciones mediante la cual se obtiene la predicción del ensamble en conjunto. Por ejemplo, en una clasificación se puede hacer una votación y en una regresión se puede realizar un promedio.

En cuanto a qué tipo de estimadores de base usar, los ensambles generados por *bagging* suelen funcionar mejor al usar estimadores *inestables*, puesto que son más susceptibles al

conjunto en donde fueron entrenados. En cambio, los estimadores más estables tienden a ser más similares entre sí, y por lo tanto, más dependientes [23]. Por ejemplo, SVM es conocido como un ejemplo clásico de estimador de base robusto, mientras que los árboles de decisión son ejemplos de estimadores inestables propicios a generar mejores resultados bajo un modelo de ensamble.

Con respecto a la combinación de los estimadores, si bien existen mecanismos sencillos como la votación, también existen otros más complejos que proponen niveles de entrenamiento y combinación. Un ejemplo de estos últimos es ***stacking***, procedimiento en el cual un estimador es entrenado para saber cómo combinar estimadores de base, también denominado como *meta-estimador*. De esta forma se tienen dos niveles de entrenamiento, el primero donde se entrena un conjunto de estimadores de base, y un segundo nivel en el que un *meta-estimador* es entrenado en la salida del primer nivel.

A continuación se introducen dos algoritmos muy utilizados para cada uno de los paradigmas de ensamble previamente mencionados.

- **Gradient Boosting Classifier (GBC)**

Gradient Boosting [24] es un método de ensamble que combina árboles de decisión de forma iterativa, disminuyendo los errores que estos van cometiendo, de modo que el ensamble resultante sea el estimador que minimice los errores cometidos. Para ello este método optimiza una función denominada *función de pérdida* (*loss functions* en inglés), la cual es evaluada en cada iteración para representar el error cometido en esa etapa.

- **Random Forest (RF)**

Random Forest [24] se trata de método de ensamble *bagging* que opera empleando árboles de decisión como estimadores de base o *learners*. Se encuentra dentro del paradigma *bagging* dado que realiza *bootstrap aggregation*, es decir, genera cada *learner* muestreando mediante *bootstrap* y realiza la predicción final agregando las predicciones de los *learners* (tomando la clase más elegida en caso de una clasificación o promediando en caso de regresión). La particularidad de RF es que para cada estimador elije un subconjunto de *features* para definir los atributos a usar del conjunto de entrenamiento, técnica conocida como *feature bagging*. De esta forma se busca disminuir la correlación entre los estimadores de base, haciéndolos más independientes entre sí y favoreciendo el ensamble resultante de la agregación.

2.2. Métodos de clusterización

- **Clusterización jerárquica**

La clusterización jerárquica consiste en una familia de algoritmos que construye un árbol sobre los datos, de modo de agruparlos en distintos subconjuntos en base a un criterio de unión (*linkage*) y una noción de cercanía. Por un lado, se encuentra la raíz del árbol que representa el principal clúster contenedor de todas las instancias del conjunto de datos. Por el otro lado, se hallan las hojas del árbol que representan cada una un clúster de un único elemento. Los nodos intermedios del árbol representan los clústeres intermedios que contienen distintos subconjuntos de datos en base a su cercanía.

La idea principal de este tipo de algoritmos consiste en realizar una agrupación con jerarquía formando *clusters de clusters*. En particular, en este trabajo se considera un enfoque de construcción aglomerativo, el cual se caracteriza por proceder desde las hojas hasta la raíz del árbol, uniendo a los grupos a medida que se determina el nuevo clúster de mayor jerarquía.

Tanto el criterio de unión como la noción de distancia empleada en el algoritmo, pueden ser determinantes para la formación del árbol. Esencialmente, el algoritmo se encarga de mantener un conjunto de clústeres activos, decidiendo en cada etapa qué par de clústeres unir en base a su cercanía. Luego, se remueve ambos individualmente del conjunto de clústeres activos y se los agrega unidos como un único clúster. A continuación se presentan los métodos de enlace más comunes (véase figura 2.3):

- Enlace simple: se basa en los dos puntos más próximos entre dos clústeres.
- Enlace completo: se basa en los dos puntos más lejanos entre dos clústeres.
- Enlace promedio: considera el promedio de las distancias entre cada elemento de un clúster y cada elemento del otro.
- Centroide: emplea el centroide de cada clúster para poder determinar la distancia entre sí.

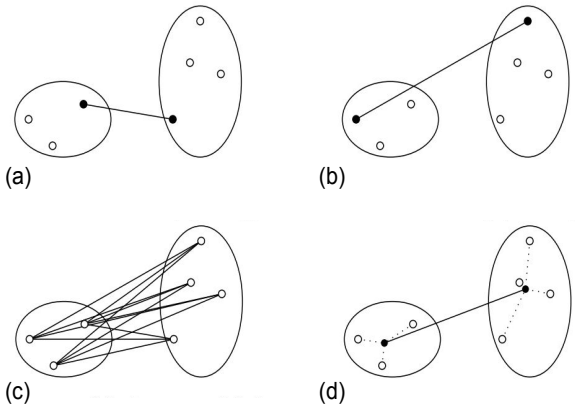

Fig. 2.3: Criterios de unión de clústeres (*linkages*): (a) Enlace simple. (b) Enlace completo. (c) Enlace promedio. (d) Centroide. Tomado de *Hierarchical Clustering – Elements of Machine Learning*, R.P. Adams (2018) [26].

.

2.3. Métodos de preprocesamiento

2.3.1. Transformación de atributos

Dado que en aprendizaje automático hay técnicas que solo pueden trabajar con atributos de valores numéricos, es común utilizar técnicas de transformación de atributos que permitan mapear valores categóricos a numéricos. A este procesamiento de las variables se lo conoce en inglés como *feature-encoding*. Entre las distintas técnicas conocidas se encuentra el *one-hot-encoding*, el cual consiste en asignar un vector a cada categoría de modo que cada dimensión del vector indique si un determinado valor de categoría es usado o no. De esta forma, se considera un atributo nuevo en los datos por cada categoría existente en la variable a transformar.

2.4. Selección de modelos

Generalmente los algoritmos de aprendizaje requieren de cierta configuración para su entrenamiento, comúnmente denominados *hiperparámetros*. Para poder evaluar cada configuración de hiperparámetros, se debe entrenar el modelo en un conjunto de datos y evaluar su desempeño posteriormente en datos no empleados para su entrenamiento. Para ello se suele resguardar un subconjunto de datos para la evaluación final, el cual no es usado en toda la etapa de desarrollo de modelos. En cambio, los datos de desarrollo se usan para encontrar los hiperparámetros óptimos, considerando distintas divisiones de este conjunto de datos según lo requiera el contexto de aplicación del modelo final y las características del *dataset*.

A continuación se introduce en qué consiste la búsqueda de hiperparámetros, así como cuáles son las posibles divisiones que se pueden hacer del conjunto de datos de desarrollo, para poder validar el desempeño de los modelos en datos no usados para entrenar.

2.4.1. Búsqueda de hiperparámetros

Entre las distintas formas de explorar el espacio de hiperparámetros de los algoritmos de aprendizaje, en este trabajo se consideraron principalmente la búsqueda exhaustiva y aleatoria (conocidos en inglés como *grid-search* y *randomized grid-search* respectivamente) [24]. Ambas técnicas se basan en una grilla de exploración de los hiperparámetros, aunque su diferencia radica en cómo se efectúa la exploración. La búsqueda exhaustiva se caracteriza por considerar cada combinación posible de hiperparámetros de la grilla, tomando como parámetros finales a la configuración con mejor desempeño. En cambio, la búsqueda aleatoria considera una distribución por sobre la grilla proporcionada para muestrear posibles configuraciones de hiperparámetros. Una diferencia importante es que la búsqueda exhaustiva puede ser costosa computacionalmente si se tiene un modelo complejo con múltiples hiperparámetros y una grilla amplia de búsqueda. En cambio en la búsqueda aleatoria se puede manipular la cantidad de configuraciones exploradas, pudiendo aumentar la complejidad de hiperparámetros sin necesariamente un alto costo computacional.

2.4.2. K-Validación cruzada

La k-validación cruzada [24] se trata de una técnica de selección de modelos que permite evaluar cada instancia del conjunto de datos de desarrollo. Esta técnica consiste en realizar una partición de los datos en k conjuntos disjuntos (en inglés cada partición se denomina *split* y cada conjunto disjunto se denomina *fold*).

En la figura 2.4 se puede encontrar un diagrama de cómo se realiza la división de datos en la validación cruzada. En cada división, se reserva uno de los *folds* para validación, mientras que los k-1 restantes se usan para entrenar el modelo. De esta forma, se generan en total k modelos entrenados con una misma configuración, de modo que al final cada configuración del modelo se valida en el total de datos de desarrollo. Esto se debe dado que cada modelo es evaluado en su *fold* de validación correspondiente, conformando estos conjuntos disjuntos el total de datos de desarrollo.

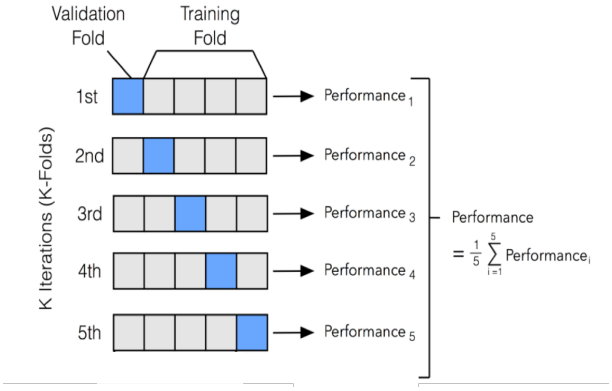

Fig. 2.4: Partición de los datos para la selección de modelos mediante validación cruzada. Por cada iteración, se tiene una porción de datos (o *fold* en inglés) usado para la validación (en azul), mientras que las partes restantes se usan como conjunto de entrenamiento (en gris). Tomado de *Python Machine Learning (2017)*[25].

En particular en este trabajo el desempeño final de la validación cruzada se computó mediante el promedio de los resultados de cada modelo al evaluarse en su conjunto de validación correspondiente.

2.5. Evaluación de modelos de clasificación

La determinación del desempeño de una clasificación es un punto clave en el desarrollo de modelos predictivos. Para ello, existen múltiples métricas que se pueden usar para medir la *performance* de un modelo según las particularidades del problema abordado. Si bien algunas de ellas se basan en umbrales, *rankings* e incluso probabilidades, muchas de ellas se calculan a partir de lo que se conoce como *matriz de confusión*. En esta tesis utilizamos matrices de confusión para evaluar los modelos desarrollados, analizando cómo características de los datos, como su desbalance de clases y la acotada cantidad disponible, pueden llegar a influir en estas matrices.

A continuación se realiza una introducción teórica a dichas características, tomando el marco teórico desarrollado por Luque *et al.* [30] al estudiar el impacto que tiene el desbalance de clases en métricas de clasificación basadas en matrices de confusión.

2.5.1. Matriz de Confusión

Para poder definir una matriz de confusión, Luque *et al* consideran formalmente un conjunto de datos de evaluación $\mathscr{D}_m$, un clasificador binario $\mathscr{C}$ y un conjunto de clases Θ, donde m es la cantidad de instancias que conforman el conjunto de evaluación y donde cada instancia tiene una clase asignada del conjunto Θ. En este contexto, se entiende por *confusión* a una operación errónea del clasificador $\mathscr{C}$ al asignar una clase distinta a la etiqueta correspondiente. De allí que una matriz de confusión se trate de una estructura matricial que muestra las confusiones realizadas por un clasificador $\mathscr{C}$ en un conjunto $\mathscr{D}_m$.

Como en este trabajo se realiza una clasificación binaria entre las clases negativa (N) y positiva (P), consideramos nuestro conjunto de clases como $\Theta = \{N, P\}$. De este modo,

la matriz de confusión queda representada por la siguiente expresión:

$$CM \equiv \begin{bmatrix} m_{N,N} & m_{N,P} \\ m_{P,N} & m_{P,P} \end{bmatrix} \tag{2.1}$$

en donde $m_{i,j}$ es la cantidad de elementos pertenecientes a la clase θ_i pero que fueron clasificados como θ_j.

Notemos que la suma de elementos positivos totales m_P y elementos negativos totales m_N es el tamaño de $\mathscr{D}$ (es decir, $m_P + m_N = m$). Además, tanto m_N como m_P se pueden expresar según la proporción etiquetada correctamente o no por el clasificador $\mathscr{C}$. Por ejemplo, se puede desdoblar la cantidad de instancias positivas como: $m_P = m_{P,N} + m_{P,P}$. De allí que se puede expresar la matriz de confusión de la siguiente forma:

$$CM \equiv \begin{bmatrix} m_{N,N} & m_N - m_{N,N} \\ m_P - m_{P,P} & m_{P,P} \end{bmatrix} \tag{2.2}$$

Dado que se desea estudiar la influencia del tamaño del conjunto de datos y del desbalance de clases en las matrices de confusión, Luque *et al* trabajan sobre esta expresión para poder escribirla en función de m, de la proporción de cada clase y de la proporción de instancias clasificadas correctamente dentro de cada clase.

Para ello, en primer lugar consideran a $\lambda_{i,j}$ como la proporción de elementos clasificados como θ_j cuya etiqueta correspondiente es θ_i (es decir, $\lambda_{i,j} = m_{i,j}/m_i$). Luego, reemplazando en la fórmula (2.2) se obtiene que:

$$CM \equiv \begin{bmatrix} \lambda_{N,N} m_N & \lambda_{N,P} m_N \\ \lambda_{P,N} m_P & \lambda_{P,P} m_P \end{bmatrix} \equiv \begin{bmatrix} \lambda_{N,N} m_N & (1 - \lambda_{N,N}) m_N \\ (1 - \lambda_{P,P}) m_P & \lambda_{P,P} m_P \end{bmatrix} \tag{2.3}$$

Esta definición se puede reescribir en función de m, despegando la cantidad de instancias de la clase negativa en función de la cantidad de la clase positiva (o viceversa). Para ello, se considera a π_i como la proporción de la clase θ_i, es decir, $\pi_i = m_i/m$. De esta forma, la matriz de confusión queda expresada en función de m y π_i, siendo θ_i una de las posibles clases (negativa o positiva). Por ejemplo, considerando $\pi_i = \pi_P$, se llega a la siguiente fórmula:

$$CM \equiv \begin{bmatrix} \lambda_{N,N}(m - m\pi_P) & (1 - \lambda_{N,N})(m - m\pi_P) \\ (1 - \lambda_{P,P}) m\pi_P & \lambda_{P,P} m\pi_P \end{bmatrix} \tag{2.4}$$

Habiendo introducido el formalismo de una matriz de confusión desarrollado por Luque *et al.* [30], definimos lo que se entiende como desbalance de clases para este autor a continuación.

2.5.2. Desbalance de clases

El desbalance de clases de un conjunto de datos se entiende como una diferencia significativa entre la cantidad de datos de etiqueta positiva contra la cantidad de datos de etiqueta negativa. Entre las distintas formalizaciones posibles, en este trabajo se considera el término denominado *coeficiente de desbalanceo* (δ), definido para una clase K en una clasificación binaria de la siguiente forma:

$$\delta_K = 2 \cdot \pi_K - 1 \tag{2.5}$$

Este coeficiente se encuentra en el rango $[-1, 1]$, donde el cero representa un balance perfecto de las clases positiva y negativa, mientras que el -1 y 1 representan un desbalance extremo hacia la clase negativa y positiva respectivamente. De esta forma, se cumple que la suma de los coeficientes de desbalanceo de ambas clases está balanceada por definición:

$$\delta_N + \delta_P = (2 \cdot \pi_N - 1) + (2 \cdot \pi_P - 1) = 2 \cdot (\pi_N + \pi_P) - 2 = 0 \tag{2.6}$$

En particular, en el marco de este trabajo se considera $\delta = \delta_P$ al ser la clase positiva de mayor relevancia y siguiendo la definición provista en (2.4) al considerar $\pi_i = \pi_P$. De esta forma, en base a las fórmulas 2.4 y 2.5 se deduce que una matriz de confusión CM se puede redefinir como una función $CM(\lambda_{N,N}, \lambda_{P,P}, m, \delta)$, al despejar π_P en función de δ. En el capítulo Métricas de Evaluación y Errores usaremos esta expresión para definir las matrices de confusión de estudio.

2.5.3. Métricas de clasificación

En este trabajo para evaluar los modelos de clasificación consideramos métricas basadas en la matriz de confusión, haciendo foco en aquellas que fuesen usualmente empleadas en clasificaciones binarias. En la tabla 2.1 se encuentran las métricas consideradas, junto a su definición formal.

Métricas		
Símbolo	Nombre	Definición
ACC	exactitud (*accuracy*)	$\frac{TP+TN}{TP+TN+FN+FP}$
PRC	precisión (*precision*)	$\frac{TP}{TP+FP}$
SNS	sensibilidad (*recall*)	$\frac{TP}{TP+FN}$
$B\text{-}ACC$	exactitud balanceada (*balanced accuracy*)	$\frac{1}{2} \cdot (\frac{TP}{P} + \frac{TN}{N})$
F_1	F1	$\frac{PRC \cdot SNS}{PRC+SNS}$
MCC	correlación de Matthew	$\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)}}$

Tab. 2.1: Métricas para clasificaciones binarias binaria consideradas.

Entre las métricas consideradas se encuentra el *accuracy*, la cual si bien es una de las métricas más populares utilizadas en el aprendizaje automático, en un conjunto de datos desbalanceado de clases no logra captar el impacto del error de clasificación de la clase minoritaria [31, 32].

También se consideraron otras métricas como el *balanced accuracy*, variante que pondera la performance relativa al tamaño de la clase, el F1 considerado como un promedio harmónico entre el *precision* y el *recall*, y otras como el coeficiente de Matthew, métrica que recientemente ha adquirido mayor relevancia para conjuntos de datos desbalanceados en particular.

El coeficiente de Matthew es una métrica que ha mostrado tener ciertas ventajas por sobre *F1* y *precision*, dando un puntaje más confiable estadísticamente, y siendo proporcional tanto al tamaño de la clase positiva como la negativa [33]. Además, dado que el grupo de investigación que originó los datos usados en esta tesis utilizó el coeficiente de

Matthew como su métrica de referencia, en este trabajo se la consideró puesto que permitiría poder comparar los resultados finales con otros existentes de este mismo conjunto de datos. Por último, cabe mencionar que para facilitar la comparación de Matthew con otras métricas, se tuvo en cuenta su versión normalizada llevando su rango desde el intervalo $[-1, 1]$ hacia el $[0, 1]$, de la siguiente manera:

$$MCC_n = \frac{MCC + 1}{2} \tag{2.7}$$

Por otro lado, volviendo a la formalización tomada de Luque *et al.* [30] sobre el coeficiente de desbalanceo (δ) y la matriz de confusión como función $CM(\lambda_{N,N}; \lambda_{P,P}; m; \delta)$, se puede notar que todas estas métricas se pueden expresar en función de estos mismos parámetros. De modo que una métrica se puede considerar como una función $\mu = \mu(\lambda_{N,N}; \lambda_{P,P}; m; \delta)$, en donde se ve que su valor no solo depende del desempeño del clasificador (definido por $\lambda_{N,N}$ y $\lambda_{P,P}$), sino que también de la cantidad de instancias y el coeficiente de desbalanceo de clases.

3. FUENTE DE DATOS

Para este trabajo empleamos un *dataset* provisto por la plataforma de generación de datos experimentales denominada *RAPID* (ver sección Estado del arte). Este *dataset* se creó con el fin de estudiar la síntesis de HOIPs mediante cristalización, para el cual se usaron distintas aminas orgánicas como reactivos posibles de partida.

En particular, Pendleton *et al.* en base a dicho conjunto de datos investigaron cómo se podían mejorar los modelos de aprendizaje automático que emplean únicamente datos experimentales, mediante el uso complementario de descriptores físico-químicos de la amina empleada en cada síntesis [15]. Como en esta tesis se busca mejorar los modelos existentes, tomamos como punto de inicio este trabajo de investigación con el objetivo de mejorar en términos metodológicos los modelos propuestos.

3.1. Introducción

El *dataset* usado consiste de experimentaciones de síntesis de HOIPs de ioduro de plomo, en donde cada una de estas emplea una amina orgánica en particular. En este contexto, decimos que una experimentación consiste en:

- datos de las condiciones experimentales de reacción (como las concentraciones de reactivos, la temperatura de reacción, los tiempos de agitación y reacción, entre otros)
- descriptores fisico-químicos de la amina orgánica (por ejemplo, cantidad de átomos de carbono, cantidad de anillos aromáticos, área superficial de reacción, etc.)

Asimismo, cada experimentación se encuentra clasificada según la calidad del cristal obtenido como producto de la síntesis. Se emplearon cuatro categorías posibles para denominar la calidad del cristal obtenido: (1) solución acuosa sin cristal alguno, (2) polvo fino, (3) pequeños cristales y (4) monocristales bien definidos (véase figura 3.1).

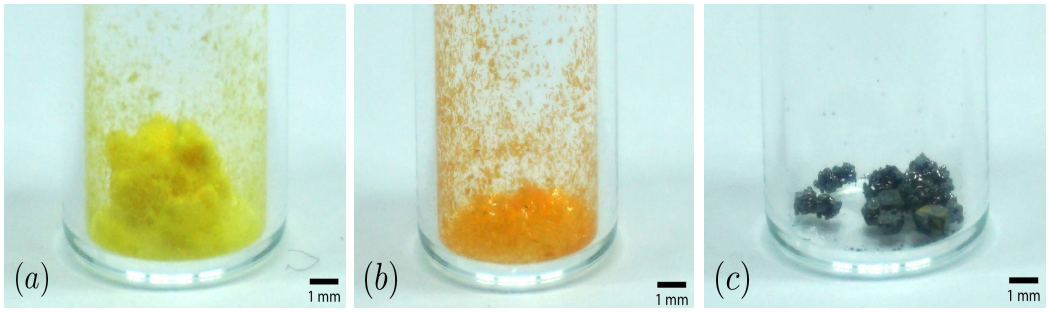

Fig. 3.1: Representación fotográfica de las categorías de la calidad del cristal: (a) polvo fino, (b) pequeños cristales, (c) monocristales.

Para simplificar la comprensión de la síntesis consideremos que preparar una perovskita HOIPs de ioduro de plomo requiere de tres reactivos de partida:

- un compuesto orgánico: organoaminas

- iones inorgánicos: de plomo y iodo
- un compuesto ácido: ácido fórmico

Notemos que tanto el compuesto orgánico como los iones inorgánicos dan lugar a la estructura cristalina $\texttt{ABX}_3$ de las HOIPs (véase figura 1.2), siendo el compuesto orgánico el que da origen al componente `A` y los dos iones inorgánicos a los componentes `B` y `X` (plomo y iodo respectivamente). Por su lado, el compuesto ácido establece el medio de reacción donde se producirá la cristalización.

Entre los datos disponibles, se encuentran las concentraciones iniciales de estos tres reactivos de partida utilizados. Además, se hallan la temperatura en la cual se produjo la reacción, el tiempo de agitación y el tiempo de reacción total.

En cuanto a la descripción físico-química de la amina orgánica, el grupo de investigación que realizó la construcción del *dataset* lo hizo apelando a su *criterio químico* para la selección del subconjunto de descriptores relevantes a considerar. Entre ellos se encuentran más de 50 descriptores que refieren a la composición química, los grupos funcionales presentes, las áreas y volúmenes de interacción entre partículas, entre otros.

Por último, cabe mencionar que los datos disponibles pertenecen a experimentaciones llevadas a cabo en distintos solventes de reacción: GBL (γ-butirolactona), DMSO (dimetilsulfóxido) y DMF (dimetilformida). Si bien el solvente principalmente usado fue GBL, la variabilidad del solvente se debe a que se buscó que la amina orgánica sea soluble en el medio de reacción (habiendo entre la lista de aminas orgánicas exploradas, varias no solubles en GBL). De allí que se usaran otros solventes conocidos para poder llevar a cabo la experimentación.

