Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Selección de características para priorización genética mediante aprendizaje automático en entornos de alta dimensionalidad

Estudante: Rubén Fernández Farelo

Dirección: Jorge Paz Ruza

María Amparo Alonso Betanzos

A Coruña, septiembre de 2025.

Para mi Madre y mi novia Lucía

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mis tutores, Jorge y Amparo, por su dedicación, orientación y apoyo durante todo el desarrollo de este trabajo. También me gustaría destacar a Bertha Guijarro, quien, aunque no figura oficialmente como tutora, se ha implicado en el proyecto y su ayuda ha sido igualmente fundamental para su desarrollo.

A nivel personal, agradezco profundamente a mi madre y a mi pareja, Lucía, por haber estado a mi lado durante estos cuatro años de carrera, tanto en los buenos momentos como en los más difíciles. Sin su apoyo incondicional, este camino no habría sido posible.

Resumen

En la priorización génica, uno de los principales retos es trabajar con conjuntos de datos biomédicos de alta dimensionalidad y etiquetado incompleto, donde solo una pequeña fracción de los genes han sido validados como relevantes. Esta situación, habitual en campos como el estudio de la restricción dietética, dificulta la construcción de modelos fiables, al aumentar el riesgo de sobreajuste, sesgo y pérdida de interpretabilidad.

Este trabajo propone una solución basada en la selección de características mediante el algoritmo *Fast-mRMR*, con el objetivo de reducir drásticamente el número de características utilizadas, conservando solo aquellas más relevantes desde el punto de vista biológico. Esto permite construir modelos más simples, interpretables y competitivos, trabajando con menos información pero de mayor calidad.

El enfoque parte de un escenario con datos parcialmente etiquetados, donde solo se dispone de ejemplos positivos confirmados, lo que refleja de forma más realista la incertidumbre inherente al problema. Los experimentos realizados sobre múltiples fuentes de datos (GO, PathDIP, GTEx, Coexpression) y sus combinaciones muestran mejoras consistentes frente a enfoques previos. Además, se evalúa el uso de técnicas de PU Learning como complemento.

Abstract

In gene prioritization, one of the main challenges lies in handling high-dimensional biomedical datasets with incomplete labeling, where only a small subset of genes is experimentally validated as relevant. This scenario, common in contexts like dietary restriction studies, makes it difficult to build reliable models without introducing bias or overfitting.

This work proposes a feature selection approach based on the *Fast-mRMR* algorithm, aiming to drastically reduce the number of features by retaining only those that are biologically informative. This strategy enables the construction of simpler, more interpretable, and competitive models, working with less data but of higher quality.

The approach is based on a partially labeled data scenario, where only confirmed positive examples are available, which more realistically reflects the inherent uncertainty of the problem. Experiments conducted on multiple data sources (GO, PathDIP, GTEx, Coexpression) and their combinations show consistent improvements over previous approaches. Additionally, PU Learning techniques are explored as a complementary component.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En los últimos años, la inteligencia artificial (IA) ha tomado un papel clave en la investigación biomédica. En particular, el aprendizaje automático (en inglés machine learning, ML) se ha consolidado como una herramienta indispensable, por su capacidad de automatizar el análisis de grandes volúmenes de datos complejos, lo que facilita la detección de patrones y la generación de hipótesis en tareas como la predicción de interacciones moleculares, el descubrimiento de biomarcadores cruciales para diagnósticos tempranos o la clasificación precisa de enfermedades, lo que allana el camino para tratamientos clínicos más personalizados.

Una de las áreas donde el aprendizaje automático ha mostrado su gran utilidad es en la priorización genética, una técnica diseñada para identificar genes que tienen una alta probabilidad de estar involucrados en un proceso biológico o patológico específico. Esta herramienta es especialmente valiosa en situaciones donde el conocimiento disponible es limitado y hay un gran número de genes candidatos, ya que ayuda a enfocar los esfuerzos experimentales en los elementos más prometedores. Al utilizar el ML para destacar los elementos más prometedores, los científicos pueden enfocar sus esfuerzos experimentales de manera mucho más eficiente y dirigida, optimizando recursos y acelerando el descubrimiento de nuevas dianas terapéuticas o mecanismos patológicos. La priorización genética impulsada por IA no solo agiliza la investigación, sino que también mejora la eficiencia del proceso experimental, impulsando un progreso más rápido y significativo en áreas de la biomedicina.

1.1 La restricción dietética como caso de estudio

La restricción dietética (RD) —que se refiere a reducir la ingesta calórica sin provocar desnutrición— es un tema que ha sido objeto de numerosos estudios debido a su potencial para mejorar la salud y aumentar la longevidad en varios organismos modelo. Estudios como el de Fontana y Partridge [2] han demostrado que la RD influye en rutas moleculares clave

relacionadas con el envejecimiento, lo que ha generado un gran interés en identificar los genes que están detrás de sus efectos positivos.

Dado que aún no se comprenden completamente los mecanismos moleculares que median en la respuesta a la RD, la priorización genética se presenta como una estrategia valiosa para ayudar a identificar genes candidatos relevantes a partir de la información genómica que ya existe.

1.2 Trabajos previos y motivación

En el contexto de la priorización genética aplicada a la RD, varios estudios han explorado el uso de técnicas de aprendizaje automático para identificar genes relevantes. Entre ellos destaca el trabajo de Vega Magdaleno et al. [1], donde se emplearon fuentes de información como Gene Ontology (GO), Pathway Data Integration Portal (PathDIP) y Genotype-Tissue Expression (GTEx) para construir representaciones de los genes y entrenar clasificadores supervisados.

Si bien los resultados obtenidos fueron prometedores, el enfoque presentaba ciertas limitaciones. Por un lado, asumía que todos los genes sin evidencia experimental podían considerarse negativos, una hipótesis que puede introducir sesgos en el entrenamiento al ignorar posibles falsos negativos. Por otro, no se aplicaban técnicas de reducción de dimensionalidad, lo cual podía afectar tanto al rendimiento como a la interpretabilidad de los modelos, especialmente considerando el elevado número de características por fuente.

1.3 Aportaciones de este trabajo

En este trabajo se plantea una mejora metodológica del enfoque anterior, incorporando dos componentes principales:

- En primer lugar, como **propuesta principal** de este trabajo se ha integrado el algoritmo **Fast-mRMR** (siendo *mRMR* el acrónimo de ”mínima redundancia y máxima relevancia”) para la selección de características, con el objetivo de reducir la dimensionalidad de los datos conservando las características más informativas. Esta técnica, basada en el principio de máxima relevancia y mínima redundancia propuesto por Peng et al. (2005) [3], ha demostrado ser eficaz en contextos bioinformáticos de alta dimensionalidad, especialmente en su versión optimizada presentada por Ramírez-Gallego et al. (2017) [4]. Su incorporación permite mejorar la eficiencia computacional, reducir el riesgo de sobreajuste y facilitar la interpretación del modelo.

- En segundo lugar, como **propuesta complementaria** a la aplicación del algoritmo de Fast-mRMR, se ha incorporado una estrategia basada en **PU Learning** (aprendizaje positivo-no etiquetado), que permite seleccionar ejemplos negativos confiables a partir del conjunto de genes no etiquetados. Esta estrategia se fundamenta en enfoques como los revisados por Bekker y Davis (2020) [5], y se toma como principal referencia la metodología propuesta por Paz-Ruza et al. (2021) [6], aplicada con éxito en tareas de priorización biológica. Su uso evita asumir que todos los ejemplos sin evidencia son negativos, lo que contribuye a reducir el sesgo en el entrenamiento y a mejorar la calidad del modelo final.

1.4 Objetivos del trabajo

El objetivo principal es desarrollar un sistema de priorización genética capaz de identificar genes candidatos relacionados con la restricción dietética, superando las limitaciones detectadas en estudios anteriores.

Los objetivos específicos son los siguientes:

- Aplicar el algoritmo **Fast-mRMR** para seleccionar las características más relevantes, reducir la dimensionalidad y mejorar el rendimiento del sistema.
- Incorporar a la propuesta principal un enfoque de **PU Learning** basado en similitud para seleccionar ejemplos negativos confiables dentro del conjunto no etiquetado.
- Evaluar el sistema mediante métricas estándar de clasificación y ordenación, aplicando validación cruzada anidada para garantizar la robustez de los resultados.
- Comparar el rendimiento del sistema propuesto con los trabajos de referencia principales, empleando tanto los conjuntos de datos de base como versiones combinadas mediante unión características e intersección de genes.