A continuación, habiendo introducido la fuente de datos usada en términos generales, se hace una descripción estadística del conjunto de datos con el fin de explorar la calidad del *dataset*, así como comprender cuál es la cobertura que se tiene de las variables disponibles.

3.2. Descripción estadística

El *dataset* está compuesto por un conjunto de variables predictoras y una única variable objetivo de tipo ordinal, la cual describe la categoría del producto cristalino obtenido. Las variables predictoras se pueden dividir en los siguientes grupos:

- `FEAT_RXN`: aquellas correspondientes a las condiciones de reacción (8 disponibles)
- `FEAT_PHYCHEM`: aquellas que describen a la amina orgánica usada mediante propiedades físico-químicas (67 disponibles)

Por practicidad, denominaremos como *organoaminas* a las aminas orgánicas, *features* a las variables predictoras y *target* a la variable objetivo.

Entre las *features* del grupo llamado `FEAT_RXN`, se destacan las concentraciones de los reactivos de reacción (componentes inorgánico, orgánico y ácido), variables continuas exploradas en un rango acotado y distinto para cada reactivo (véase sus distribuciones en la figura 3.2). Por un lado, las distribuciones de los reactivos inorgánico y orgánico (a la izquierda y centro de la figura) presentan una asimetría positiva hacia la derecha, habiendo un sesgo de la distribución hacia los valores bajos. En cambio, la distribución del componente ácido (a la derecha en la figura) mostró tener una simetría negativa, con valores sesgados hacia valores altos.

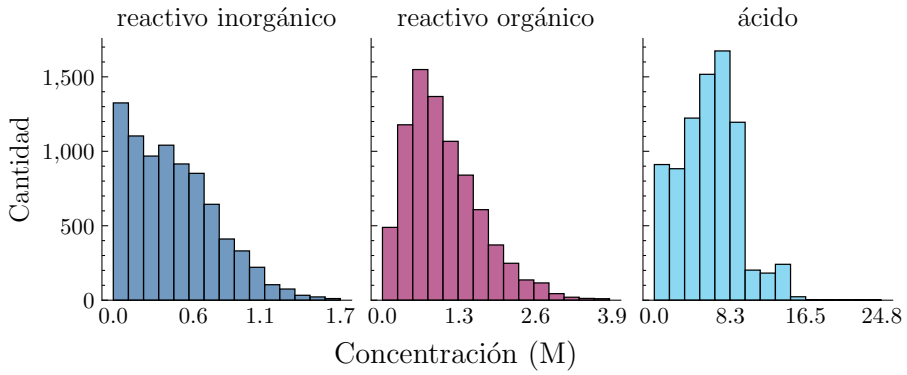

Fig. 3.2: Distribución de las concentraciones (en molaridad) de los reactivos del *dataset* RAPID: componente inorgánico (ioduro de plomo II), orgánico (organoamina) y ácido (ácido fórmico) [14].

De acuerdo al grupo de investigación que dio origen a este conjunto de datos, como el componente inorgánico se mezcló con el orgánico para facilitar su disolución en el solvente, su concentración se encontró limitada por la solubilidad del componente en la cantidad empleada de reactivo orgánico. De allí que este reactivo tenga valores de concentración más bajos que el resto, explorándose principalmente concentraciones por debajo de 1 molar. Por otro lado, la distribución del reactivo orgánico se encontró limitada por su solubilidad en el solvente de reacción, alcanzando un rango de 0 molar a 2.6 molar. En cambio, para el componente ácido se pudo explorar su concentración en un espectro mucho más amplio, dado que se no tuvo una limitación sobre su solubilidad. Para ejemplificar esto, notemos que se tienen aproximadamente 6 mil datos experimentales para concentraciones de ácido que van desde 0 molar a 8 molar, concentración a partir de la que disminuyen abruptamente los datos experimentales disponibles, habiendo solamente 600 datos aproximadamente en el rango de 8 molar a 16 molar.

En cuanto al resto de los *features* del grupo `FEAT_RXN`, se tienen variables que indican condiciones de reacción como la temperatura, la velocidad de agitación, el solvente y la organoamina empleada. La primera variable se trata de una variable nominal que indica qué solvente de reacción se usó entre las tres opciones posibles (GBL, DMSO o DMF). Las primeras dos se tratan de variables numéricas que indican la temperatura en grados Celcius y las revoluciones por minuto con las que se realizó cada reacción, respectivamente. En cambio, para indicar tanto el solvente como la organoamina, se usó una variable nominal que funciona como identificador internacional de compuestos químicos: el *Inchi Key*, [18]. Esta clase de identificadores es comúnmente usado en la química computacional puesto que existen múltiples bibliotecas ya desarrolladas que facilitan el estudio físico-químico de compuestos químicos en general.

En la sección B del Apéndice se encuentra un listado de todas las *features* que componen el *dataset*, junto a una breve descripción de cada uno de sus *features*.

Un aspecto importante a considerar sobre los datos disponibles en el *dataset* es la existencia de sesgos o desbalances que puedan condicionar los modelos predictivos a realizar. Por ejemplo, al analizar la cantidad de instancias disponibles por solvente de reacción se puede identificar un solvente mayoritario (GBL), con respecto a otros dos minoritarios (DMSO y DMF) para los cuales hay menor cantidad de datos disponibles. Como se mencionó anteriormente, dado que la organoamina empleada como componente orgánico debía ser soluble en el medio de reacción, no se pudo usar un único solvente para todos los

reactivos orgánicos dada su distinta solubilidad. De allí que se hayan empleado distintos solventes con el fin de garantizar que la organoamina sea soluble en el medio de reacción. Cabe mencionar que ninguna experimentación fue llevada a cabo en distintos solventes, por lo que cada solvente tiene un conjunto de organoaminas asignado entre los cuales no hay superposición.

Tipo de Variables	Cantidad
Condiciones de Reacción (FEAT_RXN)	
concentración de reactivos (orgánico, inorgánico, ácido)	3
solvente (GBL, DMSO o DMF)	1
temperatura de reacción	1
tiempos de agitación	2
organoamina ID	1
Descriptor de Organoamina (FEAT_PHYCHEM)	
propiedades FQ[1]	61

[1] E.g cantidad de grupos funcionales, donores o aceptores, cantidad de enlaces, etc

Fig. 3.3: Resumen de variables predictoras disponibles en el *dataset* RAPID [1], tras realizada una curación de los datos. Se encuentran categorizadas en los grupos `FEAT_RXN` y `FEAT_PHYCHEM`, *features* descriptoras de las condiciones de reacción y de la organoamina, respectivamente.

En la tabla 3.4 se puede observar la cantidad de datos por cada solvente, en donde se puede notar que hay un desbalance importante tanto con respecto a la cantidad de experimentaciones llevadas a cabo, como a la cantidad de organoaminas exploradas, siendo GBL el solvente con mayor cantidad de datos. Este tipo de desbalance por solvente se considera una de las principales características del *dataset*, puesto que se debe tener en cuenta si se quiere realizar un modelo que prediga para un solvente en particular de modo específico, o si se desea priorizar la versatilidad estudiando modelos que predigan para múltiples clases de solvente. Bien puede resultar que conocer el solvente en realidad no sea relevante para los modelos de predicción entrenados en base a este conjunto de datos, así como también puede ocurrir lo contrario. Este último caso indicaría que la relación entre la distribución de las concentraciones de los reactivos y la variable *target*, depende de cierta forma del solvente. Aunque también existe la posibilidad de que a pesar de haber una relación dependiente del solvente, esta no sea apreciable en estos datos en particular, y por ende, no se pueda desarrollar modelos desde una perspectiva «multisolvente».

Por otro lado, otra característica relevante del *dataset* es el desbalance de clases con respecto a la variable `target`, es decir, con respecto a la calidad del cristal). En la figura 3.5 se aprecia que más de la mitad de las experimentaciones exploradas no generó producto alguno de reacción, siendo casi solo dos de cada diez experimentaciones las que produjeron satisfactoriamente cristales bien formados, es decir, monocristales. La cantidad restante corresponde a experimentaciones que dieron un producto de calidad intermedia, donde se aprecian ciertas características cristalinas (polvo o pequeños cristales) pero que no alcanzan a ser el monocristal deseado.

		experimentos	
Solvente	#organoaminas	cantidad	porcentaje (%)
GBL	29	6096	75.7
DMSO	6	864	10.7
DMF	9	1096	13.6
Total	44	8,056	–

Fig. 3.4: Cantidad de experimentaciones y organoaminas empleadas en total y según el solvente de reacción. GBL, DMSO y DMF denotan γ-butirolactona, dimetilsulfóxido y dimetilformida, respectivamente. No hay superposición entre las organoaminas usadas para cada solvente.

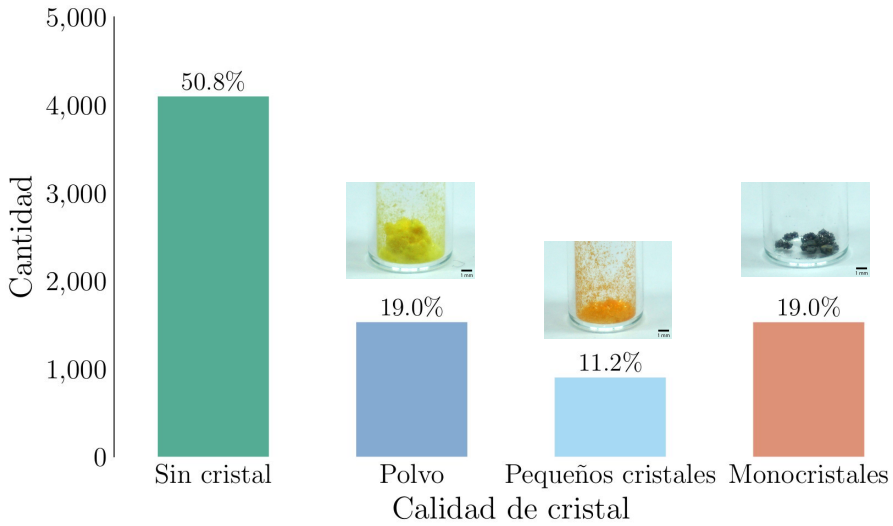

Fig. 3.5: Cantidad de experimentaciones disponibles en el *dataset* según la categoría de calidad del cristal.

De esta forma queda en evidencia la dificultad que conlleva encontrar una combinación apropiada entre las condiciones de reacción y el componente orgánico, de modo que se logre sintetizar las HOIPs en forma de monocristales satisfactoriamente.

Además de las consideraciones a tener en cuenta al trabajar con datos experimentales sesgados en su recolección, otro aspecto a considerar es cómo se modela la síntesis de HOIPs como un problema de clasificación de experimentos químicos en base al producto de síntesis. En base a la granularidad y agrupamiento que se realice de los experimentos, se puede redefinir la variable `target` en múltiples categorías de la calidad del cristal. Por ejemplo, Pendleton *et al.* en su trabajo consideraron el conjunto de datos una binarización de la variable `target`, haciendo énfasis en poder discernir entre monocristales bien formados y el *resto* de productos posibles (polvo cristalino, pequeños cristales, entre otros). De allí que consideraran sistemas de clasificación binaria que usaran las siguientes etiquetas: `no cristaliza` (incluye a las categorías *sin producto*, *polvo* y *pequeños cristales*) y `cristaliza` (incluye únicamente a la categoría *monocristales*).

Al analizar los datos basándonos en esta categorización binaria, se puede observar que

el desbalance de clases no se da de forma uniforme entre las organoaminas, sino que se pueden identificar dos grandes grupos de compuestos orgánicos:

- **Organoaminas tradicionales**: aquellas comúnmente empleadas en la literatura académica, cuyo desbalance hacia la clase `No cristaliza` es del 75 % a 80 % aproximadamente.
- **Organoaminas no tradicionales**: aquellas que no se suelen utilizar para esta síntesis en la literatura académica, cuyo desbalance de clases no sigue un patrón en particular (por ejemplo, tienen clases binarias balanceadas, o tienen un desbalance a favor de la clase `Cristaliza`, entre otros). Su cantidad de datos suele ser considerablemente más pequeña que las organoaminas tradicionales.

En la figura 3.6 se puede observar una comparación de qué tan distinta puede ser la distribución de las clases de la variables `target` según si se trata de una organoamina del grupo tradicional o no. En la figura 3.6.a se tienen tres reactivos orgánicos usualmente utilizados en la síntesis de HOIPs, para los cuales se observa un desbalance de la clase en la variable `target` en una proporción de 8:2 (`no cristaliza`:`cristaliza`), además se puede notar una cantidad considerable de experimentos por cada organoamina (más de 600 experimentaciones por cada una). En cambio, en la figura 3.6.b se observa un conjunto de organoaminas raramente utilizadas para la síntesis de perovskitas. Como se puede notar, en ciertos casos no presentan datos para alguna de las clases de la variable `target`, y en otros casos la relación `no cristaliza:cristaliza` invierte la proporción esperada (por lo general se espera que existan más casos que no cristalizan que los que sí lo hacen).

De este modo, se puede ver que el *dataset* empleado se encuentra sesgado con respecto a las organoaminas utilizadas para la experimentación, no habiéndose realizado la misma exploración de las condiciones de reacción para todos los reactivos orgánicos experimentados. Esto hace que no se puedan considerar los datos experimentales disponibles de cada organoamina como representativos de su reactividad, punto a considerar a la hora de analizar qué aplicación y en qué condiciones es pertinente usar sistemas predictivos basados en esta clase de *datasets*.

En cuanto a las *features* del grupo `FEAT_PHYCHEM`, descriptores físico-químicos del reactivo orgánico, se realizó una curación inicial de los datos descartando aquellos que tuviesen valores constantes para todas las organoaminas usadas, reteniendo finalmente 61 *features*. Cabe resaltar que no se encontró ningún valor faltante o nulo que requiriese tratamiento alguno. En la tabla 3.3 se encuentra un resumen del total de variables disponibles tras esta curación inicial de los datos.

Dependiendo de la naturaleza del tipo de dato que representan, las 61 propiedades físico-químicas se clasificaron como variables binarias, ordinales o continuas. Los descriptores que aludiesen a la presencia o no de cierta propiedad química se trataron como variables binarias. Por otro lado, aquellos descriptores estructurales de la organoamina que contaran cantidades relevantes (por ejemplo, cantidad de átomos de cierto tipo, cantidad de grupos funcionales, entre otros), fueron considerados como variables ordinales. Respecto a las variables continuas, se tuvieron en cuenta aquellas *features* que hicieran referencia a propiedades físico-químicas extensivas, como lo son las áreas y volúmenes que presentan algún tipo de carga electromagnética. En la figura 3.7 se encuentra la cantidad de propiedades físico-químicas que se tiene para cada tipo de variable: binaria, ordinal o continua.

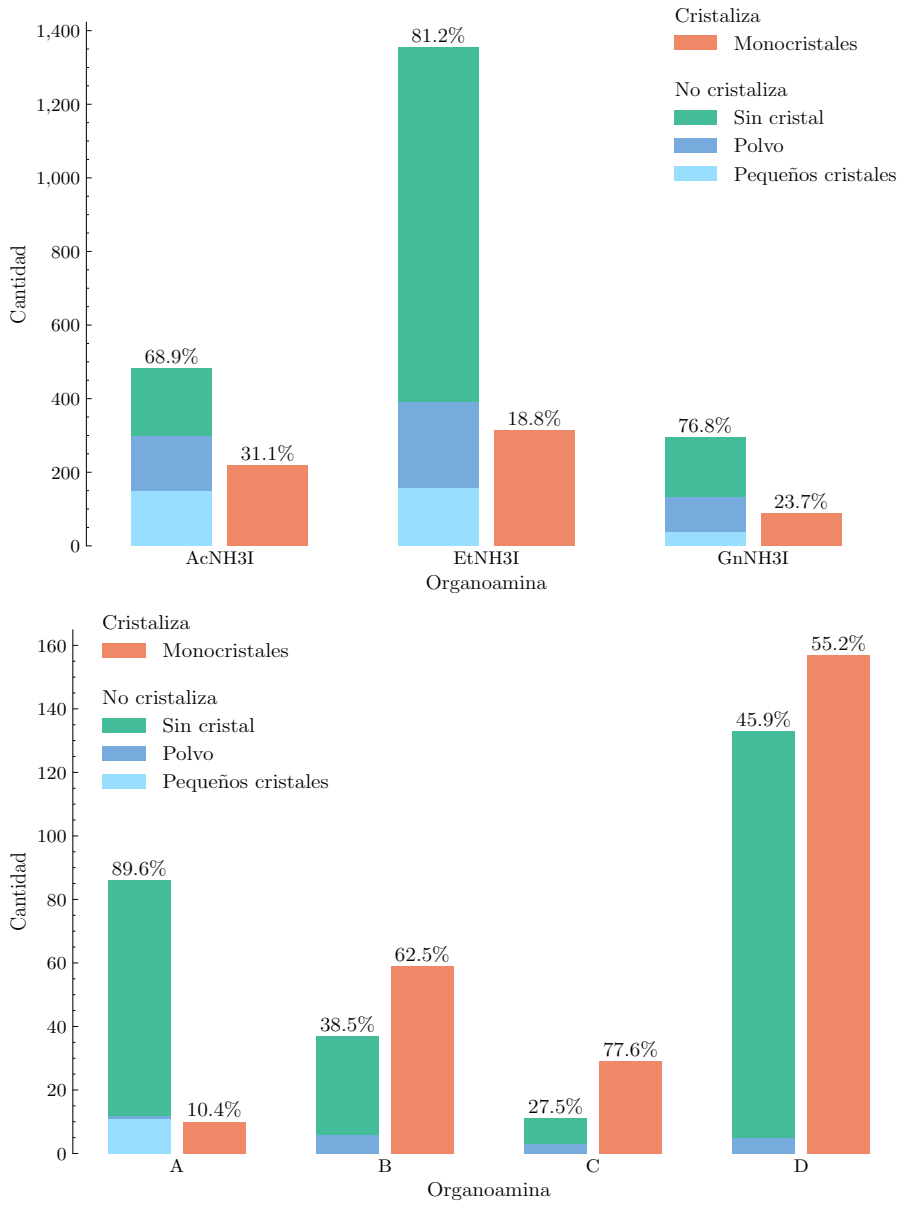

Fig. 3.6: Comparación entre organoaminas según el desbalance en la variable *target* (calidad del cristal) y según la cantidad de datos disponibles. (a) Organoaminas tradicionales **(b)** Organoaminas no tradicionales. A: EthylenediamineDihydriodide, B: CyclohexylmethylammoniumIodide, C: CyclohexylammoniumIodide, D: iButylammoniumIodide.

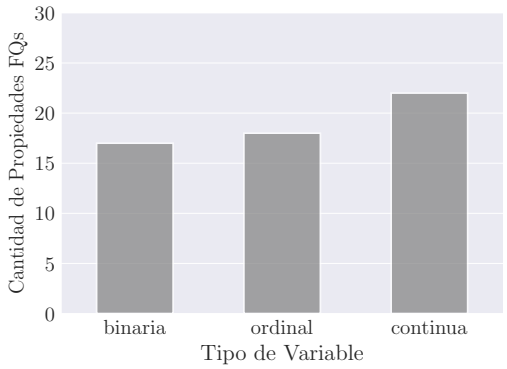

Fig. 3.7: Cantidad de propiedades físico-químicas según el tipo de variable: binaria, ordinal o continua.

La lista completa de propiedades físico-químicas se encuentra disponible en la sección B del apéndice. Allí se puede puede encontrar una estadística descriptiva de cada una, así como también se explica brevemente qué denota cada una químicamente.

En relación a las organoaminas empleadas para las experimentaciones, se las puede describir químicamente considerando qué grupos amino posee cada una: primario, secundario, terciario, cuaternario o aromático. Cabe resaltar que cada organoamina puede tener más de un grupo amino presente en su estructura.

En la figura 3.8.a se pueden ver la cantidad de organoaminas que se tiene por cada grupo amino. Del total de organoaminas usadas, el 90 % tiene al menos un grupo amino cuaternario, siendo poco común que tengan un grupo primario o secundario. Esta observación se corresponde con lo esperable puesto que en la reacción de síntesis se empleó un medio ácido (véase distribución del compuesto ácido en la figura 3.2). De allí que gran parte de los grupos amino se hayan encontrado en su forma catiónica protonada. Por otro lado, notemos que solo hay una especie que posee una amina aromática, así como también hay solo una con una amina terciaria.

Con respecto al aspecto estructural, también se pueden analizar los compuestos orgánicos usados en cada experimentación según si poseen ciclos y/o anillos (véase figura 3.8.b). En este sentido, se pueden distinguir en términos generales en aquellas que poseen anillos aromáticos y aquellas que no. Dentro de las que no lo tienen, se encuentran aquellas con estructuras cíclicas basadas en átomos de carbono y aquellas sin ciclos de ningún tipo. Por otro lado, aquellas con anillos aromáticos se pueden dividir en base a la composición del anillo en carbonoaromáticas o heteroaromáticos (anillos solamente de átomos de carbono o anillos con átomos de otros elementos).

En sí, de los 44 compuestos orgánicos empleados, se puede resaltar que un cuarto posee anillos aromáticos de algún tipo, un quinto posee ciclos alifáticos mientras que las restantes contienen cadenas alífáticas. Además, cabe mencionar que ninguno de los compuestos usados tiene tanto un anillo como un ciclo alifático.

Grupos Funcionales Amino

Cantidad de organoaminas según tipo de amina

Tipo	Cantidad
primaria	9
secundaria	6
terciaria	1
cuaternaria	40
aromática	1

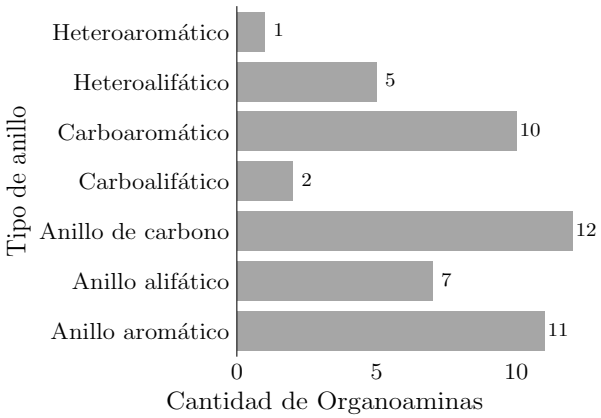

Fig. 3.8: Cantidad de organoaminas según (a) tipo de amina y (b) tipo de anillo presentes, de un total de 44 organoaminas totales. Cada organoamina puede tener más de un grupo amino o de un anillo posible.

Por último, para poder indagar posibles subgrupos de propiedades físico-químicas, se estudiaron las correlaciones existentes entre parte de las 61 *features* curadas del grupo `FEAT_PHYCHEM`. Para ello, se consideraron únicamente las variables ordinales y continuas, aplicándoles el método de Spearman para obtener su matriz de correlación. Se optó por esta técnica dado que no supone distribución alguna de las variables, así como también permite su aplicación sobre variables ordinales al buscar relaciones de tipo monotónicas.

En la figura 3.9 se encuentran la matriz de correlación reordenada para poder agrupar aquellas correlaciones similares entre sí. Para este agrupamiento se empleó una clusterización jerárquica de los vectores de correlación (véase clustering jerárquico). Se usó el promedio de los elementos de cada cluster como metodología de *enlace* y la distancia euclidiana entre los vectores para calcular cercanías (no se apreciaron diferencias en cuanto a los clústeres formados usando otros métodos de distancia).

Como se puede observar en la figura 3.9, se identifican cuatro grupos principales de propiedades físico-químicas en base a su correlación entre sí.

- **Grupo 1:** consiste en descriptores de la capacidad aceptora de hidrógenos de la organoamina, el cual incluye descriptores como la cantidad de aminas secundarias y contadores de sitios aceptores de enlaces de hidrógeno, entre otros.