Una vez definidos el contexto, la motivación y los objetivos, en los siguientes capítulos se presentan los fundamentos conceptuales y la metodología que sustentan la propuesta desarrollada. Se introducen las principales técnicas y enfoques utilizados en la literatura para abordar el problema de la priorización genética, así como las bases teóricas del sistema propuesto. Además, se detallan los conjuntos de datos empleados en trabajos previos y los criterios que guiaron el diseño experimental y la implementación del sistema propuesto.

Capítulo 2

Metodología de desarrollo, costes y herramientas

Este capítulo describe la metodología de desarrollo que se siguió para este proyecto, combinando un enfoque iterativo inspirado en **Scrum**, con gestión de tareas basada en **Kanban**. También se incluye una estimación de los costes asociados y una descripción de las herramientas utilizadas.

2.1 Metodología de desarrollo

La metodología de trabajo adoptada combina principios de **Scrum**, con sus iteraciones claramente definidas, y la agilidad visual de **Kanban**, usada para gestionar las tareas de cada fase. Scrum permite dividir el proyecto en ciclos o iteraciones que incluyen fases de planificación, diseño, implementación, evaluación de resultados y replanteamiento. Esto ha sido especialmente útil en este TFG, dado su carácter de investigación y la necesidad de adaptarse a nuevos hallazgos y validaciones constantes en función de los resultados obtenidos en la experimentación. Por su parte, el sistema Kanban, implementado con la herramienta **Trello**, permitió organizar las tareas en tres columnas: *Lista de tareas por hacer*, *En proceso* y *Hechas*, facilitando la trazabilidad del trabajo. El proyecto se desarrolló a lo largo de ocho iteraciones principales, que se describen a continuación:

Iteración 1 – Revisión teórica y análisis del problema. Esta primera fase se centró en el estudio del problema de la priorización genética, incluyendo la revisión de trabajos previos, las limitaciones de los enfoques de los mismos y la justificación del uso de técnicas como Fast-mRMR para la selección de características y la posible aplicación secundaria de selección de instancias con métodos como PU learning.

Iteración 2 – Preparación del entorno de trabajo. Se configuró el entorno experimental en Python para permitir la ejecución modular del pipeline. Esta etapa incluyó la carga y validación de los datos, la definición de flujos de preprocesamiento y evaluación, así como la integración del algoritmo Fast-mRMR implementado en C++ mediante llamadas externas controladas desde Python.

Iteración 3 – Discretización de datos continuos previa al Fast-mRMR. Se adaptaron los conjuntos de datos genéticos que se van a utilizar para la experimentación, GTEx y Coexpression, originalmente con valores continuos, a un formato discreto compatible con Fast-mRMR. Para ello, se aplicaron técnicas de binning como la discretización por cuantiles, dividiendo cada variable en intervalos equifrecuentes. Se evaluaron distintas configuraciones para preservar la estructura informativa original sin introducir sesgo excesivo.

Iteración 4 – Ejecución de Fast-mRMR. Se aplicó el algoritmo de selección de características Fast-mRMR sobre cada conjunto de datos, obteniendo un ranking de genes en función de la relevancia de sus características. A partir de estos rankings, se entrenaron clasificadores supervisados (en nuestro caso, BRF y CAT) y se evaluó el rendimiento mediante métricas de clasificación y ranking.

Iteración 5 – Bagging sobre datos complejos. En conjuntos complejos se empleó una estrategia de bagging de características para dividir el espacio de entrada y reducir la carga computacional, permitiendo la aplicación de Fast-mRMR sobre subconjuntos manejables.

Iteración 6 – Evaluación en conjuntos combinados y complejos. Se compararon resultados entre conjuntos individuales (GO, PathDIP), combinaciones (unión de características/intersección de genes) y conjuntos de alta dimensionalidad, observando mejoras al aplicar selección.

Iteración 7 – Aplicación de Fast-mRMR + PU Learning. Se integraron ambas técnicas para evaluar si la selección de características previa al etiquetado de instancias, PU Learning, ofrecía mejores resultados. Se realizaron comparativas detalladas por conjunto y clasificador.

Iteración 8 – Redacción de la memoria. En esta fase se organizó toda la información generada durante el proyecto, redactando los distintos capítulos de la memoria de manera coherente y estructurada. Se integraron los resultados experimentales, análisis comparativos, figuras explicativas, tablas de métricas y referencias bibliográficas.

2.2 Estimación temporal y de costes

En esta sección se presenta, en primer lugar, la planificación temporal inicial del proyecto, seguida de la evolución real observada durante su desarrollo. Finalmente, se incluye una estimación de los costes humanos asociados, considerando tanto al estudiante como a los tutores académicos implicados.

2.2.1 Planificación estimada

Al inicio del trabajo se elaboró una planificación preliminar, con el objetivo de estructurar el desarrollo del proyecto en función de los bloques principales que lo componen. Esta estimación contemplaba seis iteraciones, abarcando desde la revisión teórica inicial hasta la redacción final de la memoria.

Cada iteración representaba una fase clave dentro del flujo de trabajo: análisis del problema, preparación del entorno experimental, aplicación del algoritmo de selección de características, evaluación de resultados y documentación. Esta planificación, de carácter orientativo, sirvió como guía temporal y permitió anticipar de cierta forma los recursos y esfuerzos necesarios en cada fase. La distribución inicial de estas iteraciones se representa en el diagrama de Gantt mostrado en la Figura 2.1.

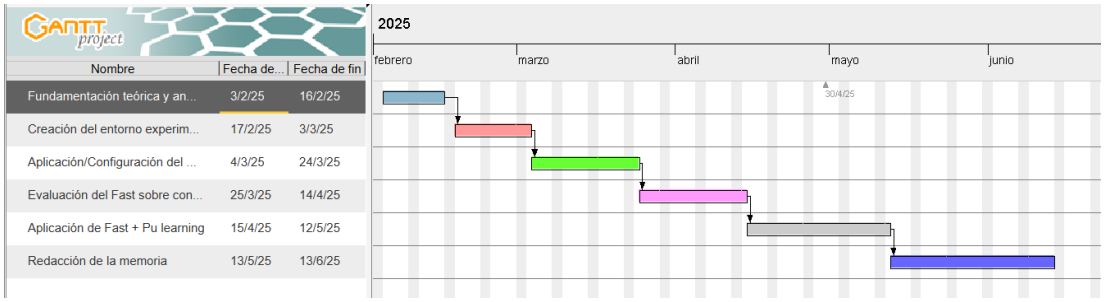


Figura 2.1: Diagrama de Gantt de la estimación inicial de la duración del proyecto.

2.2.2 Duración real del proyecto

Durante el desarrollo surgieron nuevas necesidades no previstas inicialmente, como la discretización previa de datos continuos o la necesidad de aplicar bagging sobre conjuntos complejos como Coexpression. Estas tareas adicionales, junto con una mayor dedicación a las fases anteriores destacando un mayor tiempo de dedicación sobre todo a la fase de redacción de la memoria , implicaron una reorganización y ampliación del calendario. La duración final real se muestra en la Figura 2.2.

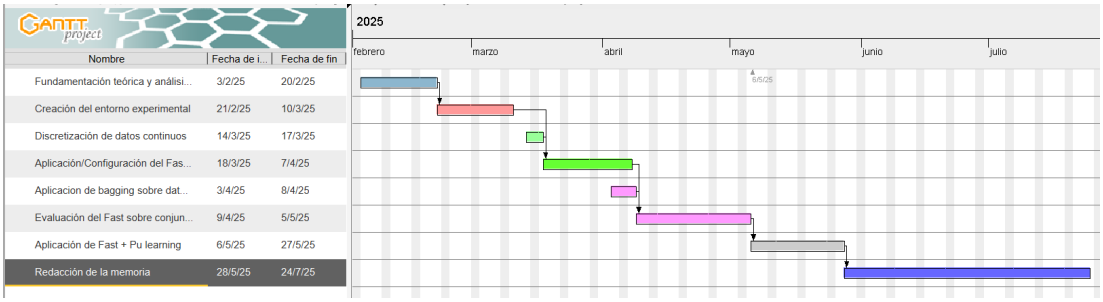


Figura 2.2: Diagrama de Gantt de la duración real del proyecto.