- **Grupo 2:** se trata de variables que aluden a la capacidad donora de la organoamina, teniendo descriptores que cuentan la cantidad de átomos de nitrógeno, la cantidad de aminas cuaternarias e incluso otras como el área superficial polar (además de aquellas como `HdonorCount` y `donorCount`).

- **Grupo 3:** consiste de solo tres descriptores que contabilizan el tamaño de anillos y ciclos. En sí presentan una similaridad con respecto al Grupo 4, el de mayor tamaño, aunque se diferencia rotundamente por tener una correlación negativa considerable con respecto a `ChainAtomCount` y levemente negativa con un subgrupo de propiedades del Grupo 4.

- **Grupo 4:** engloba aproximadamente el 50 % de las variables ordinales y continuas, siendo un bloque con fuerte correlación positiva entre sí y levemente negativa con

respecto a todas las demás (a excepción del Grupo 3). Químicamente, se trata de un grupo diverso de descriptores que incluye tanto propiedades vinculadas a áreas superficiales (de Van der Waals, no polares, de carga positiva/negativa), como al largo de las cadenas alifáticas, cantidad de enlaces, polaridad, refractividad, entre otros. Si consideramos el tipo de variable, se puede notar que el Grupo 4 agrupa gran parte de las variables continuas, habiendo solamente dos variables continuas que no quedaron clusterizadas en este grupo: `PolarSurfaceArea` y `ASAP` (área polar superficial accesible por aguas). Notemos que las variables ordinales de este grupo también se tratan de características superficiales de la organomina (por ejemplo, el largo de la cadena carbonada influye en el área de contacto).

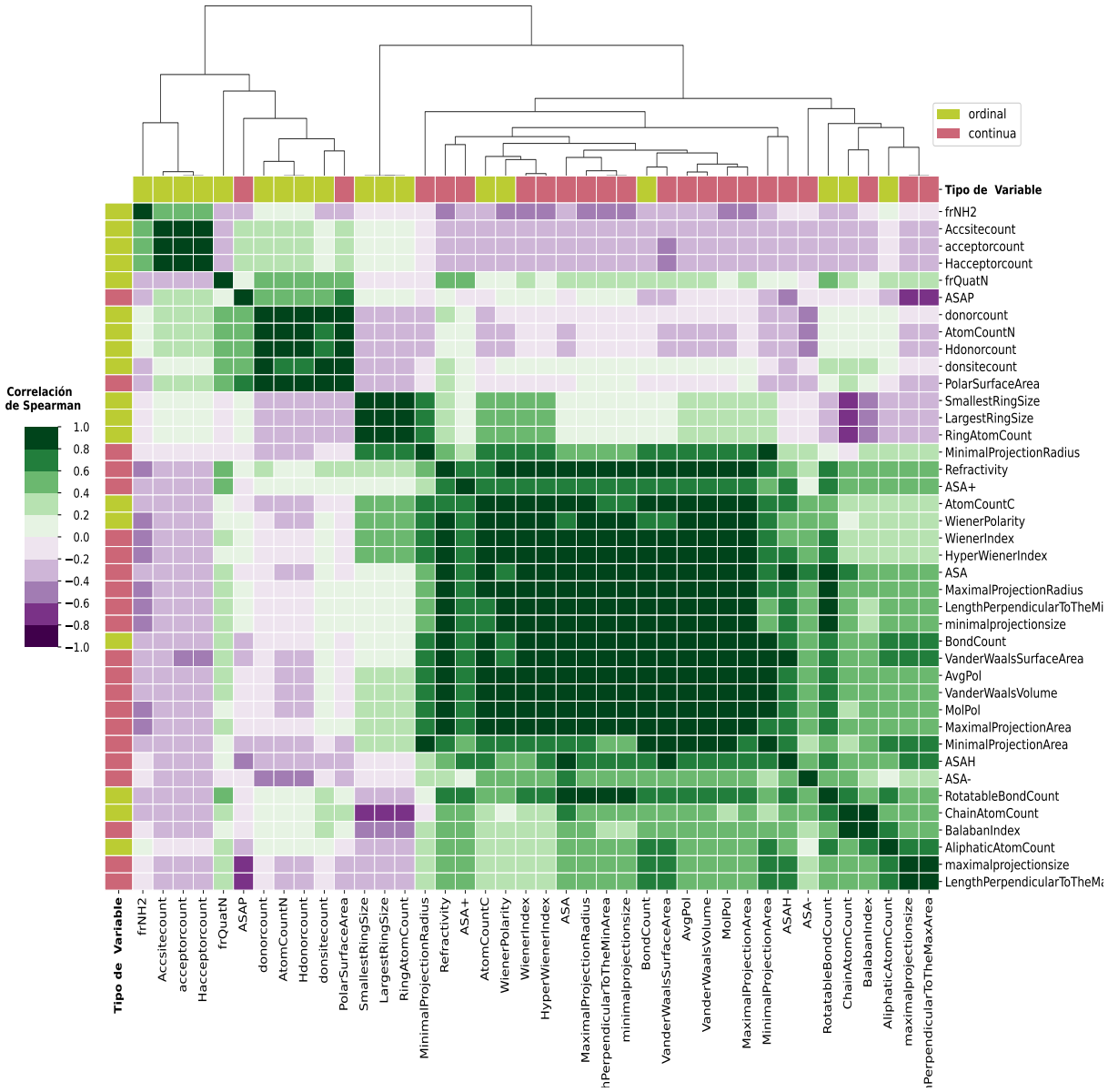

Fig. 3.9: Correlación de Spearman entre las *features* ordinales y continuas descriptoras de la organoamina empleada en la síntesis de perovskita en el *dataset* RAPID [1].

De esta forma, se pueden identificar cuatro posibles grupos de propiedades físico-químicas que aluden al mismo tipo de información de la organoamina. Analizar su correlación es el punta pié inicial para luego emplear técnicas de clusterización más robustas para poder, ya sea seleccionar aquellas propiedades que sean relevantes en sí para los modelos de aprendizaje, así como también considerar una reducción de dimensionalidad para poder disminuir el posible ruido.

Finalmente, habiendo descripto el *dataset* de trabajo, procedemos a analizar la fuente de errores que pueden tener los modelos de predicción de cristalización de perovskita, así como también estudiamos qué métricas se adaptan mejor para este problema.

4. EVALUACIÓN Y ERRORES

A la hora de escoger entre modelos basados en aprendizaje automático, es muy importante definir qué métricas se usan en la evaluación de su *performance*, y a qué responde su variabilidad en caso de existir. En este capítulo se realiza un análisis de las fuentes posibles de error que pueden tener los modelos de predictivos de la cristalización de perovskita. Además, se realiza una simulación para entender cómo las características de los datos utilizados en esta tesis -desbalance de clases y cantidad reducida de datos-, pueden impactar en las métricas de clasificación comúnmente usadas.

4.1. Contexto de aplicación y evaluación

Si bien en términos generales los modelos se utilizan para predecir, entender cómo se utilizarán los modelos en su aplicación es clave para determinar cómo evaluar los sistemas de aprendizaje automático. En este trabajo en particular, la tarea de predicción consiste la *clasificación binaria de las condiciones experimentales de una síntesis química.*

En primer lugar, consideremos que la síntesis de perovskita en forma de monocristales es un proceso químico complejo que *difícilmente* ocurre. Dicho en otras palabras, son muy limitadas las condiciones en donde realmente se produce la cristalización, en comparación con el amplio espectro posible de experimentaciones en el que no cristaliza. Esta consideración tiene impacto en el desarrollo de los modelos, dado que para un grupo de investigación en nanomateriales no será lo mismo etiquetar correctamente que la síntesis genera monocristales, que etiquetar que no lo hace. Al saber que es muy probable que no cristalice, que un modelo prediga la etiqueta positiva correctamente tiene mucho más valor informativo que uno que prediga la etiqueta negativa adecuadamente. El aporte de esta clase de modelos es más relevante cuando se puede predecir la efectiva cristalización, ya que por defecto se espera que no suceda.

De esta manera, entendemos que en el marco de la ciencia de los materiales -contexto de aplicación de estos modelos-, es de especial interés poder *capturar* la mayor proporción posible de condiciones experimentales donde efectivamente cristaliza. Por lo que se trabaja a costas de cometer un error de tipo 1 (afirmar que cristaliza frente a condiciones reales cuando no lo hace) con el fin de priorizar que el modelo *aprenda* de la acotada cantidad de datos positivos, minimizando el error de tipo 2 (afirmar que no cristaliza cuando verdaderamente lo hace).

4.2. Fuentes de error

Al usar métricas para comparar evaluaciones de distintos modelos entre sí, se busca seleccionar aquel que obtenga la mejor *performance* según lo indiquen las métricas seleccionadas. Al estar usando aquellas basadas en una matriz de confusión, se está buscando minimizar las confusiones del modelo, es decir, minimizar el error del etiquetado. Ahora bien, para entender cuáles son los errores que comete un modelo es necesario definir qué se entiende por *modelo*. En este trabajo se considera que un *modelo* de aprendizaje automático $\mathscr{M}$ viene dado por un conjunto de datos de entrenamiento y un *pipeline* de procesamiento. Luego, estos modelos se evalúan en distintos conjuntos mediante métricas

que describen distintos aspectos de la calidad de las predicciones. Al referirnos a los errores de un modelo, nos referimos a aquellos debidos a las fluctuaciones que puedan tener los datos usados para el entrenamiento, o bien a aquellos debidos al *pipeline* de procesamiento en sí.

A continuación se realiza un análisis de las distintas fuentes de error que puede tener un modelo en pos de comprender a qué se debe la variabilidad del comportamiento de las métricas en la evaluación. El propósito consiste en entender hasta qué punto mejorar la *performance* de una métrica consiste en el desarrollo de un mejor *pipeline* (ya sea con otras técnicas de preprocesamiento, algoritmos de clasificación, entre otros), o bien se encuentra limitado por las características de la naturaleza del conjunto de datos utilizado (error de difícil tratamiento cuando se tiene poca cantidad de datos).

4.2.1. Error en la recolección de datos

Dado que los datos que empleamos provienen de resultados experimentales obtenidos en el laboratorio, se tiene una inherente fuente de error dada por el dispositivo experimental que los produjo. Si bien es cierto que para estos datos se empleó un mecanismo automatizado por robótica que ayuda a minimizar los errores, no se descarta un posible sesgo sistemático o una inapropiada configuración del dispositivo, así como también una inadecuada manipulación humana[1].

Por otro lado, otra posible fuente de error proviene del diseño experimental en sí, pudiendo estar plasmado en la recolección o muestreo de los datos. Tengamos en cuenta que la distribución de la cristalización depende de la concentración de los reactivos de síntesis que se usen, entre ellos, la organoamina. Para cada una, posiblemente, se tiene un espacio de concentraciones distinto en donde efectivamente cristaliza la perovskita. De allí la importancia de analizar cuán bien se representa dicho espacio con la muestra tomada. Por ejemplo, podría ocurrir que para una organoamina que *favorece* muy poco la cristalización, justo se haya hecho un muestreo en un subespacio en particular donde sí cristalizó. Esto podría llevar a que un modelo entrenado en este conjunto de datos *aprenda* que esta clase de organoaminas es propensa a favorecer la cristalización, llevando a conclusiones potencialmente erróneas. De forma análoga, podría darse el caso contrario donde en base a los datos disponibles una organoamina pareciese desfavorecer la cristalización, aunque en realidad se trate de un sesgo de los datos en su etapa de recolección.

Todas estas fuentes de error, vinculadas a la obtención y recolección de los datos crudos, afectan al modelo por el agregado de ruido que producen, dificultando la tarea de los algoritmos a la hora de detectar patrones o realizar generalizaciones. Sin ir más lejos, también limitan las aplicaciones y los contextos en donde es pertinente usar sistemas predictivos basados en *datasets* como el usado en esta tesis.

4.2.2. Error en el *pipeline* de desarrollo

Al desarrollar modelos de aprendizaje automático, no solo son relevantes las características del conjunto de datos a utilizar, sino también lo es todo el procesamiento a aplicar sobre los datos hasta producirse el conjunto de entrenamiento en sí. A todas estas etapas en su conjunto las denominamos *pipeline* de los datos, el cual trae consigo su propio error que puede verse reflejado en los resultados finales. Por ejemplo, al usar técnicas

[1] Sin ir más lejos, se pudo haber dado una contaminación de los reactivos, un error en el control de las variables de las condiciones experimentales como la temperatura y humedad, entre otros.

de muestreo que producen datos sintéticos puede que se generen datos que químicamente sean lejanos a la realidad.

Por otro lado, otro aspecto a considerar es qué tan bien se puede generalizar la distinción entre clases usando alguna técnica en particular. Esto está relacionado con qué distribución toman los datos y qué técnicas de *decisión* usen los algoritmos de clasificación (por ejemplo, al usarse planos de corte en SVM (*Support Vector Machines*), se supone que existen planos lo suficientemente buenos usando los atributos disponibles como para diferenciar las clases).

En general, al desarrollar modelos de aprendizaje automático se busca el *pipeline* de trabajo que genere los mejores resultados para un mismo conjunto de datos. De esta forma, si se comparan los resultados de dos modelos, se espera que la diferencia se deba al procesamiento hecho (es decir, al *pipeline* de trabajo empleado) y no a las fluctuaciones que puedan tener los datos empleados. En otras palabras, se desea que la variabilidad de los modelos estudiados se deba a factores controlables en la experimentación.

En particular, dado que este trabajo modela el problema de la predicción de la cristalización como una clasificación, la evaluación de los modelos decidimos realizarla mediante métricas computadas a partir de la matriz de confusión. De allí que, para poder entender qué tan variables pueden ser los resultados de esta evaluación, se haya modelado el impacto de los errores como ruido en las matrices de confusión obtenidas al evaluar los modelos.

4.3. Simulación del error como ruido en la matriz de confusión

Con el objetivo de comprender cómo los errores de un modelo afectan su *perfomance*, se realizó una simulación de la variabilidad que pueden sufrir las matrices de confusión producto de ambos tipos de errores.

A continuación se detalla cómo se modelaron estos errores en cambios concretos en las matrices de confusión, para así posteriormente poder realizar un estudio de qué métricas se adecúan mejor a las características del *dataset* de acuerdo a lo que se desea predecir. De esta forma podremos definir mejor qué métricas usar, así como entender mejor la significancia de los resultados de la evaluación de los modelos.

4.3.1. Modelado de errores

En primer lugar, consideremos un modelo de clasificación de condiciones de síntesis de perovskita llamado $\mathcal{M}$, el cual es entrenado en un conjunto de datos y con un *pipeline* en particular. Dado un conjunto de datos de evaluación llamado D, denominamos a la matriz de confusión resultante de evaluar $\mathcal{M}$ en D como $CM(\mathcal{M}, D)$. Esta matriz puede mostrar ciertas confusiones de etiquetado debidas a cómo funciona el modelo de clasificación, así como también puede reflejar cierto error proveniente de la recolección de los datos.

Para visualizar cómo proponemos perturbar la matriz de confusión al modelar ambos tipos de errores, consideremos la siguiente matriz a modo de ejemplo.

$$CM(\mathcal{M}, \mathcal{D}) \equiv \begin{bmatrix} 97_{NN} & 7_{NP} \\ 6_{PN} & 21_{PP} \end{bmatrix}$$

En este ejemplo se considera que el conjunto de evaluación tiene un total de 131 instancias, de las cuales 97 son verdaderos negativos (NN), 21 son verdaderos positivos (PP), 7 son falsos positivos (NP) y 6 son falsos negativos (PN).

Ahora bien supongamos que se repite la recolección de los datos de evaluación $\mathcal{D}$ pero que esta vez se produce cierto ruido al generarlos, afectando las etiquetas de la clase objetivo. Por ejemplo, consideremos dos instancias que eran etiquetadas como positivas para cristalización, que ahora pasan a ser etiquetadas como negativas. A este nuevo conjunto de datos de evaluación lo denominamos $\mathcal{D}_{ruidoso}$ puesto que tienen un ruido agregado con respecto al conjunto original. Notemos que la predicción del modelo al evaluarlo en este nuevo conjunto no se vería alterada dado que se sigue considerando el mismo modelo $\mathcal{M}$ con su *pipeline* de entrenamiento original. De modo que la matriz de confusión perturbada tendría un error agregado proveniente de los datos de evaluación, donde se da un cambio de etiquetas de verdadero positivo (PP) a falso positivo (NP) en el eje de predicción (columnas):

$$CM(\mathcal{M}, \mathcal{D}_{ruidoso}) \equiv \begin{bmatrix} 97_{NN} & 9_{NP} \\ 6_{PN} & 19_{PP} \end{bmatrix}$$

Por otro lado, para simular el error agregado del modelo consideramos cambios en las etiquetas predichas, es decir, que se trabaja con un modelo ruidoso $\mathcal{M}_{ruidoso}$ que comete mayor cantidad de *confusiones*. Siguiendo con el ejemplo anterior, supongamos que en $\mathcal{D}$ una instancia de etiqueta positiva en el eje actual fuese etiquetada por $\mathcal{M}_{ruidoso}$ como negativa. Luego, veríamos un cambio de una unidad de verdadero positivo (PP) a falso negativo (PN) en el eje actual (fila):

$$CM(\mathcal{M}_{ruidoso}, \mathcal{D}) \equiv \begin{bmatrix} 97_{NN} & 7_{NP} \\ 7_{PN} & 20_{PP} \end{bmatrix}$$

De esta forma, se propone estudiar el ruido proveniente de los datos y del modelo en conjunto para analizar su efecto en ciertas matrices de confusión de interés, y así, en las métricas que se calculan en base a la matriz de confusión. Este modelado consiste en modificaciones hechas sobre una matriz de confusión inicial mediante la aplicación de sucesivos cambios de etiqueta entre clases (tanto en el eje de predicción para un error del modelo como en el eje actual para un error proveniente de los datos de evaluación). De este modo, analizando las matrices con ruido resultantes podremos estudiar la variación de las métricas para así entender la relevancia de los resultados. Cabe mencionar que con este modelado de errores, se trabaja bajo la hipótesis de que no hay interacción ni acoplamiento entre las fuentes de error consideradas.

Si bien este análisis de errores presenta ciertas limitaciones[2], como el objetivo del capítulo es adquirir una intuición de cómo pueden llegar a modificarse las métricas debido a fluctuaciones en las matrices de confusión, consideramos apropiado hacer este análisis incluso considerando estas limitaciones. Recordemos que la motivación de esta simulación del ruido tiene como objetivo poder definir mejor las métricas a utilizar para evaluar modelos desarrollados en base de un *dataset* acotado en tamaño, sesgado y con gran desbalance de la variable a predecir.

[2] Notemos que de esta forma los errores provenientes de los datos de entrenamiento están contemplados en $\mathcal{M}_{ruidoso}$, junto a los del *pipeline* de trabajo. Para un estudio en mayor profundidad sería pertinente realizar conjuntos de datos sintéticos en vez de matrices de confusión sintéticas.

4.3.2. Matrices de confusión con ruido

A continuación se presenta la metodología empleada para generar los conjuntos de matrices de confusión con *ruido* a ser analizadas. Luego, se realizan simulaciones de las posibles matrices de confusión que se pueden obtener en este trabajo considerando el tamaño y desbalance del *dataset* usado. En particular, nos enfocamos en estudiar matrices de confusión que reflejen lo que se espera como resultado de un modelo con «*alta*», «*regular*» o «*baja*» *performance*, cualidades definidas para el marco de esta tesis.

De esta forma podemos estudiar si es posible que un modelo sea considerado con cierta *performance* errónea, producto de trabajar con un conjunto de datos con *ruido* donde la acotada cantidad de datos y el desbalance de clases dificultad la apropiada valoración del modelo.

4.3.3. Metodología

En primer lugar para poder estudiar el efecto que tienen los errores simulados en las matrices de confusión, se definieron ciertos casos de referencia que son de particular interés en este trabajo. Una vez consideradas estas matrices de confusión, por cada una se generaron múltiples variantes con distinto ruido simulado, produciendo así un conjunto de matrices con ruido por cada caso de interés. Habiendo generado dichos conjuntos, se describió cada matriz mediante las métricas de clasificación tradicionalmente usadas (*recall*, *precision* y *F1*) y el coeficiente de *Matthew*, métrica de referencia usada en el trabajo que dio origen al *dataset*.

El objetivo de esta simulación consiste en poder identificar si existen métricas que sean más robustas que otras con respecto al agregado del ruido simulado. Para ello consideramos que una métrica es robusta si mapea las matrices con ruido de una misma matriz de interés a un espacio cercano, manteniendo su lejanía con respecto a las otras matrices de interés consideradas.

Matrices de Confusión de Interés

Para definir cuáles son las matrices de confusión que son de nuestro interés para analizar, consideramos la definición presentada en la sección 2.5.2. En esta la matriz de confusión se expresa como una función $CM(\lambda_{N,N}, \lambda_{P,P}, m, \delta)$, en donde:

- $\lambda_{N,N}$ es la proporción de verdaderos negativos
- $\lambda_{P,P}$ es la proporción de verdaderos positivos
- m es el tamaño del conjunto de evaluación
- δ es el coeficiente de desbalanceo de clases

Notemos que tanto $\lambda_{N,N}$ como $\lambda_{P,P}$ son parámetros que aluden a la *performance* del modelo en sí, mientras que m y δ refieren al conjunto de datos de evaluación.

Por otro lado, para definir el conjunto de matrices que son de nuestro interés poder diferenciar, consideramos tres tipos de modelos en cuanto a su desempeño para clasificar correctamente las condiciones experimentales donde sí cristaliza la perovskita (esto se debe a que se busca que los modelos aprovechen las instancias donde cristaliza):

- `Alta` ($\lambda_{P,P=0,75}$): predice 75 % de las instancias positivas

- `Regular` ($\lambda_{P,P=0,5}$): predice 50 % de las instancias positivas
- `Baja` ($\lambda_{P,P=0,25}$): predice 25 % de las instancias positivas

Por su parte, otro aspecto tenido en cuenta fue el efecto del desbalance de clases de la variable objetivo. Para ello analizamos dos casos de desbalance de distinta magnitud en favor de la clase negativa: un nivel bajo con δ = -0.2 y un nivel alto con δ = -0.6. A priori se espera que para un desbalance alto las métricas muestren mayor sensibilidad a cambios de etiqueta, en comparación a lo esperado para un desbalance bajo. Cabe aclarar que no se espera que esto afecte poder determinar el tipo de matriz de origen de una matriz con ruido. Por ejemplo, si tenemos un modelo que predice las instancias positivas con una «*buena*» *performance*, se espera que esta no se deteriore con el ruido simulado que se agrega. Sin embargo, podría pasar que el desbalance intensifique el efecto del ruido agregado en las matrices de confusión simuladas, reflejándose en los valores de las métricas usadas.

A la hora de caracterizar a las matrices de estudio, también se consideró el tamaño del conjunto de evaluación a ser utilizado, variando su valor entre los siguientes posibles tamaños: 100, 600 y 1200 instancias de evaluación. Esta elección se debe a que estos valores se mueven entre los posibles tamaños a ser usados en nuestro trabajo: aproximadamente 100 es la cantidad de instancias por organoamina que se suele tener al considerar aquellas no tradicionales (véase sección 3.2), 600 instancias son aproximadamente el 10 % del tamaño *dataset* y 1200 es un valor mayor de referencia, tomado para analizar la tendencia en caso de llegar a poseer una mayor cantidad de datos disponibles. Recordemos que la motivación original del trabajo es explorar modelos de predicción de cristalización, que posteriormente pueda ser usado usar como oráculo para la cristalización de una organoamina nunca antes explorada en el entrenamiento del modelo.