Las iteraciones que más variaron respecto a lo estimado fueron:

- **Iteración de configuración**: extendida por la integración técnica entre Python y C++.
- **Iteración de redacción**: duplicó en duración debido al número de secciones experimentales y su análisis.
- **Dos nuevas tareas**: discretización de datos continuos y bagging sobre Coexpression, necesarias para aplicar Fast-mRMR.

2.2.3 Gestión Kanban durante el desarrollo

En paralelo a la planificación iterativa basada en iteraciones del trabajo, se implementó un sistema de gestión usando la metodología Kanban utilizando la herramienta **Trello**. Este tablero permitió monitorizar el avance del proyecto de forma visual, organizado en tres secciones: *Lista de tareas por hacer*, *En proceso* y *Hechas*, como se muestra en la Figura 2.3.

Figura 2.3: Tablero Kanban utilizado para el seguimiento del proyecto.

Gracias al uso de este sistema, se logró una gestión más ágil del flujo de trabajo en cada iteración. Además, se estableció un esquema de reuniones quincenales con los tutores, en las

que se revisaban las tareas realizadas, se discutía el estado de aquellas en curso y se definían nuevos objetivos en función del progreso. Esta dinámica permitió mantener una comunicación continua y adaptar el enfoque del proyecto a medida que se obtenían nuevos resultados.

2.2.4 Estimación de costes

Para estimar el coste aproximado del proyecto, se ha considerado la dedicación del estudiante y la implicación de los tutores a lo largo del desarrollo del proyecto. Se toma como referencia el salario mensual medio de un estudiante en prácticas en España (aproximadamente 600€ al mes), lo que equivale a unos **7,5€/hora** suponiendo una jornada de 20 horas semanales. En el caso de los tutores, se ha estimado una tarifa de **30€/hora**, en base al coste promedio/hora de un PDI (Personal Docente e Investigador) con dedicación completa en la Universidade da Coruña. El trabajo del estudiante se distribuyó en tres fases diferenciadas, con una intensidad creciente:

- **Del 3 de febrero al 3 de marzo**: dedicación de 1,5 horas diarias, estimando unas 45 horas en total.
- **Del 4 de marzo al 3 de junio**: dedicación de 3 horas diarias, lo que equivale a unas 270 horas.
- **Del 4 de junio al 25 de julio**: dedicación de 4 horas diarias, sumando aproximadamente 200 horas.

Esto da lugar a un total estimado de **515 horas de trabajo por parte del estudiante**. Por su parte, los tutores participaron en reuniones quincenales durante los primeros meses y semanales en la fase final, además de realizar revisiones de la memoria. Se estima un total de **30 horas de dedicación conjunta por parte del equipo de tutores**.

- Salario estimado del estudiante: 515 h × 7,5€ = **3862,5€**
- Salario estimado de los tutores: 30 h × 30€ = **900€**

Cuadro 2.1: Costes estimados del proyecto.

Rol	**Horas**	**Coste total**
Estudiante	515	3862,5 €
Tutores (PDI)	30	900 €
Total estimado	545	**4762,5 €**

No se han considerado costes adicionales por licencias de software, adquisición de hardware o uso de infraestructuras, ya que se emplearon herramientas de código abierto y el acceso remoto a servidores fue proporcionado gratuitamente por el CITIC.

2.3 Herramientas y lenguajes utilizados

Durante el desarrollo del proyecto se emplearon diferentes lenguajes de programación y herramientas que facilitaron tanto la implementación del sistema como la organización y seguimiento del trabajo. A continuación, se detallan los principales recursos utilizados.

2.3.1 Lenguajes de programación

- **Python**: fue el lenguaje principal del proyecto, utilizado para la ejecución del *pipeline* experimental completo, incluyendo la carga y procesamiento de datos, la aplicación de técnicas de selección de características, la validación cruzada, el entrenamiento de modelos y la generación de resultados y visualizaciones.
- **C++**: empleado exclusivamente para la ejecución del algoritmo Fast-mRMR, cuya implementación original se encuentra en este lenguaje. Su integración en el entorno de trabajo se realizó mediante llamadas externas desde Python, permitiendo mantener la coherencia del flujo de experimentación.

2.3.2 Herramientas y entornos

- **Visual Studio Code**: editor de código principal, utilizado para el desarrollo y mantenimiento de los scripts en Python y para organizar el entorno de trabajo de forma modular y clara.
- **Trello**: herramienta de gestión de tareas basada en Kanban, que permitió organizar el trabajo diario en listas de tareas pendientes, en curso y finalizadas, proporcionando una visión general y estructurada del avance por iteración.
- **GanttProject**: software empleado para elaborar la planificación temporal del proyecto mediante diagramas de Gantt, tanto en su versión estimada como real, facilitando la visualización del progreso y los ajustes de calendario.
- **NoMachine**: herramienta de escritorio remoto utilizada para acceder a los recursos computacionales proporcionados por el CITIC, donde se ejecutaron parte de los experimentos y pruebas más costosas computacionalmente.

- **Neptune AI**: plataforma de gestión y seguimiento de experimentos, usada para registrar de forma estructurada los valores de hiperparámetros, las métricas obtenidas en cada iteración y el rendimiento medio tras la validación cruzada.

Capítulo 3

Marco teórico y Trabajos relacionados

Este capítulo se estructura en dos bloques complementarios. En primer lugar, se presentan los fundamentos conceptuales que sustentan la tarea de priorización genética, incluyendo las técnicas de selección de características y el aprendizaje positivo-no etiquetado (*PU Learning*) como selector de instancias. En segundo lugar, se analizan en profundidad los trabajos previos más relevantes en este campo, destacando especialmente los estudios en los que se basa este trabajo, junto con los conjuntos de datos que emplearon.

3.1 Marco teórico

En este apartado se explican los conceptos más importantes sobre los que se apoya el sistema de priorización genética. Se introduce brevemente la priorización genética, sus dificultades y cómo se intenta mejorar su rendimiento usando técnicas de selección de características y manejo de datos no etiquetados.

3.1.1 Priorización genética en biomedicina

La priorización genética es una tarea computacional cuyo objetivo es asignar una puntuación a cada gen dentro de un conjunto de candidatos, con el fin de estimar su probabilidad de estar implicado en un proceso biológico determinado. A partir de estas puntuaciones, se genera un ranking que permite a los investigadores identificar los genes más prometedores para su posterior validación experimental, tal como se describe en el estudio de Yang et al. [7], donde se ofrece una revisión exhaustiva de las herramientas disponibles para esta tarea.

Formalmente, el problema parte de un conjunto G de genes candidatos, donde cada gen $g_i \in G$ se representa mediante un vector de características $x_i \in \mathbb{R}^d$, derivado de distintas

fuentes de información biológica. Estas características o variables pueden incluir, entre otras, datos funcionales, de expresión génica, de interacción molecular o anotaciones semánticas.

Además, se dispone de un subconjunto $P \subset G$ que contiene los ejemplos positivos, es decir, genes con evidencia experimental que los relaciona con el proceso de interés. El resto del conjunto, $U = G \setminus P$, está compuesto por ejemplos no etiquetados: genes para los cuales no se conoce si están o no implicados, pero cuya exclusión como positivos no puede asumirse con certeza.

La priorización genética plantea principalmente dos desafíos: la alta dimensionalidad del conjunto de datos y la escasez de ejemplos negativos explícitos. Por todo ello, se hace necesario aplicar técnicas específicas tanto para seleccionar las características más relevantes como para gestionar de forma adecuada los datos no etiquetados.

3.1.2 Selección de características

La selección de características es una etapa fundamental en muchos sistemas de aprendizaje automático, especialmente cuando se trabaja con datos de alta dimensionalidad, como es el caso en el análisis genómico. Consiste en identificar, dentro de un conjunto de características disponibles, aquellas que son más relevantes para el problema que se quiere resolver, descartando las que resultan irrelevantes o redundantes. Su aplicación permite reducir la complejidad del conjunto de datos, evitar el sobreajuste y mejorar tanto la precisión como la interpretabilidad de los modelos entrenados.

En el contexto de la priorización genética, cada gen se describe mediante cientos o miles de características extraídas de diferentes fuentes biológicas. Esta abundancia de características puede ser problemática si no se filtra correctamente, ya que muchas de ellas pueden no aportar información útil o incluso introducir ruido en el proceso de entrenamiento, empeorando el rendimiento de los modelos. La aplicación de una técnica de selección de características permite centrarse únicamente en aquellos genes que realmente contribuyen a distinguir genes relevantes de los que no lo son.

Según la clasificación propuesta por Kumar y Minz [8], las técnicas de selección de características se pueden agrupar en tres grandes categorías:

- **Métodos filtro (Filter methods)**: evalúan la relevancia de cada característica de forma independiente del modelo de aprendizaje posterior, usando métricas estadísticas como la correlación, la información mutua o la puntuación Chi-cuadrado. Son rápidos y escalables, pero pueden no tener en cuenta interacciones entre características.

- **Métodos envolventes (Wrapper methods)**: seleccionan subconjuntos de características y evalúan su calidad usando directamente un modelo de clasificación y su rendimiento. Suelen obtener mejores resultados, ya que están adaptados al modelo concreto, pero requieren mucho más tiempo de cómputo.
- **Métodos embebidos (Embedded methods)**: integran la selección de características dentro del propio proceso de entrenamiento del modelo (por ejemplo, en árboles de decisión o regularización L1). Logran un equilibrio entre precisión y eficiencia, pero dependen del algoritmo concreto que se utilice.

Cada uno de estos enfoques tiene ventajas e inconvenientes. Los métodos filtro son útiles como paso previo rápido, los envolventes son más precisos, pero costosos, y los embebidos se adaptan bien a modelos concretos, pero no siempre son generalizables.

Se ha optado por un **enfoque de tipo filtro**, al considerar que ofrece el mejor equilibrio entre eficiencia computacional y adecuación al problema planteado. A continuación, se describe en detalle el algoritmo seleccionado.

El algoritmo Fast-mRMR

El algoritmo Fast-mRMR (Fast Minimum Redundancy Maximum Relevance) es una versión optimizada desarrollada por Ramírez-Gallego et al. [4] del clásico algoritmo mRMR, el cual fue diseñado por Peng et al.[3] para seleccionar subconjuntos de características relevantes y poco redundantes en contextos de alta dimensionalidad. Se trata de un método de tipo filtro, basado en medidas de información mutua para evaluar tanto la relevancia de cada característica con respecto a la variable objetivo (es decir, la implicación del gen en el proceso de interés) como su redundancia respecto a otras características seleccionadas.

El algoritmo mRMR original establece dos criterios principales:

- **Máxima relevancia (Max-Relevance)**: se busca seleccionar las características que presenten mayor dependencia estadística con la variable de salida, medida habitualmente mediante la información mutua $I(X;C)$ entre la característica X y la clase C.
- **Mínima redundancia (Min-Redundancy)**: se penaliza la selección de características que estén altamente correlacionadas entre sí, utilizando también la información mutua entre características como medida de redundancia.

La combinación de ambos criterios da lugar a la función objetivo del algoritmo *mRMR*, que selecciona en cada iteración la característica X_i que maximiza:

$$\text{mRMR}(X_i) = I(X_i; C) - \frac{1}{|S|} \sum_{X_j \in S} I(X_i; X_j)$$

donde:

- $I(X_i; C)$ representa la **información mutua** entre la característica candidata X_i y la clase objetivo C, es decir, mide cuán **relevante** es X_i para predecir la clase.
- $\sum_{X_j \in S} I(X_i; X_j)$ calcula la **redundancia media** de X_i respecto al conjunto de características ya seleccionadas S, penalizando la inclusión de información repetida.

En resumen, mRMR busca seleccionar características que sean **altamente informativas con respecto a la clase**, pero a su vez **lo menos redundantes posible entre sí**.

Aunque mRMR ha demostrado ser eficaz en numerosos contextos, su alto coste computacional limita su aplicabilidad cuando el número de características es muy elevado. Para abordar esta limitación, Ramírez-Gallego et al. [4] propusieron Fast-mRMR, una versión optimizada que introduce varias mejoras clave:

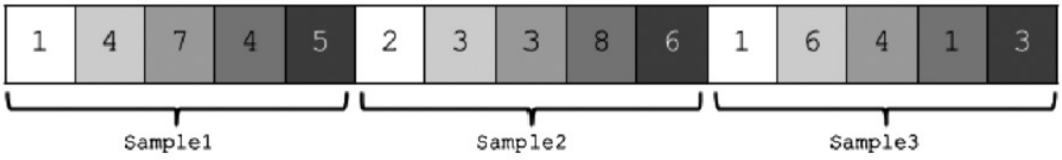
- **Cálculo acumulativo de la redundancia**: se evita calcular la redundancia entre todas las características en cada iteración, acumulando los valores ya obtenidos.
- **Almacenamiento en caché de relevancias e información marginal**: se reduce el número de cálculos redundantes reutilizando resultados intermedios.
- **Optimización del patrón de acceso a los datos**: se reorganiza el acceso a las estructuras de datos para mejorar el rendimiento, especialmente en arquitecturas paralelas como GPUs.

Una de las claves de esta última optimización es la reorganización de la matriz de entrada para **favorecer un acceso orientado por columnas** (*feature-wise*), en lugar de por filas (*sample-wise*). Dado que se trabaja con **datos tabulares**, donde cada columna representa una característica genética y cada fila un gen, esta estructura resulta especialmente eficiente en contextos de alta dimensionalidad, como en los conjuntos utilizados en este trabajo, que contienen miles de características por gen.

Esta disposición no solo mejora el acceso a memoria, sino que también facilita la ejecución de **operaciones matriciales vectorizadas**, optimizando los cálculos cuando el algoritmo se

implementa en lenguajes de bajo nivel como C++. En particular, el acceso por columnas aprovecha mejor la **localidad espacial de memoria** y permite una mayor **paralelización** tanto en CPU como en GPU, lo que se traduce en una mejora sustancial del rendimiento computacional.

La Figura 3.1, extraída del trabajo de Ramírez-Gallego et al. [4], ilustra este cambio estructural: en la parte superior (a), los datos están organizados por genes; mientras que en la parte inferior (b), la reorganización orientada a características permite optimizar el acceso a memoria.



(a) Acceso orientado por instancia

(b) Acceso orientado por característica

Figura 3.1: Comparación entre la estructura de datos original y la reorganizada en Fast-mRMR.

Estas optimizaciones permiten que Fast-mRMR mantenga la eficacia del enfoque original con un coste computacional significativamente menor, lo que lo convierte en una opción especialmente adecuada para escenarios con alta dimensionalidad y con un conjunto muy grande de características, como los conjuntos de datos biomédicos empleados.

3.1.3 Aprendizaje positivo-no etiquetado (PU Learning)

El aprendizaje positivo-no etiquetado (PU Learning) es un enfoque de aprendizaje supervisado que se aplica en contextos donde únicamente se dispone de ejemplos con conocimiento curado (es decir, genes validados como relevantes mediante evidencia experimental) y un conjunto de ejemplos no etiquetados, cuya clase verdadera se desconoce.

A diferencia de la clasificación binaria convencional, en los problemas PU no se cuenta con ejemplos negativos identificados de forma explícita, lo cual plantea desafíos específicos

en la construcción de modelos predictivos. Este tipo de situación es común en áreas como la biología computacional o la medicina, donde la validación experimental es costosa y el conocimiento existente sobre el fenómeno de estudio suele estar incompleto. En el caso de la priorización genética, se parte de un subconjunto reducido de genes con evidencia experimental (positivos), mientras que el resto del genoma carece de anotación clara, y no puede asumirse que todos los genes no etiquetados sean realmente negativos.

Bekker y Davis [5] presentan una revisión detallada de las principales estrategias desarrolladas en el ámbito del aprendizaje positivo-no etiquetado, analizando sus fundamentos teóricos y la forma en que abordan la gestión del conjunto no etiquetado. A partir de este análisis, los autores distinguen tres enfoques principales de aprendizaje positivo-no etiquetado:

- **Enfoques de dos pasos (two-step)**: consisten en una primera fase en la que se seleccionan ejemplos negativos confiables dentro del conjunto no etiquetado, seguida de una segunda fase en la que se entrena un clasificador binario tradicional con los ejemplos positivos y los negativos identificados.
- **Aprendizaje sesgado (biased learning)**: estos métodos consideran el conjunto no etiquetado como negativo durante el entrenamiento, pero corrigen el sesgo introducido mediante técnicas de ponderación o regularización.
- **Estimación del prior de clase**: en este enfoque se estima la proporción de ejemplos positivos en el conjunto no etiquetado, y esta información se utiliza para ajustar las predicciones del modelo.