Por último, cabe resaltar que se trabajó durante toda la simulación con $\lambda_{N,N}$ como un parámetro fijo, el cual hace referencia a qué tan bien los modelos identifican correctamente los verdaderos negativos, es decir, cuando no cristaliza efectivamente la perovskita. Esta decisión se basa en que para nosotros la clase negativa no tiene la misma importancia que la positiva, puesto que es esperable que, por lo general, no cristalice el producto de síntesis. Se eligió como valor arbitrario $\lambda_{N,N} = 0.8$ para toda la simulación, evitando así que su variación influya en los resultados.

Generación de matrices de confusión con ruido

Por cada matriz de confusión de interés, se produjeron múltiples variantes de matrices con ruido, simulando el efecto de los errores provenientes de los datos de evaluación, del modelo o de ambos a la vez. Por simplificación, al primero se lo denomina $Noise_{data}$ y al segundo $Noise_{model}$. En base al modelado propuesto en la sección 4.3.1, por cada matriz de confusión inicial estudiada se produjo una proporción determinada de sucesivos cambios de etiqueta efectuados al azar. Para describir estos cambios, consideramos el eje de las columnas de una matriz de confusión como el «eje de predicción», y el eje de las filas como el «eje actual». De este modo, para simular el error experimental se hicieron cambios de etiqueta en el eje actual, mientras que para el error del modelo se usó el eje de predicción (o en ambos para simular un efecto completo). De esta forma, se estudió el efecto de cada tipo de error por separado y además la combinación de ambos.

A continuación se detalla la implementación de `Simular-Ruido`, procedimiento empleado para la simulación del ruido en las matrices de confusión estudiadas. Este recibe entre

sus parámetros a la cantidad de matrices con ruido a generar por cada matriz de interés y la proporción de instancias cuya etiqueta final queda librada a un posible cambio. A estos parámetros los denominamos m y *prop_ruido* respectivamente. `Simular-Ruido` también toma como parámetros las características necesarias para definir cuáles son las matrices iniciales de interés a perturbar ($CM_{args} = \{\lambda_{N,N}, \lambda_{P,P}, n, \delta\}$). Además, en *Noise_type* se le indica si se quiere generar $Noise_{data}$, $Noise_{model}$ o ambos.

`Simular-Ruido` genera todas las matrices a ser perturbadas por el procedimiento `Perturbar-Matriz`. Este, dada una matriz de confusión, la altera mediante la aplicación sucesiva de una operación denominada `Cambiar-Etiqueta`, realizándose la modificación en uno o ambos ejes según si se combina tanto el ruido experimental como el ruido del modelo. Esta última operación se encarga de modificar la matriz de confusión reflejando el cambio de etiqueta de una instancia por vez, en una clase (positiva o negativa) y sentido elegidos al azar (ver en la figura 4.1 el esquema con los cambios de etiqueta posibles para la clase positiva a modo de ejemplo). En términos generales, para simular el ruido, se aplican sucesivos cambios de etiqueta en matrices de confusión de interés. El procedimiento que modifica las etiquetas es no determinístico, sujeto a la semilla de azar empleada.

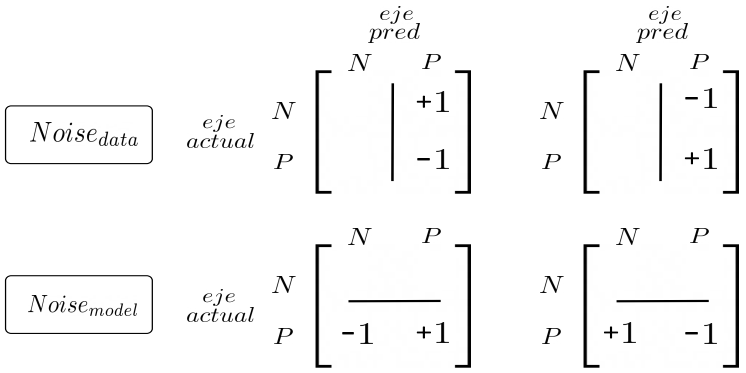

Fig. 4.1: Esquema de cómo la operación `Cambiar-Etiqueta` modifica una matriz de confusión, en particular para una instancia de clase positiva (P), según el ruido simulado: en los datos ($Noise_{model}$) o en el modelo ($Noise_{model}$). El ruido de los datos implica el cambio de la etiqueta real de una instancia falsa positiva a verdadera positiva ($NP \rightarrow PP$), o en el sentido contrario ($PP \rightarrow NP$). De forma análoga se modela el ruido en los datos, modificándose la etiqueta predicha de una instancia falso negativo a verdadero positivo, o viceversa ($PN \rightarrow PP$ o $PP \rightarrow PN$).

```
Simular-Ruido (Noise_type, CM_args, prop_ruido, m):
    CM = Generar matrices de interés definidas por CM_args
    S = Inicializar conjunto de matrices a perturbar
    Por cada cm ∈ CM, repetir m veces:
        cm_con_ruido = Perturbar-Matriz(cm, Noise_type, prop_ruido)
        Agregar(cm_con_ruido,S)

Perturbar-Matriz(cm, Noise_type, prop_ruido):
    n_op = Determinar cantidad de operaciones de Cambiar-Etiqueta aplicar
    según cm y prop_ruido
    Si el tipo de error es unico, aplicar n_op veces la operación
```

```
    Cambiar-Etiqueta(Eje-Asociado(Noise_type), cm)
    De lo contrario, simular un ruido mixto mediante Cambiar-Etiqueta
    en ambos ejes

Cambiar-Etiqueta(eje_de_cambio, m)
    Elegir aleatoriamente una etiqueta en el eje_de_cambio (eje de
    predicción o eje actual)
    Elegir aleatoriamente un sentido s para disminuir/aumentar en una
    unidad: positivo a negativo o viceversa)
        De ser posible, efectuar cambio de etiqueta en ese sentido s
        De no ser posible intentar en el sentido inverso
        En caso de que tampoco sea posible, volver a intentar la operación
        desde un inicio
```

Pseudocódigo 4.1: Procedimientos implementados para generar matrices de confusión con *ruido*, simulando el error provisto por los datos o por el modelo en sí mismo. El método `Simular-Ruido` genera matrices de confusión de interés en función de un conjunto de parámetros como el desbalance de clases, el tamaño del conjunto de evaluación y la proporción de instancias verdaderas positivas predichas correctamente. Luego, perturba cada matriz de confusión, generando un conjunto de matrices con ruido. `Perturbar-Matriz` se encarga de modificar cada matriz en base al ruido a simular (de los datos o del modelo) y la proporción de ruido aplicar. Para ello, aplica el método `Cambiar-Etiqueta` sucesivas veces modificando el valor de etiqueta de una instancia por vez aleatoriamente, siendo esta perturbación reflejo del posible error que pueda tener el sistema predictivo evaluado.

4.3.4. Efecto de la escasez y el desbalance de datos

Para realizar la simulación del ruido en las matrices de confusión, se siguió la metodología previamente detallada. Se generaron las matrices de confusión de interés definidas por la fórmula $CM(\lambda_{N,N}, \lambda_{P,P}, m, \delta)$, usando los valores de sus parámetros descriptos anteriormente (véase sección 2.5.2). En la tabla 4.1 se resumen las características de las matrices de confusión a estudiar, las categorías consideradas y los valores correspondientes a sus parámetros.

Cabe resaltar la importancia del parámetro $\lambda_{P,P}$ el cual representa qué porción de la cantidad de instancias positivas son predichas correctamente, es decir, el *recall* de la clase positiva. En base a este parámetro, a cada matriz se la clasifica como resultante de la evaluación de un modelo con *performance* «*baja*», «*regular*» o «*alta*». Si bien esta métrica podría usarse como mecanismo de evaluación final, resulta muy sensible a la cantidad de instancias negativas y no considera el tamaño completo del conjunto de evaluación, de allí que se busque una métrica que considere ambas clases en conjunto.

En esta simulación se busca comprender qué tan factible es categorizar una matriz con ruido con *performance* $\lambda_{P,P=i}$, como proveniente del grupo $\lambda_{P,P=j}$, con $i \neq j$ siendo su rango posible $[0, 1]$. En otras palabras, se quiere ver qué tan factible es poder «confundir» la *performance* de un modelo debido al ruido generado en la simulación, analizado la influencia del desbalance y tamaño del *dataset* de evaluación. Como métricas de estudio se consideraron al *recall*, *F1* y *precision* en sus versiones *macro* y *weighted*, además se tuvo en cuenta al *coeficiente de Matthew* al ser la métrica de referencia (véase en la sección

Parámetro	Categoría	Valor
Desbalance (δ)	Bajo	-0.2
	Alto	-0.6
Tamaño del conjunto de evaluación (m)	Muy chico	100
	Chico	600
	Estándar	1200
Performance del modelo ($\lambda_{P,P}$)	Baja	0.25
	Regular	0.50
	Alta	0.75

Tab. 4.1: Parámetros usados para generar las matrices de confusión según la definición $CM(\lambda_{N,N}, \lambda_{P,P}, m, \delta)$. δ indica el coeficiente de desbalance (con 0 como balance perfecto, -1 y 1 como desbalances máximos hacia las clases negativa y positiva respectivamente), m es la cantidad de instancias del conjunto de evaluación, $\lambda_{P,P}$ es la proporción de instancias positivas predichas correctamente. Se mantuvo $\lambda_{N,N}$ como constante en un valor de 0,8.

2.5.3 las definiciones de las métricas usadas). Por último, cabe mencionar que se mantuvo constante la proporción de instancias negativas predichas correctamente dado que no es de nuestro interés la sensibilidad con respecto a la clase donde no cristaliza ($\lambda_{N,N} = 0{,}8$). Por cada matriz de interés se produjeron $m = 250$ matrices con ruido diferentes para obtener un resultado más representativo[3]. Se fijó como proporción de ruido al 15 % de las instancias, siendo esta la cantidad que queda librada al efecto de la simulación. En particular, en esta sección mostraremos solamente el efecto para $Noise_{data}$ dado que los resultados obtenidos para $Noise_{model}$ mostraron la misma tendencia (los resultados para este último se pueden consultar en la sección 6.4 del Apéndice).

Para analizar los resultados de la simulación, se estudió cómo variaban las distribuciones de los valores de cada métrica para todas las matrices sintéticas con ruido consideradas. En caso de que una métrica no fuese alterada por el desbalance ni cantidad de datos, se debería poder ver la misma distribución en las distintas condiciones de evaluación. Además, si las métricas permitiesen diferenciar cada uno de los grupos de *performance* considerados, se esperaría que no hubiese superposición alguna entre las distribuciones.

[3] Se estudió a partir de qué cantidad de matrices con ruido el comportamiento general convergía, llegándose a este valor como referencia.

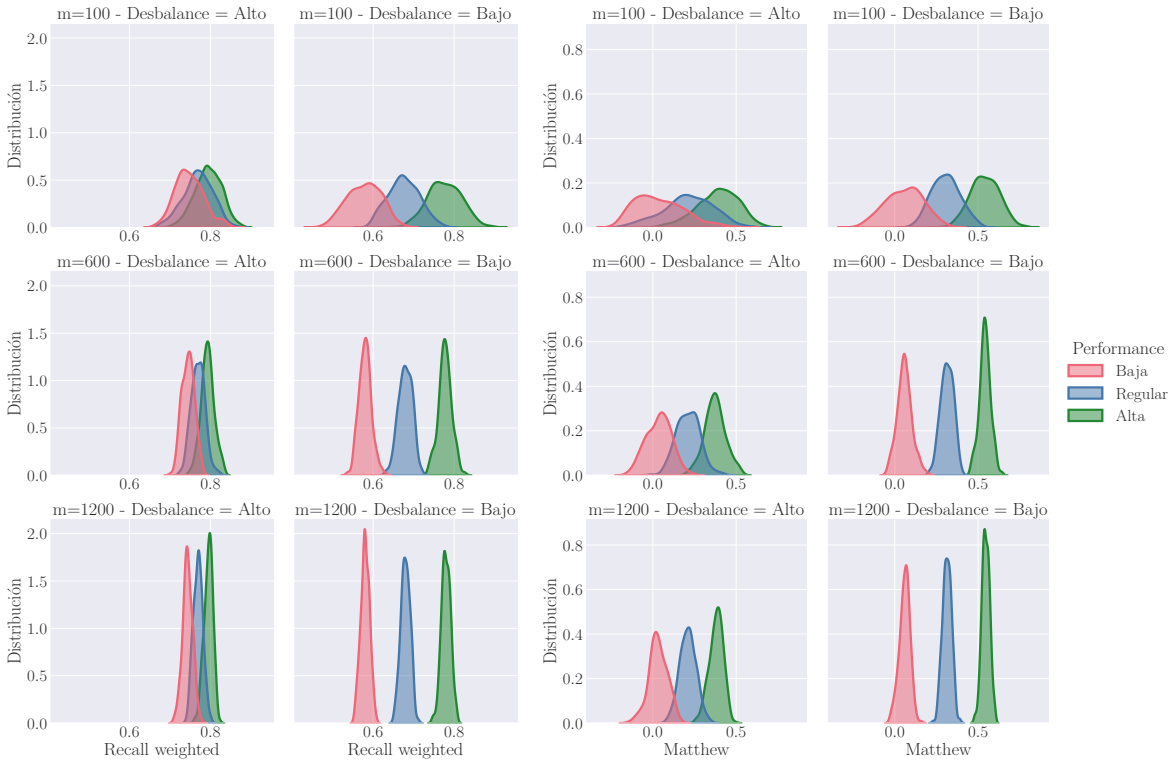

Fig. 4.2: Distribución de las métricas *recall weigthed* (primeras dos columnas) y *Matthew* (últimas dos columnas) para matrices de confusión con ruido de los datos simulados. Las matrices representan la evaluación sobre conjuntos de datos con distinto desbalance (δ), tamaño (m) y representando modelos con distinta *performance*.

Como resultado de la simulación se observó que todas las métricas resultaron sensibles al desbalance de clases, habiendo mucha más superposición entre las categorías de *performance* a desbalance alto que a nivel bajo. Esta superposición entre categorías indica que se han producido matrices con ruido de desempeño similar, a pesar de provenir de matrices de confusión iniciales categorizadas con diferente *performance*.

En la figura 4.2 solo se muestran las distribuciones de dos métricas, *recall weighted* y *Matthew*, a modo de ejemplo. En el Apéndice en la sección C se pueden observar las pruebas realizadas con otras métricas. En los resultados obtenidos se observó que a desbalance alto de clases, las métricas balanceadas por clase (de tipo *weighted*) no pudieron discriminar las matrices de confusión según la categoría de *performance* que representaban, independientemente del tamaño del conjunto de evaluación (viéndose una gran superposición para los tres tamaños considerados). Esto se debe a que toman en cuenta la cantidad de instancias de cada clase para calcular el valor de la métrica, de modo que su variabilidad depende principalmente de la clase mayoritaria, en los casos de desbalance alto. En cambio para los casos de desbalance bajo se pueden diferenciar las tres distribuciones, habiendo una superposición pequeña cuando se tiene un *dataset* reducido, de aproximadamente 100 instancias. En contraposición, las otras métricas como *Matthew* mostraron mayor sensibilidad en la clase negativa minoritaria, pudiéndose identificar los grupos de *performance* incluso con un tamaño chico del conjunto de evaluación ($m = 600$). En general, se puede notar que cuando las clases a predecir se encuentran bien balanceadas, la fluctuación dada por el tamaño del *dataset* de evaluación no imposibilita poder categorizar correctamente cada matriz de confusión (habiendo una cierta superposición a $m = 100$ que no impide

diferenciar las categorías analizadas en esta simulación).

Dado que en esta tesis se trabaja con un desbalance alto y un conjunto de evaluación reducido, se optó por explorar principalmente las métricas con resultados similares al coeficiente de *Matthew*.

4.3.5. Métricas complementarias

En el estudio anterior, se pudo observar que considerar una única métrica no es suficiente para poder diferenciar las categorías de *performance* de las matrices de confusión. De allí el interés de poder estudiar cómo se pueden complementar un par de métricas, de modo que ambas en simultáneo permitan conocer a qué tipo de matriz originalmente pertenece cada matriz de confusión con ruido.

En particular, para este estudio se consideraron matrices de confusión similares a las esperables en el *dataset* usado para la clasificación de perovskita, con un desbalance alto de la clase objetivo ($\delta = -0{,}6$) y un conjunto de evaluación mediano ($m \sim 600$). Como métricas candidatas a combinar, se usaron aquellas para las cuales se obtuvieron resultados similares al coeficiente de *Matthew* en el estudio anterior: *precision*, *recall* y *F1* en sus versiones *macro* (véase sección C).

En términos téoricos, por las definiciones de estas métricas, se espera que la complementariedad entre *F1* con respecto a *precision* y *recall* no aporte demasiado dado que *F1* se calcula como el promedio armónico de ambas. En cambio, probablemente se aprecie que *recall* y *precision* se complementan bien dado que miden distintos aspectos del modelo (exhaustividad y exactitud respectivamente).

Por otro lado, en principio no es claro intuir cómo el coeficiente de *Matthew* se puede complementar con las tradicionales métricas de clasificación. Se espera que al combinar una métrica conocida por su robustez frente al desbalance de clases, con el resto comúmmente usadas, se pueda obtener una buena métrica conjunta que facilite identificar las tres categorías por separado. Recordemos que se entiende que dos métricas se complementan satisfactoriamente si en conjunto permiten distinguir las tres categorías de *performance* definidas anteriormente mejor de lo que cada una lo haría individualmente.

Para analizar la complementariedad de las métricas, se realizó una representación bidimensional conjunta en donde se espera poder identificar los tres grupos como clústeres diferenciados.

Al analizar los resultados de la simulación se apreció que todas las combinaciones de métricas mostraron cierto grado de complementariedad satisfactoria, a excepción del par *F1 macro-precision macro*. En la figura 4.3 se presentan tres pares de métricas a modo de ejemplo del tipo de resultado obtenido para todas las combinaciones estudiadas (véase los resultados completos en el Apéndice en la sección 6.4). En la figura se aprecia que la relación del par *F1 macro-precision macro* es lineal, lo cual nos lleva a pensar que el ruido producto de la simulación afecta de igual forma a ambas métricas. De allí que se entiende que su análisis conjunto no aporta más información del que se tiene a partir de cada métrica individualmente.

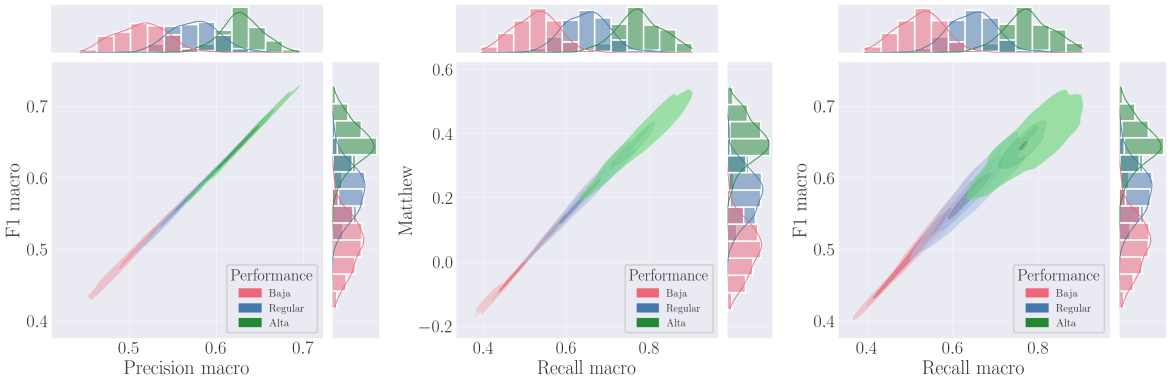

Fig. 4.3: Representación bidimensional de tres pares de métricas en orden creciente de complementariedad (de izquierda a derecha): *F1 macro-precision*, *Matthew-recall macro* y *F1 macro-recall macro*. La perturbación se debe al ruido simulado en los datos de evaluación ($Noise_{data}$).

En cambio, en los tres pares de métricas que involucran a *Matthew* se aprecia que el ruido simulado afecta de manera diferenciada a las métricas del par. En la figura 4.3 esto se encuentra representado por el par *Matthew-recall macro* (en el centro de la figura), en donde se observa una complementariedad que posibilita identificar cierta forma de clúster para la categoría de «*alta*». No obstante, no se aprecia que ninguna de las otras métricas tradicionales se destaque al complementarse con el coeficiente de Matthew.

Por otro lado, pares de métricas como *F1-recall macro* y *recall-precision* mostraron una mayor definición de las tres categorías apreciable visualmente en la representación bidimensional (representado por el gráfico derecho de la figura 4.3). Esto posibilita poder determinar para cada matriz de confusión con ruido cómo era la *performance* del modelo considerado en la matriz original.

Si bien se esperaba una baja complementariedad en el par *F1-precision* y una alta entre *recall-precision*, llama la atención que *F1-recall* sea uno de los pares que mayor aporte conjunto tiene. Se estima que esto se debe a que F1 termina siendo más influenciada por los efectos de la clase negativa mayoritaria dado el aporte de la precisión (consideremos que un mayor *recall* suele venir de la mano de una menor precisión en la predicción). En cambio en el *recall* se captura mejor la diferencia entre las matrices originales perturbadas. Entre los dos pares con mayor complementariedad visual, se entiende que es más informativa *F1-recall* dado que F1 contempla en su valor el aporte de la exactitud del modelo.

Para poder comprender si esta aparente complementariedad de métricas se debe a las características del *dataset* evaluado (tamaño y desbalance) o bien dependen del tipo de simulación empleado, se decidió repetir este estudio variando el tipo de ruido generado.

4.3.6. Ruido experimental vs ruido en datos

En las secciones anteriores se estudió la variabilidad que pueden tener las métricas basadas en la matriz de confusión debido al ruido simulado debido al error en la recolección de los datos ($Noise_{data}$). Con el fin de entender si el tipo de error influye en los resultados obtenidos, se procedió a repetir el mismo estudio mediante la representación bidimensional de los pares de métrica complementarias, pero esta vez simulando el ruido del modelo ($Noise_{model}$) en lugar del ruido de los datos. Luego, se repitió el mismo análisis considerando ambos ruidos en su conjunto (ver en la figura 4.1 el esquema de simulación de ambos errores).

Al estudiar la complementariedad de métricas en las matrices de confusión que son perturbadas con ruido proveniente del modelo, se apreció la misma tendencia de complementariedad obtenida en la sección anterior (al simular únicamente ruido de los datos). En la figura 4.4 se muestra parte de los resultados a modo de ejemplo, en donde se muestra solamente la complementariedad de los siguientes pares de métricas: *F1 macro-precision macro*, *Matthew-recall macro* y *F1 macro-recall macro* (de izquierda a derecha respectivamente). Los resultados completos se pueden consultar en la sección 6.5 del Apéndice.

En base a los resultados, se vio que nuevamente los pares *F1 macro-recall* y *precision-recall* se destacaron por ser aquellos con mayor complementariedad (representado por el gráfico de la derecha en la figura 4.4), pudiéndose identificar visualmente en la representación bidimensional las categorías de *performance* «*alta*» y «*regular*».

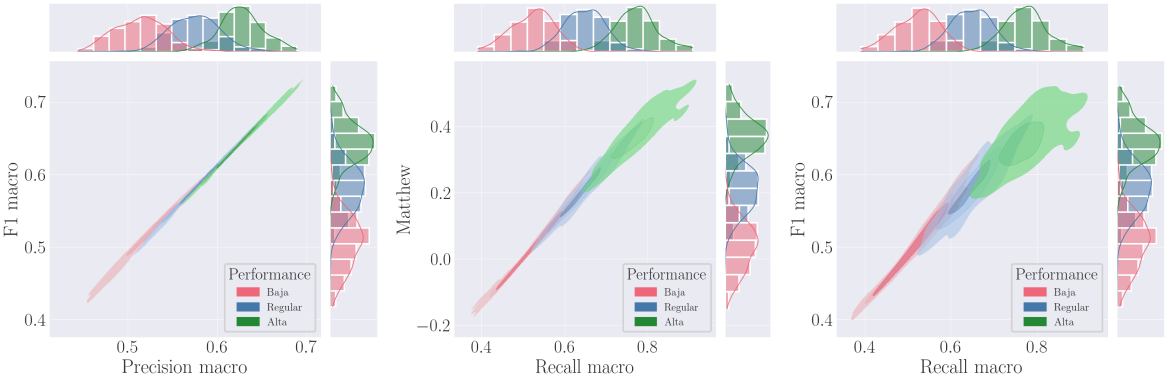

Fig. 4.4: Representación bidimensional de 3 pares de métricas en orden creciente de complementariedad (de izquierda a derecha): *F1 macro-precision*, *F1 macro-recall* y *Matthew-recall macro*. La perturbación se debe al ruido simulado en el modelo evaluado ($Noise_{model}$).

Por su lado, nuevamente ninguno de los pares que involucran al coeficiente de *Matthew* demuestran un comportamiento diferenciado que las destaque. Si bien se puede ver que para la categoría «*alta*» *performance*. el ruido simulado del modelo afecta ambas métricas de forma distinta, esto no hace que en su conjunto permitan identificar mejor las categorías de *performance* «*alta*» y «*regular*» de lo que cada métrica lo haría individualmente. Al estudiar la combinación de ambos ruidos, se obtuvieron resultados análogos a los obtenidos de cada uno individualmente. Por lo que se entiende que la complementariedad de las métricas estudiadas en este trabajo se deben a las características del *dataset* considerado (es decir, a la acotada cantidad de instancias y alto desbalance).

4.4. Conclusiones

En conclusión, en base a este análisis de la simulación de errores posibles en el desarrollo de los modelos, considerando las características del *dataset* a estudiar, se decidió utilizar para la evaluación de los modelos una representación bidimensional del par *F1-recall* dado que es uno de los pares con mayor complementariedad apreciable en la simulación realizada. Al analizar cómo varía cada métrica según las matrices de confusión que simulan posibles resultados de este trabajo, se apreció que considerar una única métrica puede llevar a una valoración incorrecta de un modelo, atribuyendo una *performance* no correspondida.

Además, se decidió emplear el coeficiente de *Matthew*, de manera independiente, como segunda métrica de referencia. Principalmente porque al analizar su distribución en la

simulación se vio que fue una de las métricas que mejor pudo distinguir las categorías de *performance* analizadas. También porque mostró cierta complementariedad intermedia similar para cualquier otra métrica considerada.

Si bien estas conclusiones se basan en la simulación propuesta para la generación de errores o ruido, a través de este estudio se puede entender mejor la importancia de la evaluación de modelos de clasificación, en particular al trabajar con *datasets* desbalanceados y acotados en tamaño. Consideramos que el trabajo realizado en este capítulo puede resultar de particular interés para abordar *datasets* similares, en especial si provienen de sistemas experimentales.

5. DESARROLLO Y SELECCIÓN DE MODELOS

En este capítulo se estudia cómo poder realizar mejoras al tipo de modelo de clasificación de cristales de perovskita planteado por Pendleton [15], considerando las características del *dataset* descriptas en el capítulo previo.

5.1. Influencia del solvente

En esta sección realizamos un estudio de la relevancia que tiene el solvente de reacción como variable predictora para los modelos de clasificación. Su interés proviene de que usar un solvente químicamente distinto implica un cambio importante en las concentraciones de los reactivos, así como también en las interacciones soluto-solvente. Recordemos que se busca poder encontrar en qué rango de concentración se ve favorecida la cristalización. La modificación del solvente podría producir que la distribución de la calidad del cristal con respecto a las concentraciones sea distinta. La identificación de estos posibles distintos patrones de cristalización podría verse dificultado si consideramos la acotada cantidad de datos experimentales disponibles, no representándose lo suficiente cada distribución por solvente.

En este sentido, desde un aspecto químico la síntesis de HOIPs realizada en diferentes solventes se puede considerar como un sistema de estudio distinto, dependiendo de qué tan diferentes sean los solventes entre sí y su influencia en la cristalización. En caso de que presenten propiedades químicas muy diferentes, realizar un modelo por solvente podría ser la opción más útil si se desea priorizar la especificidad y se tiene una cantidad de datos acotada. Sin embargo, si sus patrones de cristalización son muy diferenciados y se tiene suficiente cantidad de instancias por solvente para que los métodos de aprendizaje puedan identificarlos, un modelo multisolvente podría ser superador (pudiendo apreciar patrones escondidos de cristalización transversales al solvente usado).

En la literatura, hay trabajos que indican que la naturaleza del solvente puede afectar ciertas propiedades deseables de las HOIPs sintetizadas (por ejemplo, el factor de conversión energética solar-eléctrica, la compacidad y cristalinidad de los cristales, entre otros)[34, 35]. De allí la motivación de que los modelos puedan lidiar con dinámicas dadas por solventes diferentes, así como también entender qué casos merecen ser considerados utilizando modelos específicos de predicción y cuáles no.

En este trabajo se identificó que el conjunto de datos usados tiene datos provenientes de experimentaciones llevadas a cabo utilizando tres solventes: GBL (mayoritario), DMSO (minoritario) y DMF (minoritario). Si bien hay múltiples maneras en que las propiedades de cada solvente pueden afectar a la cristalización, se pueden destacar la evaporación, la solubilidad de los reactivos y la capacidad de coordinación con el plomo, halógeno y la amina orgánica, entre otros. Por ejemplo, si consideramos el punto de ebullición, se puede destacar el solvente DMF por su volatilidad característica que puede facilitar la cristalización no uniforme del producto de síntesis. En cambio, como DSMO y GBL presentan un punto de ebullición más alto, esta característica podría aportar estabilidad a la formación de cristales (o también podría dificultarla si este punto es demasiado elevado). La formación de los cristales requiere una interacción intermolecular adecuada y precisa, sensible al efecto dado por las partículas del solvente.

En este contexto, se realizaron las experimentaciones que se detallan a continuación para evaluar la incorporación o no del solvente como variable predictora de cristalización.

5.1.1. Experimentaciones

La primera experimentación consistió en evaluar si la identificación del solvente, como variable categórica, permite que los modelos puedan predecir mejor la cristalización.

Para ello, se consideró un primer modelo el cual fue entrenado en datos que no tenían información alguna del medio de reacción (es decir, se consideró como si existiese un único solvente universal). Dicho modelo se lo denominó `NO_SV`, aludiendo a que no tienen información del solvente[1].

En contraposición, se consideró un segundo modelo que incorpora la identidad del solvente mediante la técnica *one-hot-encoding* (véase sección 2.3.1), denominado `ENCODED_SV`. Notemos que de esta forma `NO_SV` supone que todos los datos de las experimentaciones provienen de un mismo sistema químico en estudio, mientras que el otro considera 3 sistemas diferenciables según el solvente usado (GBL, DMSO y DMF), en los cuales ocurre (o no) el mismo fenómeno de cristalización.

Para la generación de las muestras nos aseguramos de representar a todas las organoaminas en los datos de entrenamiento y evaluación, así como de mantener en las muestras el mismo desbalance de clases que la variable de cristalización tiene en el *dataset* original. Además, dichos conjuntos se prepararon de modo que contuviesen experimentaciones provenientes de los tres solventes, respetando sus proporciones originales en el *dataset*. Para la evaluación, tomamos una muestra del 20 % del *dataset* completo, la cual no fue utilizada en el entrenamiento.

Para comparar los resultados obtenidos, además de considerar las tradicionales métricas de clasificación en su versión *pesada* por clases (*precision*, *recall* y *F1*), se tuvo de referencia al coeficiente de Matthew (véase sección 2.5.3).

De este modo se pretende analizar si para el mismo conjunto de experimentaciones, desestimar la identidad del solvente afecta los resultados obtenidos para este *dataset* en particular, en caso de mantener la proporción original de los solventes. Notemos que no se están considerando sus propiedades químicas, dado que esta experimentación es una prueba de concepto de lo que posteriormente se podría considerar en una mayor descripción del solvente.

Se usaron dos algoritmos de clasificación diferentes (*Random Forest* y *Gradient Boosting*) para poder corroborar que los resultados no estén estrictamente sujetos a la técnica del clasificador. Para el entrenamiento de `NO_SV` y `ENCODED_SV`, se usaron los mismo parámetros, de modo que su diferencia recaiga únicamente en las variables predictoras usadas (en la sección E del apéndice se encuentran los hiperparámetros de entrenamiento usados para esta experimentación). Además, dado que se apreció cierta variabilidad de los resultados al usar un conjunto de entrenamiento distinto, se repitió la experimentación 50 veces usando una semilla de azar distinta para generar los conjuntos de entrenamiento-testeo en cada ejecución.

Como resultado, en caso de que el solvente aporte información, se espera que el modelo `ENCODED_SV` tenga un mejor *recall* que el modelo sin información del solvente, dado que conocer la distribución de la cristalización en tres solventes distintos podría facilitar la generalización de este fenómeno en entornos químicos diferentes.

[1] En química, el término «solvente» se suele abreviar como «sv».

En la figura 6.6 se aprecia la comparación entre los dos modelos entrenados con y sin datos del solvente como *feature*, `ENCODED_SV` y `NO_SV` respectivamente (usando GBC como clasificador). Se puede observar que no se aprecia una diferencia significativa entre considerar o no al solvente como parte de la descripción del entorno de síntesis. En efecto, incluso al evaluar empleando una métrica especializada para desbalance de clases como el coeficiente de Matthew (ver figura 6.6.b, no se obtiene una tendencia clara que favorezca considerar al solvente en modelos de predicción de la cristalización. Levemente se aprecia que el modelo `NO_SV` tiene una media mayor con respecto a `ENCODED_SV` en todas las métricas analizadas, aunque esta diferencia es muy chica como para representar una diferencia significativa (nótese la superposición de la mayor parte de los *boxplots*). Al analizar los resultados usando RF como clasificador, tampoco se apreciaron diferencias entre considerar o no la identidad del solvente, validando lo apreciado con el clasificador GBC (véase el Apéndice en la sección E.2 los resultados usando RF).

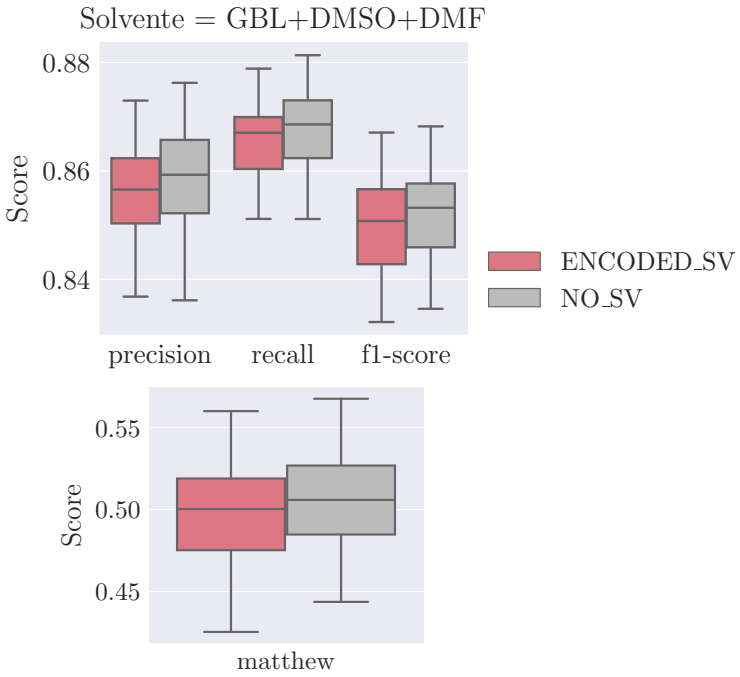

Fig. 5.1: Relevancia del solvente como predictor en la predicción de cristalización de HOIPs. Se compara un modelo con el solvente de reacción codificado (`ENCODED_SV`), versos un modelo sin información del solvente (`NO_SV`). Las métricas se encuentran en su versión pesada por clase, a excepción de Matthew. Se empleó GBC como técnica de clasificación y se usaron instancias de los 3 posibles solventes: GBL, DMSO y DMF.

La segunda experimentación consiste en realizar un modelo por solvente de manera específica y compararlo con el modelo multisolvente `ENCODED_SV`. De esta forma se trabaja suponiendo que son sistemas químicos independientes, en donde la cristalización se ve afectada lo suficiente como para necesitar estudiarlos por separado.

Para la evaluación se tomó una muestra por solvente sobre la cual se testearon tanto los modelos específicos por solvente (`MONO_SV`) como el modelo multisolvente `ENCODED_SV` (el cual es entrenado con datos de los tres solventes) para poder comparar sus resultados. Se consideraron solamente los solventes GBL y DMF para los modelos específicos, dado que la pequeña cantidad de datos y el desbalance de clases dificultaba realizar esta experimentación con los datos de DMSO (solo se tenían alrededor de 20 muestras positivas).

Al igual que la experimentación anterior, se repitió su estudio en 50 corridas diferentes usando una semilla de azar distinta para generar los conjuntos de entrenamiento-testeo.

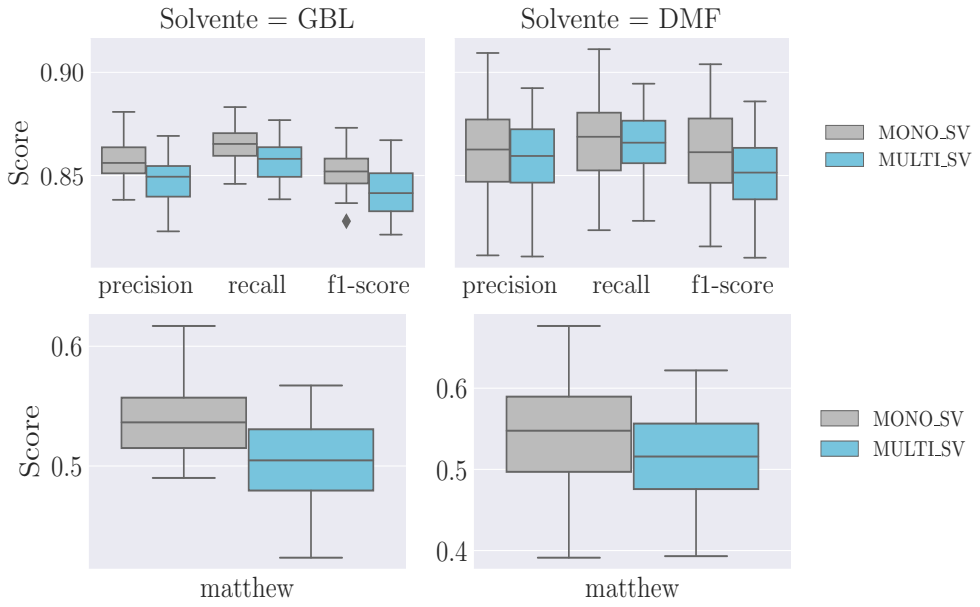

Fig. 5.2: Comparación entre modelos de predicción de cristalización de perovskita mono (`MONO_SV`) y multisolvente (`ENCODED_SV`), evaluados en experimentaciones que usan GBL (columna izquierda) o DMF (columna derecha) como solvente. El modelo `ENCODED_SV` fue entrenado con experimentaciones llevadas a cabo en GBL, DMF y DMSO, mientras que los modelos `MONO_SV` solo fueron entrenados en datos de un solvente (GBL o DMF). Se empleó GBC como método de clasificación.

En los resultados que se pueden apreciar en la figura 5.2, se observó una tendencia leve a favor de los modelos específicos monosolventes, principalmente en el solvente GBL al considerar el coeficiente de Matthew. En DMF por su lado no se apreciaron diferencias en los resultados, tanto al emplear un modelo específico como un multisolvente, obteniendo similares *precision* y *recall*. Al comparar los resultados entre solventes, se aprecia que los valores de las métricas para DMF son más sensibles y presentan mayor variabilidad entre corridas. Posiblemente esto se deba a la diferencia en cuanto a cantidad de datos usados, dado que el modelo monosolvente de DMF fue entrenado solamente con aproximadamente 900 instancias, mientras que el modelo específico de GBL con un conjunto bastante más grande (de 6 mil observaciones).

Para analizar en mayor detalle esta posible tendencia de mejora al usar el modelo específico para GBL, se analizaron los resultados por clase para este solvente en particular[2].

[2] Nótese que el coeficiente de Matthew no está definido por clase, por lo que no se puede usar esta métrica para un estudio por clases.

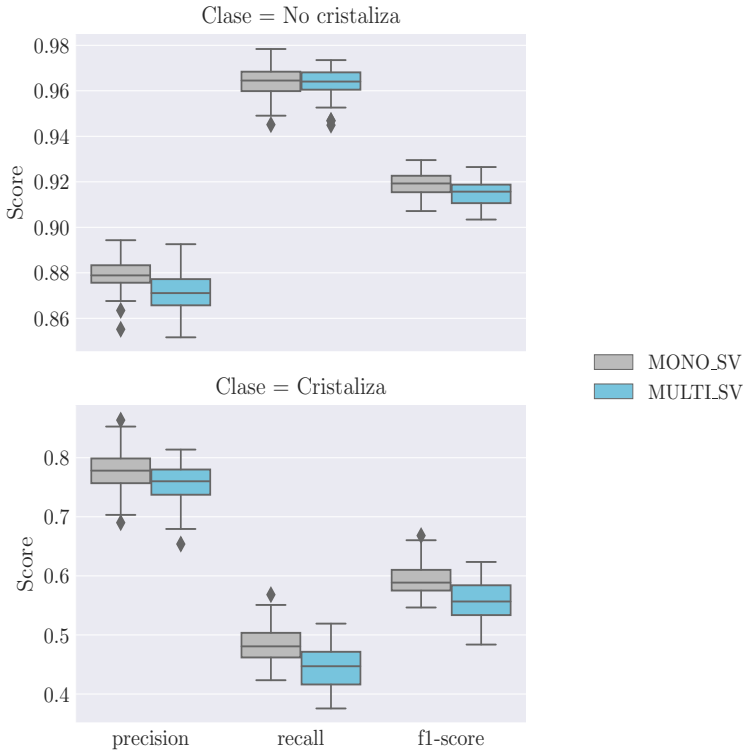

Fig. 5.3: Comparación entre modelos de predicción de cristalización mono y multisolvente (`MONO_SV` y `ENCODED_SV`) analizando su desempeño por clase: `Cristaliza` y `No cristaliza`. La evaluación se realizó sobre instancias del solvente GBL, calculando las métricas por clase y usando GBC como algoritmo de clasificación.

En la figura 5.3 se aprecia que el efecto de especificidad no es igual en ambas clases, el modelo `MONO_SV` logró un mayor *recall* y *precision* en la clase positiva (`Cristaliza`) que el modelo multisolvente `ENCODED_SV`. En cambio, para la clase negativa (`No Cristaliza`) solo mejoró el *precision* del modelo monosolvente, manteniéndose el mismo *recall*. Estos resultados indicarían que para los datos experimentales del solvente GBL, un modelo entrenado específicamente con datos de este solvente permite predecir mejor para las instancias positivas donde efectivamente cristaliza, incrementándose el *recall* en un 3 % aproximadamente. En cambio, al usar un modelo entrenado en otros solventes se obtuvo una ligera menor precisión, afectando la clase negativa mayoritaria donde no cristaliza el producto de síntesis. Esto indicaría que el entrenamiento en otros solventes pudo haber generado cierto ruido en las predicciones de `ENCODED_SV`.

Cabe resaltar que al comparar los resultados entre las clases, en general se aprecia que para la clase `No cristaliza` se obtienen mejores resultados. Esto era esperable dado que es más fácil predecir correctamente la clase negativa dado su gran proporción de instancias.

5.1.2. Conclusiones

Como conclusión se considera que para esta magnitud de cantidad de datos (del orden menor a las 10 mil observaciones), la especificidad por solvente presenta una leve ventaja con respecto a optar por una metodología multisolvente al momento de desarrollar modelos de predicción de cristalización.

Sin ir más lejos, al estudiar distintos clasificadores con *datasets* que incluyen o no al

solvente como parte de las condiciones de reacción, no se apreció una diferencia clara en la predicciones que indique que considerar al solvente aporta de manera positiva a los modelos.

En efecto, al realizar modelos específicos por solvente solamente para el mayoritario (GBL) se observó una leve tendencia a favor del modelo específico por sobre el multisolvente (mejorando particularmente la precisión de la clase negativa mayoritaria), mientras que para DMF (aproximadamente mil instancias) esta misma tendencia bien podría deberse a la variabilidad de las métricas a causa de la poca cantidad de observaciones.

Si bien el enfoque de un modelo multisolvente no mostró resultados positivos para este volumen de datos, en términos teóricos resulta relevante poder aplicar esta clase de clasificadores en pos de poder generalizar la predicción de cristalización a sistemas químicos no explorados aún (para los cuales aún no se tengan datos). De allí que se propone como trabajo futuro experimentar con un mayor volumen de datos donde los solventes estén balanceados.

De acuerdo a este análisis, se procedió a estudiar cómo mejorar los métodos de clasificación de cristalización de perovskita usando el enfoque monosolvente. Para ello se consideró al sistema que emplea únicamente GBL como un sistema de referencia de estudio al ser el solvente mayoritario, desestimando los datos de experimentaciones llevadas a cabo con DMSO y DMF. En la siguiente sección se explorarán técnicas para poder mejorar los modelos de predicción de cristalización desde un punto de vista metodológico, considerando las características de los datos disponibles para GBL.

5.2. Desarrollo de modelos

En el desarrollo de los modelos se estudió aplicar técnicas que permitiesen encarar las dificultades que conlleva trabajar con una acotada cantidad de instancias. Para ello, se estudió si la aplicación del método de modelos ensambles permite obtener mejores resultados que el uso de estimadores individuales (incluso habiendo optimizados estos últimos).

5.2.1. Metodología

Para el desarrollo y evaluación de los modelos de esta sección, se definió un *pipeline* común de trabajo y se consideró un un conjunto de datos de evaluación. Este último se usó para comparar el desempeño de los modelos entre sí, considerando las métricas previamente estudiadas que mejor se adaptaron a las características de este *dataset*).

- **Conjunto de Evaluación**

Se inició con la preparación del conjunto de evaluación, separando del conjunto total de datos del solvente GBL un subconjunto de testeo, el cual se denominó `TEST_SET`. La motivación de esta práctica consiste en poder obtener una evaluación de las hipótesis planteadas, de modo que esta se dé en un conjunto independiente no empleado durante el desarrollo de los modelos.

Para esta muestra de evaluación se tomó el 10 % de las instancias, manteniendo la proporción de las clases de la variable *target*, así como también la proporción de cada organoamina en el conjunto total. El remanente de datos se consideró como conjunto de aprendizaje de los modelos, el cual se denominó `LEARN_SET`.

- **Pipeline**

Para desarrollar cada uno de los modelos propiamente dichos, se empleó un mismo *pipeline* de desarrollo cuyas etapas se detallan a continuación.

1. **Preprocesamiento**: se realiza una estandarización de los datos debido a la existencia de variables en distintas escalas, puesto que se usan estimadores sensibles a las escalas de los datos (por ejemplo, KNN). Si bien no todos los tipos de estimadores a usar presentan esta misma sensibilidad, se decidió realizar la misma transformación por igual para evitar un efecto en la comparación posterior de los resultados.

2. **Búsqueda de parámetros**: Según el tipo de estimador considerado se define un espacio de parámetros de exploración, así como también un mecanismo de búsqueda, como por ejemplo *Grid Search* o *Randomized Search* (ver sección 2.4.1 del capítulo Métodos de Aprendizaje Automático). Para evaluar cada combinación de parámetros, se realiza una validación cruzada en el cual se consideran distintas particiones del `LEARN_SET` en conjuntos de entrenamiento y validación. En base a la *performance* del estimador en el conjunto de validación, se seleccionan los parámetros finales a emplear para cada estimador.