En este trabajo se ha adoptado un enfoque de tipo ***two-step***, en el cual se aplica un procedimiento de selección de genes negativos basado en similitud, siguiendo la estrategia propuesta por Paz-Ruza et al. [6]. La idea central consiste en identificar, dentro del conjunto no etiquetado U, aquellos genes que presentan baja similitud con los ejemplos positivos conocidos P, considerándolos como negativos confiables $\mathcal{RN}$.

Sea x_i el vector de características de un gen $g_i \in U$, y $N_k(x_i)$ el conjunto de sus k vecinos más cercanos en el espacio de características, calculado mediante una métrica de similitud como el índice de Jaccard. Un ejemplo x_i se considera confiablemente negativo si:

$$\frac{|\{x_j \in N_k(x_i) \mid x_j \in U\}|}{k} \geq t \quad \text{y} \quad \arg\max_{x_j \in N_k(x_i)} sim(x_i, x_j) \in U$$

donde t es un umbral de tolerancia (por ejemplo, $t = 0.8$), y $sim(x_i, x_j)$ es la función de similitud entre genes. Es decir, la mayoría de los vecinos más cercanos de x_i deben ser no

etiquetados, y su vecino más similar no debe pertenecer a P.

El conjunto final de negativos confiables $\mathcal{RN}$ se define como:

$$\mathcal{RN} = \{x_i \in U \mid x_i \text{ cumple los criterios anteriores}\}$$

Una vez identificado $\mathcal{RN}$, se construye el conjunto de entrenamiento binario como $P \cup \mathcal{RN}$, etiquetando P como clase positiva y $\mathcal{RN}$ como negativa. Esto permite entrenar modelos discriminativos sin recurrir a supuestos fuertes sobre la distribución real de clases en U.
La Figura 3.2 resume gráficamente este enfoque de dos fases. A partir del conjunto de ejemplos positivos (P) y del conjunto no etiquetado (U), se selecciona un subconjunto de negativos confiables ($\mathcal{RN}$), con el que se construye un conjunto de entrenamiento binario. Esta estrategia ha sido adaptada del trabajo original de Paz-Ruza et al. [6].

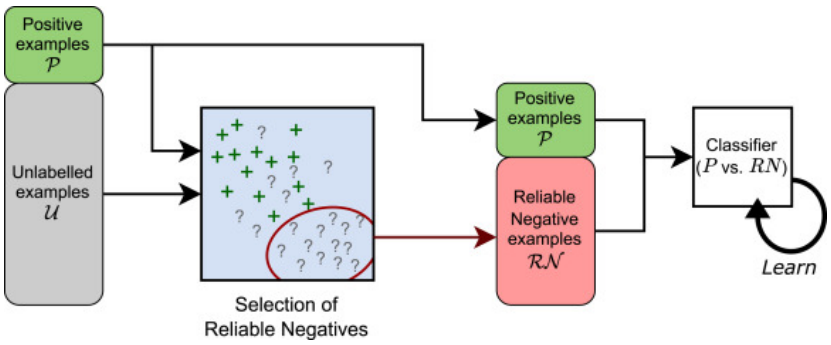


Figura 3.2: Esquema del enfoque de aprendizaje positivo-no etiquetado basado en selección de negativos confiables.

Al incorporar esta técnica, se mejora la calidad del conjunto de entrenamiento sin introducir sesgos innecesarios, permitiendo así construir modelos más robustos y realistas en contextos con anotaciones escasas e incompletas, como ocurre en la priorización genética.

3.2 Trabajos relacionados

Para poder desarrollar este proyecto se han revisado varios estudios anteriores que han aplicado aprendizaje automático a la priorización genética, especialmente en el contexto de la restricción dietética. A continuación, se explican brevemente los más importantes y cómo han influido en el desarrollo del TFG. Cabe destacar que, en este trabajo, cada instancia del conjunto de datos corresponde a un gen, representado mediante un vector de características extraídas de diversas fuentes biológicas. Por tanto, los términos "instancia" y "gen" se utilizarán de forma equivalente a lo largo del texto.

3.2.1 Asociaciones restricción-envejecimiento con aprendizaje automático

Uno de los estudios de referencia en el ámbito de la priorización genética aplicada a la restricción dietética es el desarrollado por Vega Magdaleno et al. [1]. En dicho estudio, los autores plantearon un sistema basado en técnicas de aprendizaje automático con el objetivo de identificar genes potencialmente asociados a la RD, partiendo de un conjunto inicial de genes vinculados previamente al proceso de envejecimiento.

El pipeline experimental desarrollado en dicho trabajo se estructuraba en varias fases claramente diferenciadas:

- **Selección de fuentes de datos**: se utilizaron múltiples bases de información biológica relevantes, incluyendo Gene Ontology (GO) [9], PathDIP [10], GTEx [11], KEGG [12] y datos de interacción proteína-proteína (PPI) [13].
- **Representación de los genes**: a partir de cada fuente, los genes fueron descritos mediante estructuras de datos en formato tabular, donde cada instancia correspondía a un gen y cada columna representaba una característica derivada de la fuente considerada (por ejemplo, términos GO, rutas PathDIP, rutas KEGG, interacciones proteína-proteína, datos de expresión GTEx o descriptores de secuencia)
- **Entrenamiento y evaluación de modelos**: se entrenaron distintos clasificadores supervisados utilizando los datos obtenidos, aplicando validación cruzada y evaluando su rendimiento mediante métricas de clasificación y ordenación.

Los resultados obtenidos indicaron que las características derivadas de fuentes de conocimiento curado, como los términos de Gene Ontology (GO) y las rutas biológicas integradas en PathDIP, presentaban un mayor poder predictivo. En comparación, las características extraídas de datos más propensos al ruido, como los perfiles de expresión génica, ofrecieron un rendimiento inferior en los modelos evaluados.

Sin embargo, el estudio presentaba algunas limitaciones relevantes. En primer lugar, asumía que todos los genes sin evidencia experimental podían considerarse negativos durante el entrenamiento, lo que podía introducir sesgos en los modelos y limitar la capacidad de descubrir genes realmente relevantes, pero aún no identificados. En segundo lugar, no se aplicó ninguna técnica de reducción de dimensionalidad, lo cual podía afectar al rendimiento y la interpretabilidad de los modelos, especialmente teniendo en cuenta el elevado número de características utilizadas por fuente.

En este trabajo de fin de grado se han reutilizado varios de los conjuntos de datos generados en ese estudio, especialmente los basados en GO y PathDIP, como base para construir y evaluar el nuevo sistema. A partir de esta base, se han introducido mejoras metodológicas para abordar las limitaciones detectadas, como la integración del algoritmo Fast-mRMR para la selección de características y la incorporación adicional como propuesta secundaria de una estrategia de aprendizaje positivo-no etiquetado para refinar la selección de ejemplos negativos.

El objetivo de estas mejoras es doble: por un lado, incrementar la interpretabilidad de los resultados al reducir la complejidad del modelo mediante una selección más informativa de características; y por otro, mejorar el rendimiento predictivo del sistema en el contexto de la priorización genética.

3.2.2 Aprendizaje Positivo y Sin Etiquetas para descubrir genes de restricción dietética relacionados con el envejecimiento

El trabajo de Paz-Ruza et al. [6] presenta una metodología específica de aprendizaje positivo-no etiquetado (PU Learning) orientada a resolver una de las principales limitaciones detectadas en enfoques anteriores de priorización genética: la asunción de que todos los genes sin evidencia experimental pueden ser tratados como negativos. Esta hipótesis, utilizada por ejemplo en el estudio de Vega Magdaleno et al. [1], introduce un sesgo significativo en el proceso de aprendizaje, al no considerar que el conjunto no etiquetado puede contener genes relevantes aún no identificados.

Para superar este problema, Paz-Ruza et al. proponen una estrategia basada en similitud que permite identificar un subconjunto de ejemplos negativos confiables dentro del conjunto no etiquetado, sin necesidad de etiquetarlo completamente. El enfoque se encuadra dentro del paradigma de aprendizaje positivo-no etiquetado en dos pasos que explicamos en el apartado anterior.

El método implementado emplea una variante del algoritmo de los vecinos más cercanos (KNN), utilizando como métrica de comparación el índice de Jaccard. Un gen no etiquetado es considerado confiablemente negativo si cumple dos condiciones:

1. Su vecino más similar (según Jaccard) también es un gen no etiquetado.
2. Al menos una proporción t de sus k vecinos más cercanos tampoco están presentes en el conjunto de positivos.