3. **Entrenamiento de estimadores finales**: se aplican las mismas transformaciones iniciales a los datos y se entrenan los estimadores individuales en todo el conjunto `LEARN_SET`, empleando los *mejores parámetros* encontrados en la etapa anterior. En caso de ser un modelo de ensamble, se realiza un paso adicional que consiste en la generación del ensamble en base a los estimadores individuales.

4. **Evaluación de modelos**: cada modelo entrenado (con los parámetros optimizados de la etapa anterior) se evalúa con el `TEST_SET`, conjunto de datos separados por fuera de la etapa de desarrollo. Para medir la *performance* final, se emplearon el par de métricas complementarias estudiadas en la sección 4, así como el coeficiente de Matthew para comparar los resultados con los del trabajo de referencia.

En la figura 5.4 se presenta el esquema general de la metodología de trabajo empleada para desarrollar los modelos de aprendizaje automático, en donde además se puede visualizar cada etapa del *pipeline* en un contexto más general de la metodología usada.

5.2.2. Ensamble de modelos

La técnica de ensamble de modelos se basa en la diversidad de los estimadores individuales a combinar, generando un estimador ensamblado más robusto que cada uno de ellos de manera individual. Esta diversidad puede surgir ya sea por los distintos algoritmos de aprendizaje que se usan, por la modalidad de entrenamiento empleada o mismo por cómo se realiza la combinación en sí misma (véase sección 2.1.2).

En este trabajo se realizó un estudio de su aplicación en la cristalización de perovskita considerando dos aspectos principales como fuentes de estimadores diversos:

(1) el sesgo que puede producir una cantidad de observaciones reducida, con una cantidad considerable

(2) la combinación de distintas técnicas de aprendizaje supervisado

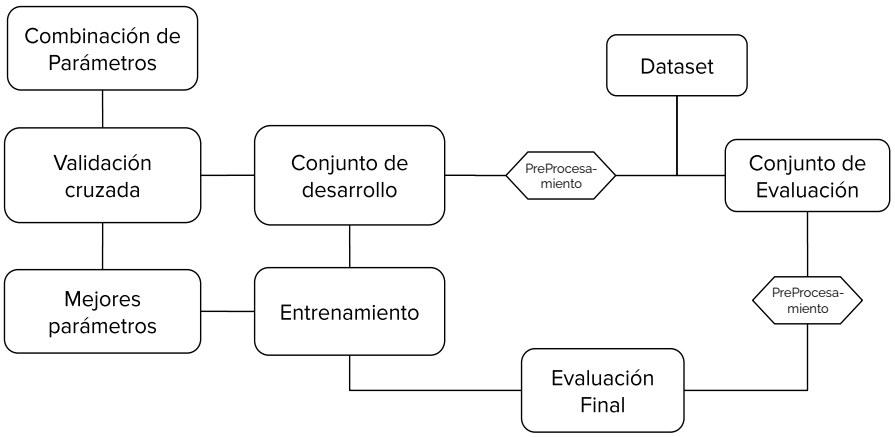

Fig. 5.4: Esquema de la metodología de desarrollo de modelos de aprendizaje automático.

En primer lugar, se estudió si ensambles homogéneos (de un mismo estimador) permiten obtener mejores resultados que un estimador individual optimizado. Posteriormente, habiendo estudiado el desempeño de cada familia de ensambles homogéneos, se estudió su combinación mediante la técnica de votación, generando modelos ensambles heterogéneos. Para ello, se buscó desarrollar un modelo ensamblado diverso al combinar aquellos que hayan tenido distinto tipo de desempeño con respecto a las clases de la variable a predecir.

- **Experimentación**

Para comenzar estudiamos el enfoque *bagging* aplicándolo a distintos tipos de estimadores. Este método busca generar independencia entre los estimadores de base a combinar mediante *bootstrap* (véase sección 2.1.2). Tradicionalmente, al aplicar *bagging* se prefiere emplear estimadores de base sensibles a los datos de entrenamiento, para favorecer que realmente sean diversos los modelos individuales combinados.

En caso de que un estimador sea «poco» sensible a los datos, al usar *bootstrap* se estarían generando conjuntos de entrenamiento muy similares, y por ende, los modelos individuales combinados también, no dando lugar a las ventajas de usar un ensamble en términos teóricos 2.1.2. En cambio, con múltiples estimadores diversos se estaría disminuyendo la varianza general del modelo al confeccionar la agregación, y por lo tanto, disminuiría el error cometido en las predicciones. No obstante, también existe la posibilidad de que el sesgo que tengan los estimadores a combinar -producto del *bootstrapping*-, superen la reducción de la variabilidad del modelo ensamblado. En este caso, se apreciaría una disminución del desempeño en el modelo ensamblado con respecto al individual entrenado en todos los datos (sin hacer *bootstrap*).

La primera experimentación consiste en estudiar si existe una diferencia al usar un estimador individual entrenado en `LEARN_SET` (con parámetros optimizados para este problema en particular), versus utilizar un ensamble homogéneo bajo la técnica de *bagging*. De esta forma, para cada par de modelos a comparar, se usa el mismo algoritmo de clasificación y el mismo conjunto de datos de entrenamiento, pero dos mecanismo de entrenamiento diferenciados.

Dado que los datos usados presentan un sesgo proveniente del muestreo en su recolección, al aplicar *bagging* sobre este tendría sentido obtener subconjuntos de datos que

reflejasen aspectos distintos de la cristalización (por ser la clase minoritaria). De allí la hipótesis de que se pueda generar cierta independencia entre los estimadores de base a combinar, al estar siendo entrenados en subconjuntos de datos diferentes. Por ejemplo, puede que haya estimadores que no sean expuestos a algunas organoaminas mientras que otros sí, pudiendo tener una distribución de la variable *target* distinta. Recordemos que se apreció en la exploración del *dataset* cierta diferencia entre el tipo de cristal obtenido según si eran organoaminas tradicionales o no tradicionales (véase figura 3.6).

Con respecto a los estimadores de base utilizados, si bien en términos teóricos tiene sentido hacer sólo *bagging* de estimadores inestables (*sensibles* al conjunto de entrenamiento), para esta experimentación se usaron estimadores de base muy estables (como SVM y LG), estables (como KNN) e inestables (como DT).

De esta forma, en esta experimentación se consideró como hipótesis que el uso de *bagging* como método de ensamble homogeneo permite estabilizar estimadores sensibles dada la reducida cantidad de datos disponibles. Esta hipótesis se evaluó para distintos tipo de estimadores, buscando corroborar que los ensambles homogéneos por *bagging* se vean favorecidos al usar estimadores de base inestables, obteniendo modelos con menor *performance* a mayor estabilidad.

Para evaluar los modelos, se realizó inicialmente una validación cruzada de tres divisiones sobre `LEARN_SET` usando el coeficiente de Matthew como métrica de evaluación. En el caso de los ensambles por *bagging*, se repitió esta validación para distinta cantidad de estimadores combinados (1, 5, 100, y 300) con el objetivo de definir la cantidad en base a donde converge su variación. Cada estimador usado (tanto en el ensamble como en el modelo individual) se entrenó utilizando los parámetros optimizados mediante *Randomized Search*. En la sección F del Apéndice se encuentran detalles de la implementación, la grilla de búsqueda hiperparámetros y los hiperparámetros óptimos encontrados. Cabe mencionar que dado el gran desbalance de clases con los que se trabajó, para los métodos DT, LG y SVM se optó por usar el hiperparámetro que permite balancear el peso de las clases (con el cual se le asigna a cada clase un peso inversamente proporcional a su frecuencia).

En la figura 5.5 se puede ver cómo varía el desempeño de los ensambles por *bagging* a medida que se incrementa la cantidad de estimadores combinados. Además, se puede comparar la *performance* del modelo ensamblado con respecto a su versión individual (entrenado directamente en cada división). Como tendencia general se observó que a mayor cantidad de estimadores, menor es la variación del ensamble (usando distintas semillas de aleatoriedad al realizar la validación cruzada). También se corroboró que efectivamente la mejora de los ensambles depende de la estabilidad del estimador usado.

Para el estimador inestable DT, el ensamble mostró una mejora del 20 % con respecto a su versión simple. En el caso de un tipo de estimador más estable como lo es KNN, si bien no se obtuvo una mejora de igual magnitud que el estimador sensible DT, el ensamble convergió a un valor similar valor al máximo alcanzado por el modelo simple en su mejor corrida. Es importante notar que el ensamble de *bagging* para este estimador estable mostró mayor robustez que su versión simple (entrenada en todos los datos de la división directamente). En cambio, el ensamble de LG mostró dar un resultado similar a la mediana de los modelos simples basados en este mismo clasificador. Se entiende que para LG, en promedio, los estimadores generados por *bootstrap* tuvieron un desempeño similar al modelo simple individual entrenado en todo el conjunto de datos, por lo que al evaluarse se tuvieron básicamente los mismos resultados. Esto indicaría que el «aprendizaje» dado por LG en estos datos termina siendo el mismo que si se entrena en toda la división

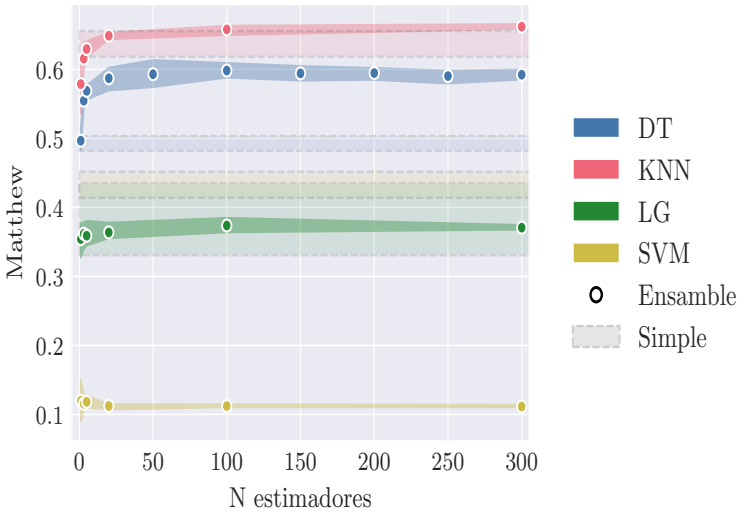

Fig. 5.5: Desempeño de ensambles *bagging* a mayor cantidad de estimadores ensamblados según el coeficiente de Matthew para distintos algoritmos de clasificación (DT, KNN, LG y SVM), evaluado por validación cruzada en `LEARN_SET`. Además, se encuentra el desempeño de sus respectivos modelos *simples* (en color claro) para su comparación. Se realizaron diez corridas distintas en todos los casos, obteniendo una franja de variación dependiente de la cantidad de estimadores en el caso de los ensambles.

o tomando distintas muestras bajo la técnica de *bootstrap*. Por otro lado, en el caso de SVM se obtuvo una considerable baja de la *performance*, siendo apenas mejor que una clasificación binaria completamente aleatoria (valor cero del coeficiente de *Matthew*). Se cree que la considerable cantidad de *features* de los datos y el desbalance de clases pudo haber producido estimadores individuales demasiado sesgados para el ensamble de este tipo de algoritmo. Consideremos que en nuestros datos la densidad de la clase negativa es mucho mayor a la densidad de la clase positiva. Esto trae como consecuencia que el hiperplano que separa las clases a clasificar pudo haber quedado corrido hacia la clase positiva en pos de disminuir el error en la predicción. En la figura 5.6 se ilustra cómo podría darse este corrimiento al subestimar un hiperplano de corte *ideal* por otro que, si bien minimiza los errores de predicción, empeora considerablemente las predicciones en la clase minoritaria.

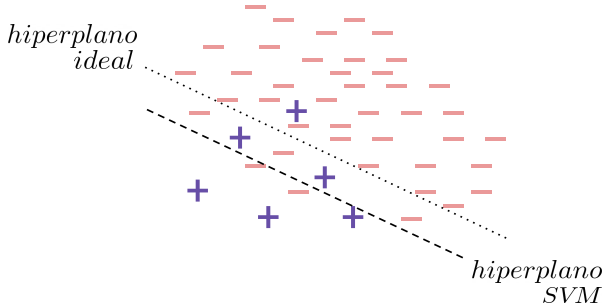

Fig. 5.6: Ilustración del efecto del desbalance de clases en modelos basados en SVM: corrimiento del hiperplano de corte hacia la clase minoritaria (en este ejemplo la clase positiva), disminuyendo la cantidad de predicciones incorrectas a costas de un peor desempeño en la clase minoritaria.

Para la evaluación final de los cuatro modelos homogéneos y sus respectivas versiones

individuales, se empleó el conjunto de datos `TEST_SET` tomando como métricas el par *F1-recall*, quedando representados los resultados en un plano bidimensional. Para los modelos ensambles se combinaron 300 estimadores individuales, cantidad desde la que se apreció convergencia para Matthew al realizar la validación cruzada (ver la figura 5.5). En la figura 5.7 se observan los resultados de la evaluación, superpuestos con los tres grupos de *performance* simulados en la sección Métricas Complementarias[3]. Los resultados de esta figura se corresponden con los obtenidos en la validación cruzada, habiendo una mejora al usar ensambles *bagging* cuanto más inestables son los estimadores combinados. La versión *bagging* de un estimador inestable como DT muestra una clara mejora, siendo esto apenas apreciable para otros estimadores como KNN y LG. Por su parte, el ensamble SVM tuvo peores resultados que el modelo simple, lo cual se corresponde con lo esperable dado su gran estabilidad y el posible corrimiento de su hiperplano de corte (produco del desbalance de clases). Cabe resaltar que los modelos basados en KNN obutvieron una *performance* del 80 %, siendo casi 10 % más que su sucesor DT. Si bien en la validación cruzada se había visto esta misma tendencia, esta era solo del 5 %.

A la hora de comparar los resultados obtenidos con las simulaciones realizadas en el capítulo Métricas de evaluación y errores, se puede ver que KNN fue el único modelo que superó lo considerado como «*alta*» *performace*, categoría donde se ubicaron ambos modelos basados en DT (ensamble y su versión individual). En cambio, tanto los modelos basados en LG como la versión individual de SVM, se ubicaron en el área de superposición ruidosa entre las categorías de *performance* regular y alta.

En base a los resultados de esta experimentación, se puede afirmar que se pudo validar la relevancia de la sensibilidad de los estimadores a la hora generar ensambles por *bagging*. Se apreció una considerable mejora en los ensambles al usar estimadores de tipo inestables como los árboles de decisión, siendo levemente mejor para otros más estables como KNN Y LG. En particular para estos datos con gran desbalance de clases, el desempeño del ensamble de SVM fue mucho peor que el modelo individual entrenado en todo el *dataset* de evaluación. Por último, se destaca que entre los distintos algoritmos de clasificación, los modelos basados en KNN obtuvieron un desempeño destacable que superó lo esperado en la etapa de la simulación de resultados esperables.

[3] Recordemos que cada área de color representa la zona de matrices ruidosa de cierta *performance*, simulando posibles errores de clasificación azarosos.

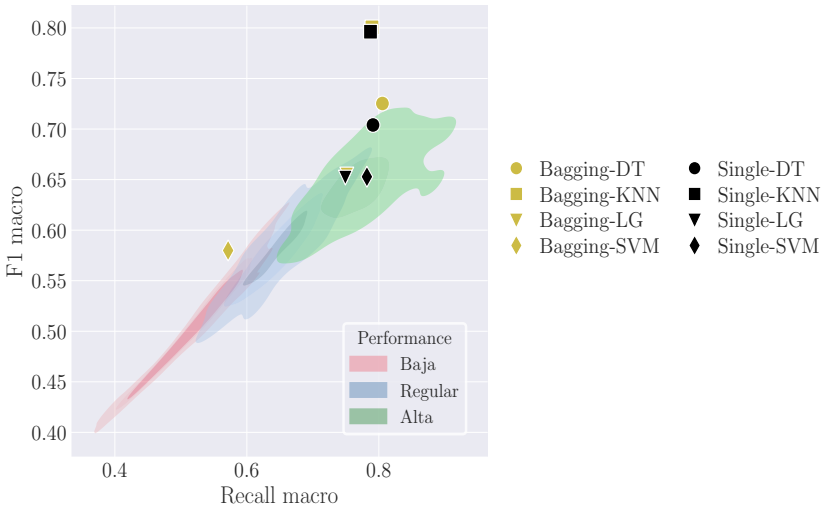

Fig. 5.7: Evaluación de ensambles *bagging* y estimadores individuales (en color amarillo y negro, respectivamente) para modelos basados en DT, SVM, LG y KNN (diferenciables por el tipo de marcador). Para los ensambles se combinaron 300 estimadores individuales.

La segunda experimentación consistió en estudiar combinaciones de modelos de modo de que se complementen según su desempeño con respecto a las dos clases de la variable *target* (`Cristaliza` y `No-Cristaliza`). Al analizar los resultados obtenidos de la validación cruzada de la experimentación anterior, se apreció que algunos estimadores (como SVM y KNN) obtuvieron un muy buen desempeño en una clase en particular (mayor al 90 %), mientras que en la otra mostraron un valor considerablemente menor (datos disponibles en la tabla 5.1). Si bien en esta tesis se prioriza al *recall* de la clase positiva, aquellos estimadores con este este valor elevado tienden a ser muy imprecisos desde una perspectiva más general (no obtiene buenos resultados para la clase negativa). De allí la motivación de generar ensambles heterogéneos complementando los distintos tipos de clasificadores. La hipótesis que se desea corroborar es que efectivamente ensambles de modelos con desempeño distinto según la clase, permiten generar modelos más robustos en general.

En la tabla 5.1 se observan los resultados de los modelos de validación cruzada, en donde además se consideraron otros tipos de modelos más complejos como GBC y RF. Para los estimadores DT, LG y RF no se apreció una diferencia marcada entre clases positiva y negativa, sin embargo para GBC, KNN y SVM sí se obtuvieron discrepancias. En base a esta diferencia, se clasificó los modelos según si tenían una clase «preferente», (es decir, si su *performance* era claramente mejor para una de las clases), tomando como umbral de diferencia entre las clases del 0,2. De esta forma, se marcaron GBC y KNN con estimadores con preferencia por la clase negativa, mientras que SVM se registró como con preferencia por la clase positiva, siendo los restantes clasificadores descriptos como sin preferencia.

Para validar la hipótesis de que ensambles de modelos con *performance* diferenciada por clase funcionan mejor que ensambles de estimadores sin preferencia por alguna clase, se realizaron distintas combinaciones analizando su desempeño en el `TEST_SET`. En la tabla 5.2 se encuentran las combinaciones exploradas para hacer los ensambles, donde el nombre

denota si se combinaron estimadores con preferencia o no.

En caso de validarse la hipótesis, los modelos ensamblados E_{SPN} y E_{PN} deberían superar los desempeños de los estimadores individuales que los componen, mostrando así que el ensamble muestra una clara mejoría por ser combinación de estimadores diversos. En cambio, se espera que el ensamble E_S no presente mejoría con respecto a su mejor estimador individual combinado puesto que los tres estimadores ensamblados tienen *performance* similar en cuanto a las clase a predecir. Similares resultados se estiman para E_{SN} y E_{SP}, a pesar de combinar estimadores sin preferencia y algún estimador con una clase preferente en particular.

	Métrica			
Estimador	Recall - Clase Negativa	Recall - Clase Positiva	Matthew	Clase preferente
DT	0.75	0.84	0.49	-
LG	0.69	0.79	0.38	-
RF	0.77	0.85	0.52	-
GBC	0.95	0.64	0.64	Negativa
KNN	0.94	0.68	0.64	Negativa
SVM	0.64	0.91	0.44	Positiva

Tabla 5.1: Desempeño de clasificadores en función de las métricas *recall* (discriminado por clase) y el coeficiente de Matthew, evaluados por validación cruzada realizada en `LEARN_SET`). La clase preferente está definida para los estimadores que tengan una clase con *recall* mayor a la otra clase por 0,2.

	Estimadores					
	Sin Preferencia			Pref. Negativa		Pref. Positiva
Ensamble	DT	LG	RF	KNN	GBC	SVM
E_S	✓	✓	✓	✗	✗	✗
E_{SP}	✓	✓	✓	✗	✗	✓
E_{PN}	✗	✗	✗	✓	✓	✓
E_{SN}	✓	✓	✓	✓	✗	✗
E_{SPN}	✓	✓	✓	✓	✗	✓

Tabla 5.2: Composición de modelos ensamblados por votación, donde el nombre indica qué tipo de estimador lo compone: `S` denota sin clase de preferencia, mientras que `P` y `N` indican preferencia por las clases positiva y negativa respectivamente.

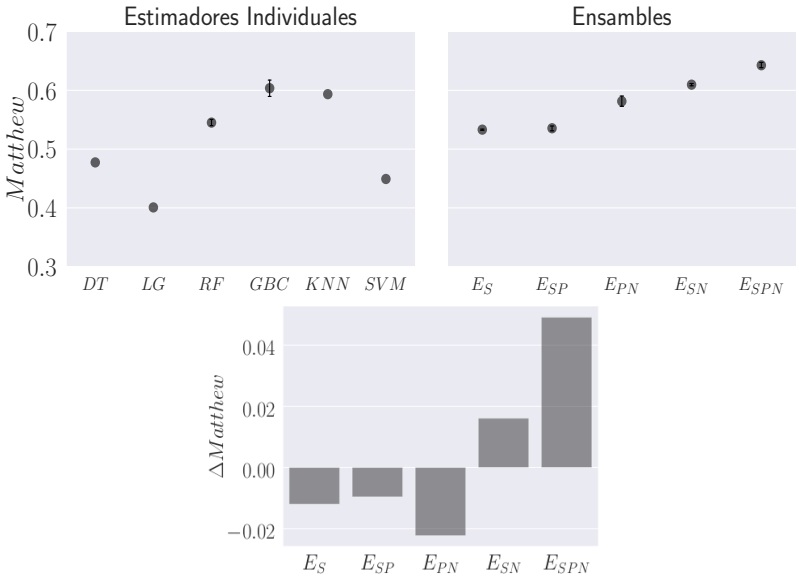

Fig. 5.8: Evaluación de los estimadores individuales y los modelos ensambles, realizado en el conjunto TEST_SET usando el coeficiente de Matthew como métrica (figuras superiores, respectivamente). Diferencia entre el coeficiente de Matthew de los ensambles y el mejor estimador individual que compone cada ensamble (figura inferior).

En la figura 5.8 se encuentran los resultados de la evaluación de los ensambles considerados en la tabla 5.2, así como también de los distintos estimadores individuales usados para generar los ensambles. Además, para analizar cuáles ensambles dieron mejores resultados que los estimadores que los componen, se muestra la diferencia entre el coeficiente de *Matthew* de cada modelo ensamblado y el mejor estimador individual que lo compone (de acuerdo a los resultados de la validación cruzada detallada en la tabla 5.1).

Al evaluar los resultados de los distintos ensambles producidos, se observó que el ensamble E_{SPN} mostró una mejoría destacable con respecto al resto de ensambles, siendo este el único que combina tanto estimadores sin preferencia de clase como con preferencia. Además, E_{SPN} se destaca porque no solo superó al mejor estimador individual que combinó (KNN), sino que también a otros clasificadores más complejos no considerados en este ensamble (como GBC).

En menor medida, otro ensamble que mostró cierta mejoría con respecto a su mejor estimador combinado fue E_{SN}. Si bien este también combinaba estimadores sin preferencia y con preferencia de clase, en particular no tenía estimadores con preferencia por la clase positiva. Análogamente se tiene al ensamble E_{SP}, el cual a diferencia del último no tenía estimadores con preferencia de la clase negativa pero sí de la clase positiva. Este modelo mostró un desempeño menor al mejor estimador combinado, lo cual indicaría que las variantes de este ensamble no se complementaron para generar un ensamble superador.

Entre los ensambles con desempeño bajo, se destaca E_{PN}) como el ensamble que más deterioro produjo, a pesar de combinar el estimador con mayor *performance* individual (GBC). Si bien este ensamble presenta diversidad al combinar modelos con clases de preferencia distinta (positiva y negativa), su diferencia principal con respecto al resto de ensambles es que no combina estimadores que *no tengan ninguna clase preferente* (es decir, que tengan similar rendimiento por clase). Se cree que esta pueda ser la razón por la cual este modelo se destaca por resultado negativo, lo cual indicaría que considerar estimadores

sin preferencia resulta importante para generar cierta «estabilización» en las predicciones del ensamble, sin depender tanto de los modelos con una clase preferencial. Queda para un trabajo futuro poder analizar otras combinaciones de estimadores individuales, en donde se pueda validar esta hipótesis con otros estimadores.

5.2.3. Conclusiones

Como conclusión del estudio de modelos ensamblados para esta aplicación en particular, destacamos la importancia de analizar en qué aspectos los modelos individuales se pueden complementar de modo que su ensamble tenga una *performance* superadora. Por ejemplo, al comparar ensambles *bagging* con respecto a los modelos individuales usando distintos estimadores, no se apreció una mejora destacable. En cambio, al considerar la diversidad de modelos con respecto a su desempeño *por clase*, se logró desarrollar un ensamble con mayor *performance* que cualquier otro modelo individual considerado. Esta diferencia de resultados enfatiza la relevancia de estudiar bien en qué aspectos la combinación de modelos puede generar ensambles interesantes, considerando las particulares del conjunto de datos disponible (en este caso la diferencia de *performance* por clase responde al desbalance de clases en los datos usados).

Por otro lado, también se pudo corroborar la relevancia de la estabilidad del estimador para los ensambles de tipo *bagging*. Para estimadores inestables como los árboles de decisión, se vio una mejora en el ensamble. Mientras que para los estimadores estables como K-vecinos más cercanos o regresión logística, se obtuvieron resultados similares entre los ensambles y la versión individual del modelo. Por lo que en este último caso, la agregación de múltiples modelos confeccionados por *bootstrap* no produjo una reducción de la varianza que superase los sesgos que cada modelo individual tenía, teniendo así el ensamble una *performance* similar o *peor* que los estimadores ensamblados.

Además, comparamos los resultados obtenidos de los modelos con las simulaciones realizadas en el capítulo Métricas de evaluación y errores, las cuales permitían diferenciar en un mapa bidimensional de métricas hasta qué punto la *performance* de un modelo podía verse distorsionado por el efecto del ruido o error (resultando sensible a factores no controlables por el desarrollo del modelo en sí). De esta forma, se pudo categorizar los modelos realizados en base a las 3 categorías de *performance* consideradas en el capítulo previamente mencionado, teniendo así una forma de considerar cómo el ruido pueden afectar las métricas de un modelo y por lo tanto, teniendo mayor criterio a la hora de comparar la *performance* de distintas técnicas.

6. CONCLUSIONES Y TRABAJO FUTURO

6.1. Conclusiones

En este trabajo desarrollamos distintos modelos de clasificación para predecir la cristalización de perovskita usando aprendizaje automático, poniendo énfasis en un abordaje metodológico que considere trabajar con datos experimentales escasos y desbalanceados.

Para ello, en una primera etapa analizamos las posibles fuentes de error que pueden generar *confusiones* en la predicción, ya sea que provengan de la recolección de los datos o del desarrollo de los modelos. Para entender cómo estas fuentes de error pueden impactar en la *performance* de los modelos, consideramos ciertas matrices de confusión de interés y las perturbamos simulando el efecto de dichos errores. En base a estas simulaciones, vimos que trabajar con pares de métricas complementarias en una representación bidimensional permite discernir mejor entre modelos que presentan similar *performance* al usar métricas individuales.

En una segunda etapa, nos abocamos a estudiar la influencia del solvente en la síntesis de perovskita mediante cristalización. Para ello desarrollamos modelos con dos enfoques distintos: específicos por solvente o multi-solvente. Los resultados obtenidos mostraron una leve mejora al emplear el enfoque específico por solvente, la cual consideramos que puede deberse a un posible sobreajuste del modelo a los datos dada su acotada cantidad disponible. Debido a que la cantidad de instancias por solvente era desbalanceada en el conjunto de datos empleado, limitamos el problema a un único solvente, trabajando solamente con aquel con la mayor proporción de datos.

En una tercer etapa, estudiamos la generación de modelos a través del método de ensambles homogéneos por *bagging*, usando estimadores de distinta estabilidad (comúnmente empleados para problemas de clasificación como DT, KNN, LG y SVM). Se pudo corroborar que cuanto más estables son los estimadores usados en *bagging* (como SVM o LG), menor es la mejora de los modelos de ensamble con respecto a usar un único estimador individual. Entre los estimadores empleados, obtuvimos mejores resultados para los ensambles de estimadores inestables como DT y, en menor medida, KNN.

Por último, en la cuarta etapa del trabajo, estudiamos la generación de modelos a través del método de ensambles heterogeneos. Para ello categorizamos distintos estimadores en base a su desempeño por clase -con preferencia por alguna de las dos clases o sin preferencia alguna-, estudiando combinaciones de estimadores de una o múltiples categorías. Los resultados obtenidos mostraron que el ensamble con mayor diversidad de estimadores (distinto tipo de preferencia y sin preferencia de clases), logró la mejora más alta en *performance*. El resto de ensambles, incluso a pesar de combinar estimadores sin preferencia y con preferencia por alguna clase en particular, no mostraron el mismo nivel de mejoría. Por lo que recomendamos seguir explorando cómo complementar modelos que tengan una *performance* preferencial por clase mediante ensambles, en particular si se tratan de datos desbalanceados en clase.

En cuanto a las lecciones aprendidas mediante este trabajo, destacamos en primer lugar las dificultades que conlleva trabajar con datos experimentales de un problema real.

En particular, debido a que el muestreo y recolección de los datos no se realizó con el fin de desarrollar sistemas predictivos, existen ciertas consideraciones metodológicas a

tener en cuenta. Desde la poca cantidad de datos disponibles hasta los distintos sesgos cometidos, no solo en la clase a predecir sino también entre las distintas agrupaciones de los datos (como por ejemplo, según la organoaminas, el solvente, entre otros). Esto repercute en qué contextos se pueden usar estos modelos, dado que la evaluación está limitada en estas condiciones de recolección y por lo tanto la aplicación debiese darse en contexto similares donde los mismos sesgos se encuentren presentes.

Por otro lado, resultó importante comprender cómo se desea medir el desempeño de los modelos, analizando qué métricas se ajustan al problema y las características de los datos manipulados. Por ejemplo, una evaluación basada en una métrica estándar como la precisión habría generado resultados sobre «*prometedores*» debido al desbalance hacia la clase mayoritaria negativa. No obstante, al no tener la clase negativa la misma relevancia que la clase positiva en este problema en particular, no estaríamos evaluando apropiadamente los sistemas predictivos.

6.2. Trabajo Futuro

Sobre el trabajo realizado, nos han quedado pendientes áreas para seguir investigando, en particular asociadas a cómo aplicar técnicas de aprendizaje automático al usar datos experimentales escasos y desbalanceados.

En nuestro trabajo, para entender cómo el desbalance y la cantidad de datos pueden afectar a las predicciones, trabajamos con matrices de confusión con *ruido sintético* analizando cómo variaban las métricas de clasificación ante distintos cambios en los valores de las matrices. Si bien este enfoque permite abstraerse de los atributos correspondientes a los datos experimentales disponibles, se encuentra limitado debido a la forma en que se simulan los errores en las matrices de confusión. Una propuesta de camino a seguir es directamente generar observaciones sintéticas, para así poder estudiar cómo influye la cantidad de datos disponibles y el desbalance de clases analizando distintas posibles muestras. Además, de esta forma nos permitiría considerar distintas agrupaciones que existen en los datos de cristalización (por ejemplo, según organoamina o solvente).

Por otro lado, otra propuesta de continuación del trabajo es investigar mecanismos de fusión de datos para combinar datos experimentales de distinto origen. Si bien combinar datos recolectados de distintos sistemas experimentales tiene sus ventajas, como la posibilidad de generar conjuntos de datos de mayor tamaño, trae consigo desafíos particulares ligados a su abordaje metodológico (por ejemplo, vinculado a las condiciones de recolección de datos, el contexto donde pueden ser evaluados y usados los modelos resultantes del conjunto de datos fusionados, entre otros).

En cuanto al problema químico de la cristalización de perovskitas usando el *dataset* disponible, queda pendiente estudiar otras maneras de modelar la clasificación considerando otra modularización de las clases. En este trabajo simplificamos el enfoque considerando solo dos clases según si cristalizaba o no el producto de síntesis. Sin embargo, se podría estudiar un enfoque que estudie en qué condiciones se obtiene cada tipo de cristal (por ejemplo, polvo cristalino fino, pequeños cristales), mediante ensambles de modelos que consideren una clasificación multiclase. También se podría estudiar qué tipo de propiedades físico-químicas de las organoaminas resultan relevantes para los modelos, aunque este estaría sujeto al muestreo diferencial que se hizo por organoamina, por lo que se recomendaría considerar otra recolección de datos para seguir con esta línea.

APÉNDICE

A. Implementación

Toda la implementación de los modelos, el código de experimentación y generación de las figuras se encuentra disponible en el repositorio público en el siguiente link: https://github.com/beluticona/licentiate-thesis-repo.

B. Lista de propiedades físico-químicas

Name	Descripción	mean	std	min	25%	50%	75%	max
acceptorcount	Conteo de átomo de aceptación de enlaces de hidrógeno en molécula, calculado en _rxn_ph	0.16	0.43	0.00	0.00	0.00	0.00	2.00
Accsitecount	Multiplicidad del aceptador de enlaces de hidrógeno en molécula, calculada en _rxn_ph	0.21	0.56	0.00	0.00	0.00	0.00	2.00
AliphaticAtomCount	Número de átomos alifáticos en la molécula	5.05	2.49	0.00	3.00	5.00	6.00	13.00
AliphaticRingCount	Número de anillos alifáticos	0.16	0.37	0.00	0.00	0.00	0.00	1.00
AromaticAtomCount	Número de átomos aromáticos en la molécula	1.53	2.65	0.00	0.00	0.00	3.00	6.00
AromaticRingCount	Número de anillos aromáticos	0.26	0.44	0.00	0.00	0.00	0.50	1.00
ASA	Área de superficie accesible de agua de la molécula, calculada en _rxn_ph	268.14	62.39	163.99	229.50	256.81	302.12	524.51
ASAH	Área de superficie accesible de agua de todos los átomos hidrófobos con carga parcial positiva, calculada en _rxn_ph	206.03	76.24	33.15	173.37	201.33	232.38	481.73
ASAP	Área de superficie accesible de agua de todos los átomos polares con carga parcial positiva, calculada en _rxn_ph	62.10	34.58	13.65	41.63	50.92	79.62	155.76
ASA-	Área de superficie accesible de agua de todos los átomos con carga parcial negativa, calculada en _rxn_ph	61.25	30.19	0.00	42.88	64.46	72.49	149.19
ASA+	Área de superficie accesible de agua de todos los átomos con carga parcial positiva, calculada en _rxn_ph	206.88	48.48	120.89	177.76	197.61	229.80	398.73
AtomCountC	Número de átomos de carbono	5.02	2.36	1.00	3.50	5.00	7.00	12.00
AtomCountN	Número de átomos de nitrógeno	1.30	0.51	1.00	1.00	1.00	2.00	3.00
AvgPol	Polarización molecular promedio (en pH indicado por _RXN_PH)	11.45	4.17	4.23	8.80	11.57	13.66	26.67
BalabanIndex	Índice de gráficos moleculares de Balaban	2.23	0.50	1.00	1.96	2.19	2.50	3.75
BondCount	Número de enlaces en la molécula	17.72	6.33	7.00	14.00	18.00	21.00	40.00
CarboaliphaticRingCount	Número de anillos alifáticos compuestos únicamente de átomos de carbono	0.05	0.21	0.00	0.00	0.00	0.00	1.00
CarboaromaticRingCount	Número de anillos aromáticos compuestos únicamente de átomos de carbono	0.23	0.43	0.00	0.00	0.00	0.00	1.00
CarboRingCount	Número de anillos compuestos únicamente de átomos de carbono	0.28	0.45	0.00	0.00	0.00	1.00	1.00
ChainAtomCount	Número de átomos que forman parte de la cadena (no parte de un anillo)	4.14	2.82	0.00	2.00	4.00	6.00	13.00
ChiralCenterCount	Número de centros estereogénicos tetraédricos	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CyclomaticNumber	Número ciclomático de gráfico molecular	0.42	0.50	0.00	0.00	0.00	1.00	1.00
donorcount	Conteo de átomos del donante de enlace de hidrógeno en molécula, calculado en _rxn_ph	1.28	0.50	1.00	1.00	1.00	1.50	3.00
donsitecount	Multiplicidad del donante de enlaces de hidrógeno en molécula, calculada en _rxn_ph	3.28	1.18	1.00	3.00	3.00	3.50	6.00
frAmidine	Número de grupos amidinos	0.02	0.15	0.00	0.00	0.00	0.00	1.00
frAr NH	Número de aminas aromáticas	0.02	0.15	0.00	0.00	0.00	0.00	1.00
frArN	Número de n grupos funcionales adjuntos a aromáticos	0.00	0.00	0.00	0.00	0.00	0.00	0.00
frDihydropyridine	Número de dihidropiridinas	0.00	0.00	0.00	0.00	0.00	0.00	0.00
frGuanido	Número de grupos de guanidina	0.02	0.15	0.00	0.00	0.00	0.00	1.00
frImine	Número de imines	0.02	0.15	0.00	0.00	0.00	0.00	1.00
frNH0	Número de aminas terciarias	0.02	0.15	0.00	0.00	0.00	0.00	1.00
frNH1	Número de aminas secundarias	0.14	0.35	0.00	0.00	0.00	0.00	1.00
frNH2	Número de aminas primarias	0.28	0.63	0.00	0.00	0.00	0.00	3.00
frPiperdine	Número de anillos de piperidina	0.02	0.15	0.00	0.00	0.00	0.00	1.00
frPiperzine	Número de anillos de piperzina	0.00	0.00	0.00	0.00	0.00	0.00	0.00
frPyridine	Número de anillos de piridina	0.02	0.15	0.00	0.00	0.00	0.00	1.00
frQuatN	Número de aminas cuaternarias	1.12	0.50	0.00	1.00	1.00	1.00	2.00
Hacceptorcount	Fracción de los carbonos SP3 (valor FSP3)	0.16	0.43	0.00	0.00	0.00	0.00	2.00
Hdonorcount	Número de anillos heteroalifáticos	1.30	0.51	1.00	1.00	1.00	2.00	3.00
HeteroaliphaticRingCount	Número de anillos heteroaromáticos	0.12	0.32	0.00	0.00	0.00	0.00	1.00
HeteroaromaticRingCount	Índice Hyper Wiener de gráfico molecular	0.02	0.15	0.00	0.00	0.00	0.00	1.00
HyperWienerIndex	Punto isoeléctrico de la molécula	128.56	221.26	1.00	21.00	58.00	130.50	1365.00
LargestRingSize	Número de miembros en el anillo más grande	2.44	2.92	0.00	0.00	0.00	6.00	6.00
LengthPerpendicularToTheMaxArea	Longitud perpendicular al área de proyección máxima	5.25	0.79	3.40	5.06	5.27	5.70	7.19
LengthPerpendicularToTheMinArea	Longitud perpendicular al área de proyección mínima	8.72	2.48	4.81	6.88	8.86	9.93	19.06
MaximalProjectionArea	Masa molecular de organoamina como sal	38.18	11.60	17.05	29.04	39.14	45.17	80.65
MaximalProjectionRadius	Masa molecular de organoamina como especie neutra	4.44	1.21	2.67	3.63	4.49	4.97	9.65
maximalprojectionsize	Radio de proyección máxima	5.25	0.79	3.40	5.06	5.27	5.70	7.19
MinimalProjectionArea	Área de proyección mínima	22.70	4.79	13.66	19.16	22.95	25.60	34.16
MinimalProjectionRadius	Radio de proyección mínimo	3.21	0.40	2.33	2.90	3.25	3.40	4.32
minimalprojectionsize	Radio de proyección mínimo	8.72	2.48	4.81	6.88	8.86	9.93	19.06
MolPol	Polarización molecular (en pH indicado por _RXN_PH)	11.29	3.93	4.04	9.00	11.42	13.47	24.35
PolarSurfaceArea	Área de superficie polar topológica 2D, calculada en _rxn_ph	31.51	12.92	14.14	27.64	27.64	32.08	77.63
Refractivity	Refractividad calculada	41.92	11.17	21.21	34.41	41.21	50.10	71.89
RingAtomCount	Número de átomos que forman parte de un anillo (no parte de una cadena)	2.44	2.92	0.00	0.00	0.00	6.00	6.00
RotatableBondCount	Número de enlaces rotativos	1.44	1.94	0.00	0.00	1.00	2.00	10.00
SmallestRingSize	Número de miembros en el anillo más pequeño	2.44	2.92	0.00	0.00	0.00	6.00	6.00
VanderWaalsSurfaceArea	Van der Waals superficie de la molécula	187.06	65.46	78.34	146.29	178.29	221.16	417.97
VanderWaalsVolume	Van der Waals Volumen de la molécula	105.90	35.96	42.58	81.22	107.81	127.74	229.29
WienerIndex	Índice de gráfico molecular	55.37	65.34	1.00	15.50	32.00	65.50	364.00
WienerPolarity	Polaridad de Wiener del gráfico molecular	4.19	3.79	0.00	1.00	3.00	7.00	15.00

Fig. 6.1: Descriptores físico-químicos de las aminas orgánicas usadas en el *dataset* RAPID [1], también denominadas `FEAT_CHEM` en el trabajo.

C. Simulación de ruido en matrices de confusión

En general se aprecia que Matthew y el resto de métricas en su versión macro son menos afectadas por el desbalance de clases, pudiéndose en estas distinguir mejor la categoría original de *performance*.

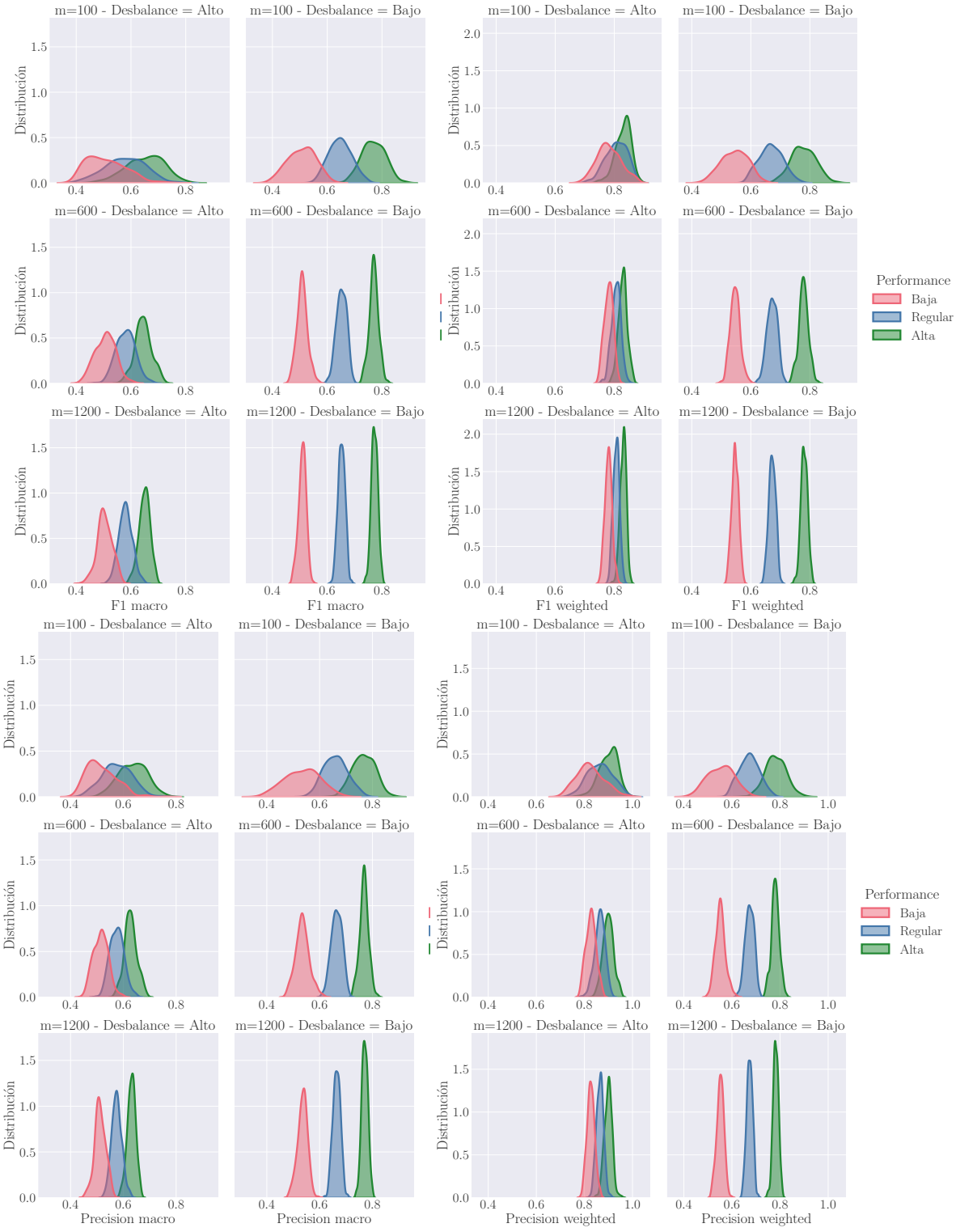

Fig. 6.2: Distribución de las métricas *F1* y *precision* (y sus respectivas versiones macro y pesada) para matrices de confusión con ruido simulado del modelo de los datos. Las matrices representan la evaluación sobre conjuntos de datos con distinto desbalance (δ), tamaño (m) y representando modelos con distinta *performance*.