Este procedimiento permite generar un subconjunto $\mathcal{RN}$ de ejemplos negativos con bajo riesgo de contener falsos negativos. Al reducir el ruido en el conjunto de entrenamiento, se

mejora la robustez del modelo y se evita penalizar injustamente a genes potencialmente implicados en el proceso de estudio.

Esta técnica se ha incorporado como parte complementaria del pipeline de priorización genética, contribuyendo directamente a corregir la limitación identificada en el enfoque original. Su implementación permite construir un conjunto de entrenamiento más fiable y realista, mejorando así la capacidad del sistema para detectar nuevos genes candidatos relacionados con la restricción dietética.

3.2.3 Conjunto de datos utilizados en los trabajos de referencia

Para la construcción del sistema de priorización genética en el contexto de la restricción dietética (RD), se emplearon conjuntos de datos previamente utilizados por Vega Magdaleno et al. [1], compuestos por genes humanos vinculados al envejecimiento (*GenAge*) y a intervenciones nutricionales como la restricción dietética (*GenDR*). La intersección entre ambas fuentes define el conjunto positivo, mientras que los genes restantes se consideran no etiquetados.

Cada conjunto de datos presenta una estructura tabular en la que cada fila representa un gen distinto. La primera columna contiene el identificador del gen, mientras que las siguientes columnas recogen diferentes características asociadas a ese gen, derivadas de fuentes biológicas como ontologías funcionales, rutas metabólicas, perfiles de expresión o redes de interacción. Finalmente, la última columna, denominada *Class*, indica si el gen está relacionado con la restricción calórica (etiquetado como ***CR***) o si no se dispone de evidencia de dicha relación (etiquetado como ***Not CR***).

Cuadro 3.1: Resumen conceptual de los conjuntos de datos utilizados en los trabajos originales

Fuente	**N.º genes**	**N.º características**	**Tipo de datos**
GO	1,124	8,640	Variables binarias (asociación a términos de Biological Process en Gene Ontology, incluyendo ancestros jerárquicos).
PathDIP	986	1,640	Variables binarias (participación en vías biológicas integradas como Reactome, KEGG y WikiPathways).
GTEx	1,111	55	Valores continuos (expresión génica mediana en 55 tejidos humanos, versión GTEx v8).
PPI (adyacencia)	850	5,718	Variables binarias (matriz de adyacencia de interacciones físicas entre proteínas, basada en BioGRID).
Coexpression	1,048	44,946	Valores continuos (perfiles de correlación de expresión génica generados con GeneFriends).

Estos conjuntos de datos constituyen la base sobre la cual se construyeron todas las configuraciones experimentales evaluadas. Los detalles relativos al preprocesamiento, las técnicas de discretización aplicadas y la representación final de los datos utilizados en el pipeline se desarrollan en la Sección 5.1.

Es relevante destacar que, aunque las características disponibles varían en función de la fuente de datos utilizada, **el conjunto de genes representado se mantiene prácticamente constante en todos los casos**. Esta circunstancia no es habitual en estudios biomédicos, donde es más común que los distintos conjuntos difieran también en los genes representados. En este caso, contar con múltiples descripciones sobre el mismo grupo de genes permite llevar a cabo un análisis más equilibrado y comparable entre las distintas fuentes, evaluando de forma más precisa qué tipo de información resulta más útil para la tarea de priorización.

Capítulo 4

Método Propuesto

Basándose en los conceptos introducidos en el Capítulo 3.1, esta sección presenta el método propuesto de priorización genética en el contexto de restricción dietética, el cual combina diversas técnicas de aprendizaje automático para afrontar dos desafíos clave : la alta dimensionalidad del espacio de características como reto principal a resolver y la ausencia de ejemplos negativos confiables. Estos problemas han sido documentados en trabajos como el de Vega Magdaleno et al. (2022) [1], centrado en priorización génica con restricciones dietéticas, así como en las revisiones de Yang et al. (2018) [7], que analizan los principales retos del aprendizaje automático en biología funcional.

4.1 Visión General del Pipeline

El sistema propuesto se organiza como un *pipeline* modular compuesto por distintas fases secuenciales, diseñadas para abordar la maldición de la alta dimensionalidad de los datos. Cada fase cumple una función específica dentro del flujo de priorización, desde la selección de características hasta la evaluación final del modelo. A continuación, se describe brevemente cada una de estas etapas:

1. **Selección de características con Fast-mRMR**: se aplica un método filtro basado en información mutua que selecciona características con alta relevancia respecto a la clase y baja redundancia entre sí. El algoritmo requiere una representación binaria en formato *.mrmr* como entrada, optimizada para su ejecución eficiente. Esta fase permite reducir significativamente la dimensionalidad y aumentar el rendimiento del modelo.

2. **Modelos de clasificación**: se utilizan dos clasificadores con enfoques complementarios: Balanced Random Forest (BRF), basado en árboles de decisión y especialmente efectivo en escenarios con múltiples características, y CatBoost, un modelo de gradiente boosting eficiente con buen comportamiento en conjuntos con variables categóricas

o heterogéneas. Ambos modelos se entrenan a partir del subconjunto de características seleccionadas en cada iteración, y su rendimiento se evalúa siguiendo un esquema jerárquico de validación cruzada.

3. **Generación del ranking de genes**: se ordenan los genes no etiquetados en función de la probabilidad estimada de pertenecer a la clase positiva, permitiendo priorizar los candidatos más prometedores, gracias a la aplicación previa del algoritmo de **Fast-mRMR** se obtiene un ranking más preciso y mejor con respecto al no aplicar esta técnica.

4. **Evaluación mediante métricas de clasificación y ordenación**: a partir de las predicciones obtenidas, se calculan métricas cuantitativas que permiten evaluar el rendimiento del sistema. En concreto, se calculan métricas de clasificación como: F1-score, AUC-ROC, AUC-PR y G-Mean, así como métricas específicas de ordenación como: NDCG@10, Precision@10 y Recall@10, orientadas a valorar la calidad del ranking generado, las cuales se explican en la Sección 5.6.

En la Figura 4.1 se muestra una representación visual del pipeline. En ella, A1, A2, etc., representan las distintas características funcionales utilizadas para describir cada gen.

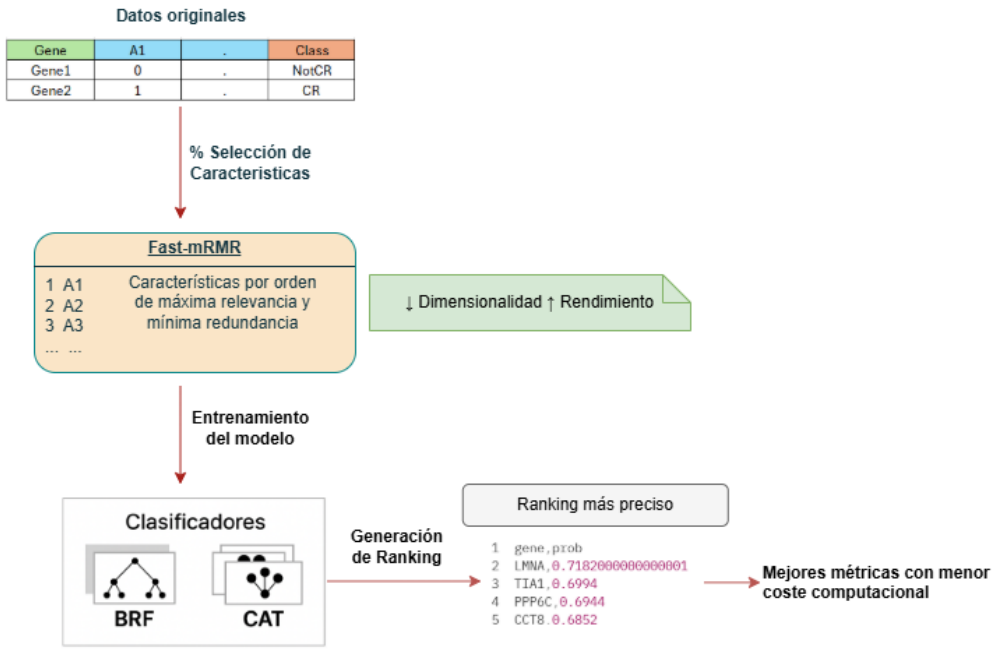


Figura 4.1: Esquema general del pipeline propuesto. Se ilustra el flujo desde los datos originales hasta el cálculo de las métricas para evaluar el ranking, destacando la reducción de dimensionalidad y la mejora del rendimiento.

4.2 Fase de Reducción de Dimensionalidad con Fast-mRMR

Tal como se introdujo en la Sección 3.1.2, el algoritmo *Fast-mRMR* permite seleccionar subconjuntos de características informativas mediante la maximización de la relevancia con la clase y la minimización de la redundancia interna, empleando información mutua como criterio base. En este trabajo se aplica como mecanismo de reducción dimensional previo al entrenamiento, con el objetivo de mejorar tanto la eficiencia como la interpretabilidad de los modelos.

La implementación concreta se basa en la versión optimizada propuesta por Ramírez-Gallego et al. [4], adaptada para su ejecución eficiente sobre conjuntos de datos con miles de características.

4.2.1 Preparación de los datos para Fast-mRMR

Antes de ejecutar el algoritmo, se realiza un preprocesamiento estructurado que garantiza la compatibilidad con la implementación en C++:

- **Discretización**: En el caso de características continuas, se aplica de manera manual una transformación basada en particiones por cuantiles, dividiendo cada variable en 10 intervalos con el mismo número de instancias (estrategia uniforme por cuantiles). Este paso convierte los valores reales en categóricos para el cálculo posterior de información mutua. Si las variables ya son binarias o categóricas, no se aplica ninguna transformación.

- **Conversión al formato binario *.mrmr***: La tabla resultante se reestructura siguiendo un orden *column-major*,lo cual significa almacenar los valores secuencialmente por columnas en lugar de por filas debido a la alta dimensionalidad de características – esta mejora ya fue descrita en el capítulo 3.1.2. Esta organización permite al ejecutable C++ acceder a las características de forma eficiente. Además, los datos se codifican en formato binario utilizando enteros sin signo de 8 bits (*uint8*), excluyendo las columnas de *Gene* y *Class*.

  Para adaptarse al formato requerido por el ejecutable en C++ de Fast-mRMR, la columna de clase (*Class*) se coloca en primer lugar, y sus valores se transforman: los genes etiquetados como *CR* se codifican como *1*, mientras que los *Not CR* se codifican como *-1*. Esta reorganización asegura que el archivo resultante sea compatible con la entrada esperada por el algoritmo.

4.2.2 Ejecución del algoritmo de Fast mRMR

Esta versión optimizada está específicamente orientada a reducir los elevados costes computacionales asociados al cálculo de la información mutua en contextos con un gran número de características. En este trabajo se emplea la implementación original usada de referencia en C++ aprovechando las ventajas que nos brinda este lenguaje con respecto al algoritmo original mRMR implementado en Python. Fast mRMR se integra mediante la ejecución de subprocesos desde Python. Esta llamada externa permite especificar parámetros como el número de características deseadas (m) y la ruta del archivo de entrada, previamente transformado al formato binario `.mrmr` (véase Sección 4.2.1).

Una vez iniciada la ejecución, el algoritmo sigue una estrategia dividida en tres etapas principales, reflejadas en el pseudocódigo de la Figura 1:

1. **Cálculo de relevancia:** se evalúa la información mutua entre cada característica candidata f y la clase c, y se almacena su valor para evitar recomputaciones posteriores. Esta operación se realiza una única vez por característica.

2. **Selección inicial:** se identifica la característica con mayor relevancia respecto a la clase. Esta pasa a formar parte del conjunto de seleccionadas S y servirá como referencia para los cálculos de redundancia siguientes.

3. **Selección iterativa:** en cada iteración, el algoritmo evalúa la redundancia de las características no seleccionadas con respecto a las ya elegidas. Se acumulan los valores de información mutua y se calcula la puntuación mRMR como se describe en la fórmula que aparece en la sección 3.1.2.

Este procedimiento equilibra relevancia y diversidad en la selección, maximizando la información útil y evitando la presencia de atributos redundantes.

Algorithm 1 Selección de características mediante *Fast-mRMR*

Require: Dataset D, conjunto de características F, clase c, número de características deseadas m

Ensure: Subconjunto $S \subseteq F$ de tamaño m (características seleccionadas)

1: Discretizar (si aplica) y convertir a formato binario `.mrmr`
2: **for all** $f \in F$ **do**
3: Calcular $I(f; c)$ ▷ Relevancia entre la característica f y la clase c
4: $Red(f) \leftarrow 0$ ▷ Inicializar la redundancia acumulada de f
5: **end for**
6: $S \leftarrow \{\arg\max_{f \in F} I(f; c)\}$ ▷ Seleccionar la característica más relevante
7: **while** $|S| < m$ **do**
8: **for all** $f \in F \setminus S$ **do**
9: $Red(f) \leftarrow Red(f) + I(f; s_{\text{último}})$ ▷ Actualizar redundancia
10: $mRMR(f) \leftarrow I(f; c) - \frac{Red(f)}{|S|}$ ▷ Calcular puntuación mRMR
11: **end for**
12: $f^* \leftarrow \arg\max_{f \in F \setminus S} mRMR(f)$ ▷ Seleccionar la mejor puntuada
13: $S \leftarrow S \cup \{f^*\}$ ▷ Añadirla al conjunto de seleccionadas
14: **end while**
15: **return** S ▷ Devolver las m características más relevantes y no redundantes

Cabe destacar que el conjunto S representa el subconjunto ordenado de características seleccionadas más relevantes, construido de manera iterativa a partir del equilibrio entre relevancia con la clase y redundancia con respecto a las características ya seleccionadas. El tamaño final de S está definido por el parámetro m, que indica cuántas características se desean conservar.

Gracias a su estructura iterativa y al almacenamiento eficiente de los cálculos intermedios, esta versión de mRMR resulta especialmente adecuada para contextos como el abordado en este trabajo, donde los conjuntos de datos presentan una elevada dimensionalidad. Su integración como paso previo a la clasificación permite reducir la complejidad del modelo y mejorar su capacidad de generalización sin introducir costes computacionales adicionales significativos.

Robustez de ejecución

Para garantizar la fiabilidad del proceso, se incorpora un sistema de control de errores: si la ejecución del algoritmo falla por problemas de formato, lectura o salida vacía, se lanza automáticamente un nuevo intento tras limpiar los archivos temporales. Esta medida asegura

la estabilidad del sistema incluso en entornos con alta carga o configuraciones atípicas.

4.3 Combinación de conjuntos de datos

Este trabajo analiza también la combinación de distintos conjuntos de datos que ofrecen información complementaria sobre los genes (como funciones biológicas, rutas o niveles de expresión). Si bien en estudios previos, como el de G. D. Vega Magdaleno [1], este enfoque no produjo mejoras destacables, aquí se retoma desde una perspectiva diferente. En particular, se aprovecha que muchos genes aparecen simultáneamente en varios conjuntos, lo que permite fusionar sus características de forma más coherente y sin pérdida de información.

Se exploran dos estrategias principales:

- **Intersección de genes con unión de características:** A partir de los genes comunes que aparecen en todos los conjuntos de datos utilizados, se construye una única estructura combinando todas las columnas de características disponibles. De este modo, cada gen queda descrito con mayor riqueza de información, sin introducir valores ausentes.
- **Intersección de genes con análisis independiente por conjunto:** Los genes comunes se analizan por separado en cada conjunto de características, aplicando un pipeline completo e independiente sobre cada conjunto. Esto facilita la comparación entre las diversas fuentes y con respecto al primer caso del conjunto formado por intersección de genes y unión de características.

A continuación, se representan de forma visual en la Figura 4.2 las dos estrategias mencionadas para la combinación de características.