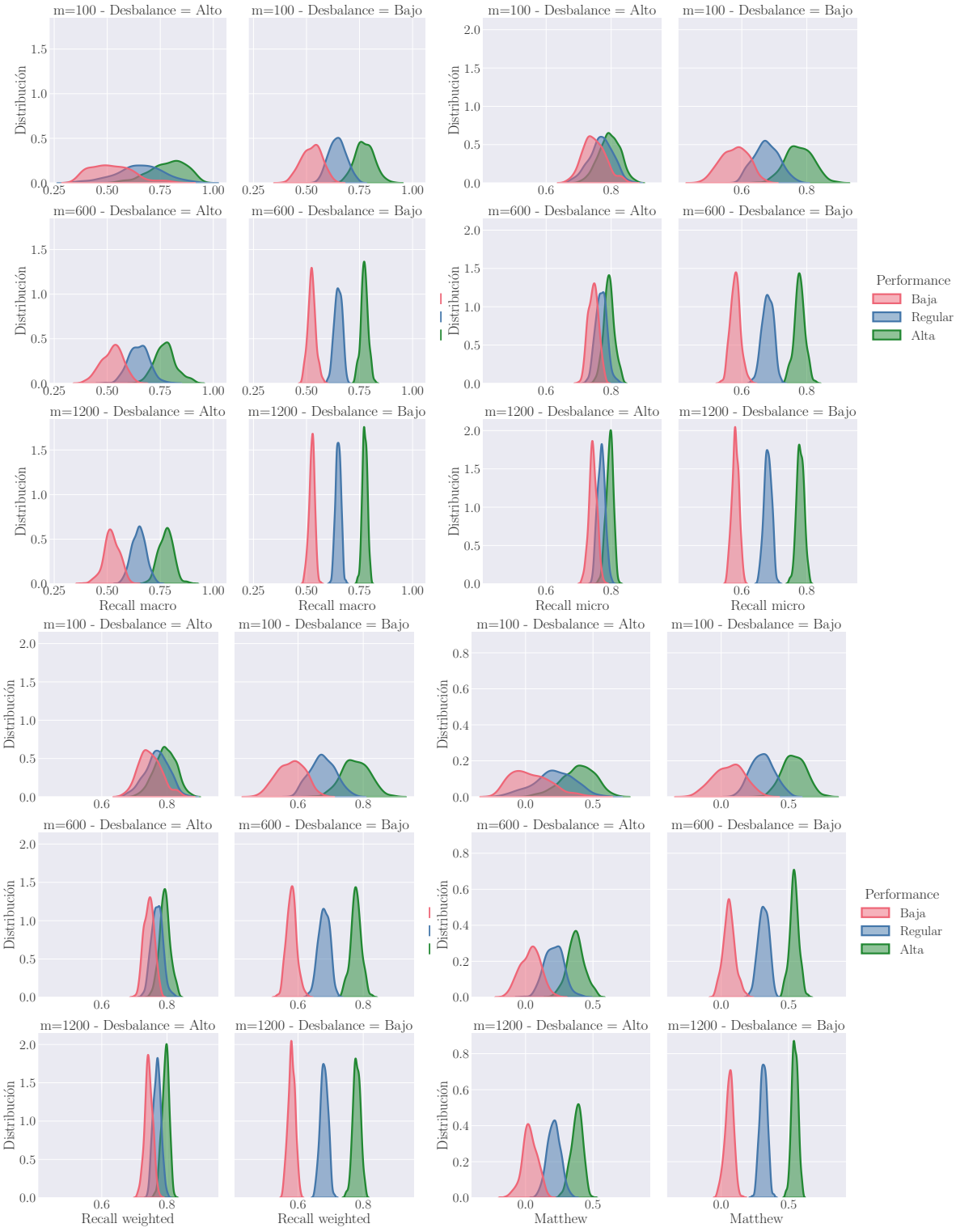

Fig. 6.3: Distribución de las métricas *recall* (en versión pesada, micro y macro) y *Matthew* para matrices de confusión con ruido del modelo simulado. Las matrices representan la evaluación sobre conjuntos de datos con distinto desbalance (δ), tamaño (m) y representando modelos con distinta *performance*.

La versión *weighted* de las métricas son más sensibles a cambios dado en las instancias negativas dado el desbalance negativo del conjunto de evaluación. De allí la dificultad de identificar cada categoría de *performance*, definida en función de las instancias positivas.

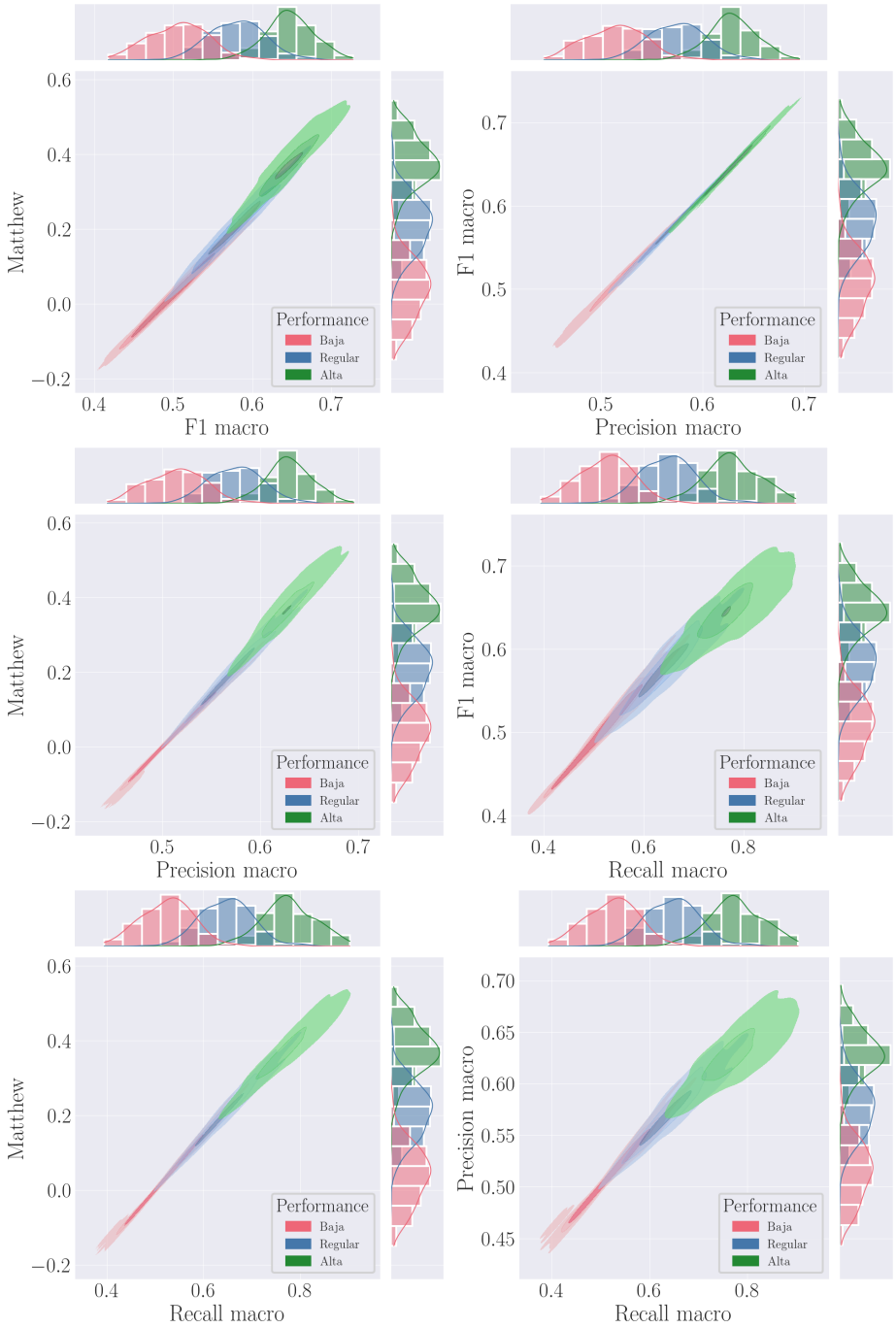

Fig. 6.4: Representación bidimensional de la complementariedad entre combinaciones de las métricas *F1 macro*, *Matthew* y *recall macro*. La perturbación se debe al ruido simulado en los datos de evaluación ($Noise_{data}$). Se dice que son complementarias si se identifican zonas diferenciadas para cada tipo de *performance* considerada

D. Métricas Complementarias

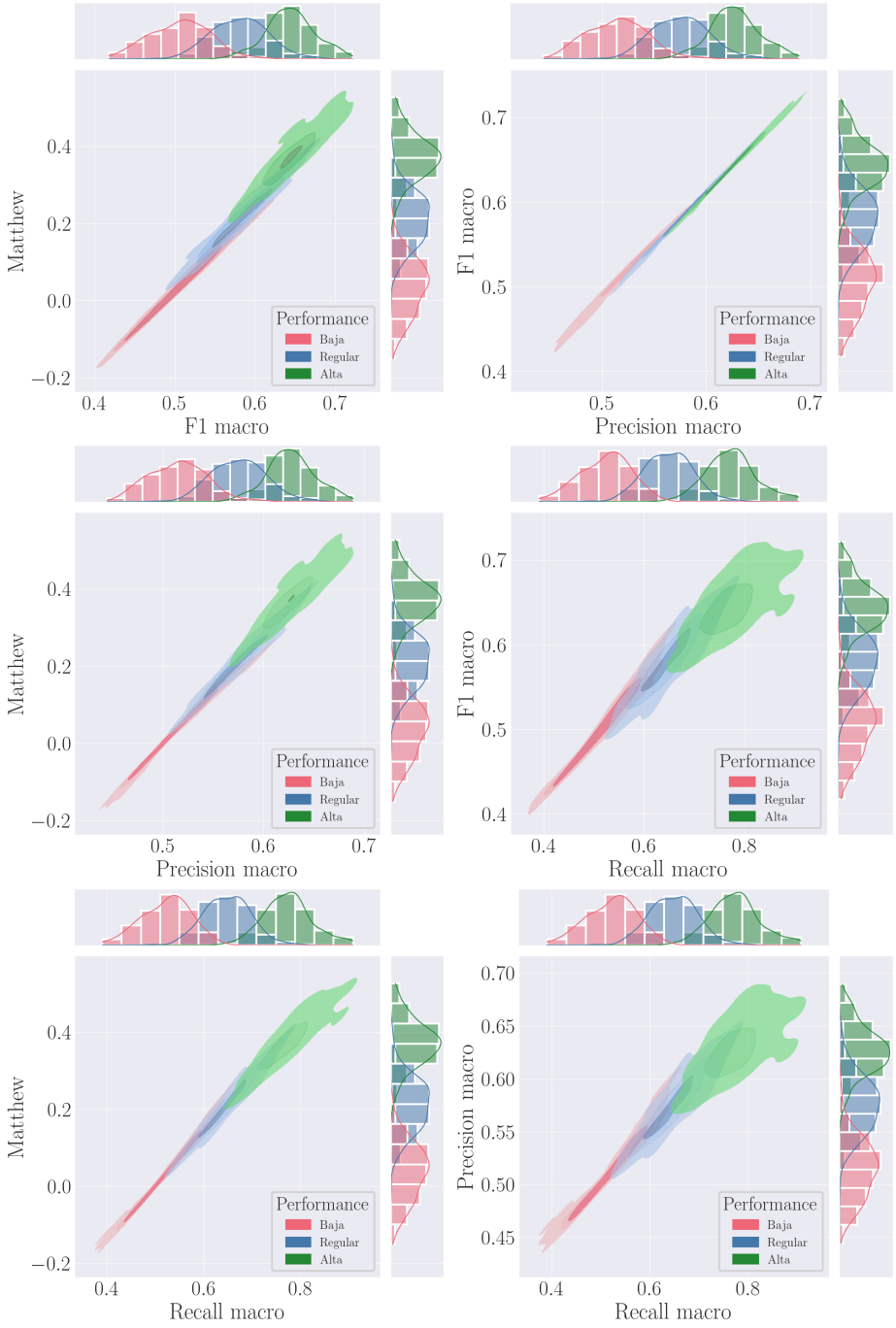

Fig. 6.5: Representación bidimensional de la complementariedad entre combinaciones de las métricas *F1 macro*, *Matthew* y *recall macro*. La perturbación se debe al ruido simulado en los datos de evaluación ($Noise_{data}$). Se dice que son complementarias si se identifican zonas diferenciadas para cada tipo de *performance* considerada

E. Influencia del solvente

E.1. Hiperparámetros para entrenamiento de los modelos

Hiperparámetros usados para los algoritmos de aprendizaje y configuración para la validación cruzada *k-fold*.

```
knn_params = {
    'n_neighbors' : 5,
    'algorithm' : auto,
    'metric' : 'minkowski',
    'p' : 2
}

gbc_params = {
    'loss' : 'log_loss',
    'learning_rate' : 0.1,
    'n_estimators' : 100,
    'criterion' : 'friedman_mse',
    'min_samples_split' : 2,
    'max_depth' : 3
}

rf_params = {
    'n_estimator' : 100,
    'criterion' : 'gini',
    'min_samples_splitint' : 2,
    'max_features' : 'sqrt',
    'bootstrap' : True,
    'class_weight' : None
}

k_fold_config = {
    'random_state': SEED,
    'n_repeats': 2,
    'n_splits': 3,
}
```

E.2. Multisolvente vs monosolvente

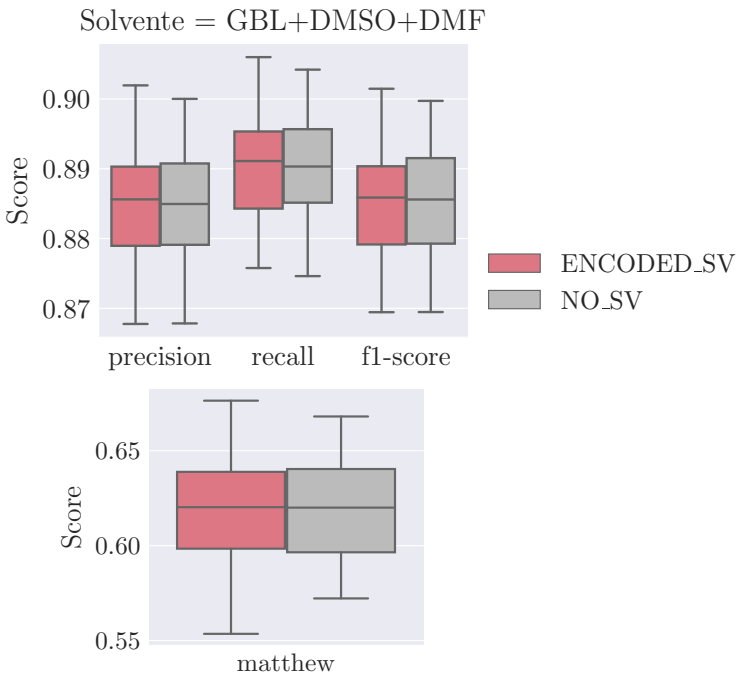

Fig. 6.6: Relevancia del solvente como predictor en la predicción de cristalización de HOIPs. Se compara un modelo con el solvente de reacción codificado (`ENCODED_SV`), versos un modelo sin información del solvente (`NO_SV`). Las métricas se encuentran en su versión pesada por clase, a excepción de Matthew. Se empleó RF como técnica de clasificación y se usaron instancias de los 3 posibles solventes: GBL, DMSO y DMF.

F. Modelos de ensamble

Grilla de parámetros explorada mediante Búsqueda Aleatoria (o *RandomSearch* en inglés) para encontrar los valores óptimos de para los modelos ensamblados por *voting*. Para cada parámetro con más de un valor explorado, se deja comentado el valor del parámetro final encontrado.

```
LG_params_grid = dict(
                    C=[0.15]
                    dual=[False],
                    penalty=['l1','l2'], #l1
                    class_weight='balanced',
                    solver=['saga'],
                    max_iter = [3000]
                    )
RF_params_grid = dict(
                    min_samples_split=[10,7,15,20], #10
                    min_samples_leaf=[3,6], #6
                    warm_start=[False],
                    class_weight=['balanced'],
                    max_depth = [8],
```

```
                    )
GBC_params_grid = dict(
                    subsample=[0.9],
                    min_samples_split=[10,7,15,20], #7
                    min_samples_leaf=[5],
                    max_depth = [7],
                    learning_rate = [0.15]
                    )
SVM_params_grid = dict(
                    C=[0.15],
                    class_weight=['balanced'],
                    kernel=['linear', 'poly', 'rbf'], #rbf
                    degree=[3,4,5], #3
                    )
KNN_params_grid = dict(
                    n_neighbors = [6]
                    weights=['uniform', 'distance'], #distance
                    )
BAGGING_params = {
                    'weights' : 'distance',
                    'n_estimators': 300
                }
```

Bibliografía

[1] Pendleton, I.; Caucci, Mary K.; Tynes, M.; Dharna, A.; Najeeb, M.; Chan, E.; Norquist, A.; Schrier, J. *Untangling How Machines 'Learn' Perovskite Crystallization Chemistry Through Stepwise Data Sample Comparisons* https://petreldata.alcf.anl.gov/mdf/detail/darpa_sd2_perovskites_v1.2/

[2] Tao Q.; Xu P.; Li M.; Lu, W. *Machine learning for perovskite materials design and discovery* DOI: 10.1038/s41524-021-00495-8

[3] Tao Q.; Xu P.; Li M.; Lu, W. *Halide Perovskite Photovoltaics: Background, Status, and Future Prospects* DOI: 10.1021/acs.chemrev.8b00539

[4] Radha K. Kothan daraman, Yan Jiang, Thomas Feurer, Ayodhya N. Tiwari, Fan Fu. *Near-Infrared-Transparent Perovskite Solar Cells and Perovskite-Based Tandem Photovoltaics*. Volume 6, Issue 2, Small Methods. DOI: 10.1002/smtd.202000395

[5] Fang, Raissa, R., Abdu-Aguye, M., Adjokatse, S., Blake, G.R., Even, J. and Loi, M.A. (2015), *Photophysics of Organic–Inorganic Hybrid Lead Iodide Perovskite Single Crystals*. Adv. Funct. Mater., 25: 2378-2385. https://doi.org/10.1002/adfm.201404421

[6] National Renewable Energy Laboratory. NREL Efficiency Chart. Tao Q.; Xu P.; Li M.; Lu, W. *Organometal halide perovskites as visible-light sensitizers for photovoltaic cells* https://www.nrel.gov/pv/cell-efficiency.html/

[7] Kojima, A., Teshima, K., Shirai, Y.; Miyasaka, T. Organometal halide perovskites as visible-light sensitizers for photovoltaic cells. J. Am. Chem. Soc. 131, 6050–6051 (2009). Tao Q.; Xu P.; Li M.; Lu, W. *Organometal halide perovskites as visible-light sensitizers for photovoltaic cells*

[8] Kour, R., Arya, S., Verma, S., Gupta, J., Bandhoria, P., Bharti, V., Datt, R., Gupta, V., Potential Substitutes for Replacement of Lead in Perovskite Solar Cells: A Review. Global Challenges 2019, 3, 1900050. DOI: 10.1002/gch2.201900050

[9] Shevlin, M. *Practical High-Throughput Experimentation for Chemists*. ACS Medicinal Chemistry Letters 2017 8 (6), 601-607 DOI: 10.1021/acsmedchemlett.7b00165

[10] Weike Ye; Chi Chen; Zhenbin Wang; Iek-Heng Chu; Shyue Ping Ong (2019). Deep neural networks for accurate predictions of crystal stability [Dataset]. DOI: 10.5061/dryad.760r5b6

[11] Kim, Chiho; Tran, Huan D.; Krishnan, Sridevi; Ramprasad, Rampi (2017). Data from: A hybrid organic-inorganic perovskite dataset [Dataset]. DOI: 10.5061/dryad.gq3rg

[12] T. Thomas (2020). Data for: Crystal structure classification in ABO3 perovskites via machine learning [Dataset]. DOI: 10.17632/dfsf6n6g7y.1

[13] A. W. Stewart (2021). The effect of solvent composition on stability and performance of CsPbIBr2 [Dataset]. DOI: 10.17632/j9p2zz849v.1

[14] Z. Li, M. Najeeb, L. Alves, A. Z. Sherman, V. Shekar, P. Parrilla, I. M. Pendleton, W. Wang, Philip W. Nega, M. Zeller, J. Schrier, A. J. Norquist, E. M. Chan. *Robot-Accelerated Perovskite Investigation and Discovery* Chemistry of Materials 2020 32 (13), 5650-5663 DOI: 10.1021/acs.chemmater.0c01153

[15] I. Pendleton, M. K. Caucci, M. Tynes, A. Dharna, M. Nellikkal, Z. Li, E. M. Chan, A. J. Norquist, J. Schrier. *Can Machines Learn Halide Perovskite Crystal Formation without Accurate Physicochemical Features?* The Journal of Physical Chemistry C 2020 124 (25), 13982-13992 DOI: 10.1021/acs.jpcc.0c01726

[16] Cover, T. and Hart, P., 1967. Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), pp.21-27.

[17] L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Classification and Regression Trees", Wadsworth, Belmont, CA, 1984.

[18] Heller, S.R., McNaught, A., Pletnev, I. et al. *InChI, the IUPAC International Chemical Identifier*. J Cheminform 7, 23 (2015). DOI: 10.1186/s13321-015-0068-4

[19] V. Cohen-Addad, V. Kanade, F. Mallmann-Trenn, C. Mathieu *Hierarchical Clustering: Objective Functions and Algorithms*. Proceedings of the 2018 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). 2018, 378-397. DOI: 10.1137/1.9781611975031.26

[20] Song YY, Lu Y. *Decision tree methods: applications for classification and prediction*. Shanghai Arch Psychiatry. 2015;27(2):130-135. DOI:10.11919/j.issn.1002-0829.215044

[21] Noble, W. *What is a support vector machine?*. Nat Biotechnol 24, 1565–1567 (2006). DOI/10.1038/nbt1206-1565

[22] Zhi-Hua Zhou. *Ensemble methods: Foundation and Algorithms*. CRC Press. Chapman Hall. 2012.

[23] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H. and Herrera, F., 2011. *A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches.* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(4), pp.463-484.

[24] T. Hastie, R. Tibshirani and J. Friedman. Elements of Statistical Learning Ed. 2, Springer, 2009.

[25] Raschka, S., Mirjalili, V. (2017). Python Machine Learning, 2nd Ed. Birmhingham, UK: Packt Publishing. ISBN-13: 978-1787125933

[26] Ryan P. Adams. *Hierarchical Clustering*. COS 324 – Elements of Machine Learning. Princeton University.
Disponible en: https://www.cs.princeton.edu/courses/archive/fall18/cos324files/hierarchical-clustering.pdf. Consultado el 01/06/2022.

[27] J. Kirman, A. Johnston, D. A. Kuntz, M. Askerka, Y: Gao, P. Todorović, D. Ma, G. G. Privé, E. H. Sargent, *Machine-Learning-Accelerated Perovskite Crystallization*, Matter, Volume 2, Issue 4, 2020, Pages 938-947, ISSN 2590-2385, DOI: 10.1016/j.matt.2020.02.012

[28] S. Sun, A. Tiihonen, F. Oviedo, Z. Liu, J. Thapa, Y. Zhao, N. Titan P. Hartono, A. Goyal, Thomas Heumueller, C. Batali, A. Encinas, J. J. Yoo, R. Li, Zekun Ren, I. Peters, C. J. Brabec, M. G. Bawendi, V. Stevanovic, J. Fisher, T. Buonassisi. *A data fusion approach to optimize compositional stability of halide perovskites* Matter, Volume 2, Issue 4, 2020, Pages 938-947, ISSN 2590-2385, DOI: 10.1016/j.matt.2021.01.008

[29] Ian M. Pendleton, Mary K. Caucci, Michael Tynes, Aaron Dharna, Mansoor Ani Najeeb Nellikkal, Zhi Li, Emory M. Chan, Alexander J. Norquist, and Joshua Schrier *Can Machines "Learn" Halide Perovskite Crystal Formation without Accurate Physicochemical Features?* The Journal of Physical Chemistry C 2020 124 (25), 13982-13992 DOI: 10.1021/acs.jpcc.0c01726

[30] Luque, A., Carrasco, A., Martín, A. de las Heras, A. *The impact of class imbalance in classification performance metrics based on the binary confusion matrix.* Pattern Recognition 91, 216–231 (2019).

[31] Sokolova, M., Japkowicz, N. Szpakowicz, S. *Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation.* in AI 2006: Advances in Artificial Intelligence (eds. Sattar, A. Kang, B.) 1015–1021 (Springer, 2006). DOI:10.1007/11941439_114.

[32] Gu, Q., Zhu, L. Cai, Z. *Evaluation Measures of the Classification Performance of Imbalanced Data Sets. in Computational Intelligence and Intelligent Systems* (eds. Cai, Z., Li, Z., Kang, Z. Liu, Y.) 461–471 (Springer, 2009). DOI:10.1007/978-3-642-04962-0_53.

[33] Chicco, D. Jurman, G. *The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation.* BMC Genomics 21, 6 (2020).

[34] Manjunatha, S.N., Chu, YX., Jeng, MJ. et al. The Characteristics of Perovskite Solar Cells Fabricated Using DMF and DMSO/GBL Solvents. J. Electron. Mater. 49, 6823–6828 (2020). DOI: 10.1007/s11664-020-08283-8

[35] Huang P-H, Wang Y-H, Ke J-C, Huang C-J. The Effect of Solvents on the Performance of CH3NH3PbI3 Perovskite Solar Cells. Energies. 2017; 10(5):599. DOI: doi.org/10.3390/en10050599