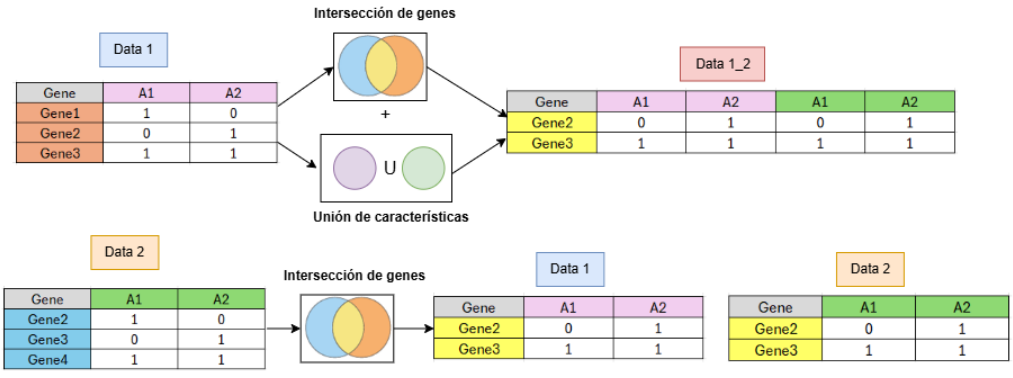


Figura 4.2: Estrategias para combinar conjuntos de datos: Intersección de genes y unión de características (arriba) y análisis independiente de características con genes comunes (abajo).

Estas pruebas se incorporan como una extensión al método propuesto, motivadas por una situación poco habitual en el ámbito biomédico: la mayoría de los genes están presentes en varios conjuntos de datos, aunque cada uno aporta tipos de características diferentes. En estudios similares, lo habitual es encontrarse con datasets que difieren tanto en genes como en los descriptores. Por ello, se aprovechó esta coincidencia poco común para explorar combinaciones que no suelen ser lo habitual en este tipo de investigaciones.

4.4 Implementaciones complementarias al método propuesto

Además de las fases principales que componen el pipeline general, se han incorporado dos estrategias complementarias que fortalecen el rendimiento del sistema en contextos especialmente desafiantes. Estas aportaciones no forman parte del método propuesto original, pero resultan fundamentales en escenarios con alta dimensionalidad o ausencia de ejemplos negativos confiables.

En particular, se detalla a continuación el uso de un esquema de partición en subespacios como preprocesamiento previo a la selección de características, así como una estrategia basada en vecindad para la identificación de ejemplos negativos fiables bajo el paradigma de PU Learning.

4.4.1 Bagging en subespacios de características

En conjuntos de datos con una dimensionalidad extremadamente alta (como *Coexpression*), se aplica una partición disjunta del espacio de características con el objetivo de reducir el coste computacional asociado a la selección. Esta estrategia se introduce antes del uso del algoritmo *Fast-mRMR*, cuyo coste computacional escala de forma cuadrática con respecto al número de características, debido al cálculo de redundancia entre pares de características. En contextos con decenas de miles de características por gen, esta operación se vuelve especialmente costosa tanto en tiempo de ejecución como en consumo de memoria.

Este problema es particularmente evidente en el conjunto *Coexpression*, que contiene más de 44.000 características. La aplicación directa de Fast-mRMR sobre este espacio completo no es factible por las restricciones computacionales. Por este motivo, se introduce una estrategia basada en el principio de *feature bagging*, que permite dividir el conjunto completo de variables en bloques independientes y tratarlos de forma separada. De esta forma, se reduce de forma significativa la complejidad del proceso de selección, haciéndolo viable incluso en contextos de muy alta dimensionalidad.

A diferencia de versiones clásicas del *feature bagging* [14], esta implementación no utiliza muestreo con reemplazo ni agregación de predicciones, ya que el objetivo no es construir un ensamblado de modelos, sino reducir el espacio de características de forma previa al entrenamiento. Se adopta además una variante determinista de la técnica, donde los subespacios son disjuntos y bien definidos, lo que mejora la interpretabilidad y reproducibilidad del proceso.

La elección de parámetros como el número de bloques o el porcentaje de características seleccionadas por bloque se realizó tomando como referencia otros conjuntos manejables como GO. Por ejemplo, se buscó que el conjunto final tras la reducción se situase en una escala similar (alrededor de 8.000–9.000 características), permitiendo así que Fast-mRMR pudiera ejecutarse posteriormente de forma directa sin fragmentación adicional.

La Figura 4.3 muestra de forma visual el proceso seguido: a partir del conjunto original de datos (genes × características), se realiza una división en bloques disjuntos. Cada bloque es evaluado por separado mediante *Fast-mRMR*, seleccionando las características más relevantes en cada caso. Posteriormente, las características seleccionadas se combinan para conformar el conjunto final reducido.

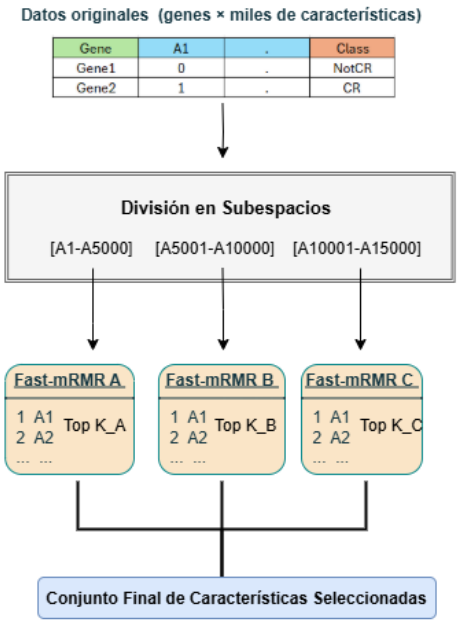


Figura 4.3: Esquema del proceso de partición en subespacios y selección de características mediante Fast-mRMR.

Este enfoque se inspira en trabajos como el de Wu et al. [14], donde se evalúa el *bagging* en subespacios para selección en alta dimensionalidad, y en la propuesta original de Fern y Brodley [15], centrada en mejorar la diversidad y estabilidad de los modelos mediante proyecciones aleatorias. En el presente trabajo, sin embargo, el uso de esta técnica se adapta con un enfoque funcional y computacionalmente eficiente, orientado a garantizar la viabilidad del sistema ante conjuntos de datos de gran escala como Coexpression.

4.4.2 Identificación de negativos fiables con PU Learning

Como se ha indicado previamente, PU Learning se introduce como una fase complementaria en el pipeline, actuando a nivel de genes sobre el conjunto no etiquetado tras la selección de características realizada por *Fast-mRMR*. Su objetivo es reducir el ruido presente en U (genes no etiquetados), identificando ejemplos que puedan considerarse negativos con un alto grado de confianza.

El método implementado sigue un enfoque *two-step*, basado en la vecindad de los ejemplos positivos en el espacio de características. Se parte de la hipótesis de que los genes positivos tienden a agruparse, por lo que un gen no etiquetado que se encuentra rodeado mayoritariamente por otros no etiquetados y alejado de los positivos puede considerarse un negativo fiable.

Similitud y selección vecinal

Dado que las representaciones de los genes son **binarias y dispersas**, se utiliza el **índice de Jaccard** para medir la similitud entre instancias. Esta métrica es especialmente adecuada en este contexto, ya que se enfoca en la presencia conjunta de características relevantes, penalizando coincidencias triviales debidas a la ausencia simultánea de características:

$$J(x_i, x_j) = \frac{\sum_k x_i[k] \cdot x_j[k]}{\sum_k x_i[k] + x_j[k] - x_i[k] \cdot x_j[k]}$$

donde $x_i[k]$ denota si la característica k está presente en el gen x_i.

El proceso de selección de ejemplos negativos confiables se basa en las siguientes dos condiciones:

1. Al menos una proporción t de los k vecinos más cercanos de x_i (según Jaccard) deben pertenecer también al conjunto no etiquetado U.

2. El vecino más similar de x_i no debe pertenecer al conjunto positivo P.

Solo los genes no etiquetados $x_i \in U$ que cumplen ambas condiciones son considerados **negativos confiables**, y se añaden al conjunto $\mathcal{RN}$. A continuación, se construye un conjunto de entrenamiento binario compuesto por P (etiquetado como positivo) y $\mathcal{RN}$ (etiquetado como negativo), sobre el cual se entrena un clasificador convencional.

Ejecución del algoritmo

El procedimiento completo se describe en el Algoritmo 2 basado en el algoritmo original de Jorge Paz [6]:

Algorithm 2 Selección de negativos confiables por vecindad

Require: Conjuntos P (positivos), U (no etiquetados); parámetros k, t; subconjunto de características F

Ensure: Conjunto $\mathcal{RN} \subseteq U$ de negativos confiables

$D \leftarrow P \cup U$ ▷ Unir conjuntos para calcular similitud
Calcular matriz de similitud S sobre D usando índice de Jaccard con F
$\mathcal{RN} \leftarrow \emptyset$ ▷ Inicializar conjunto de negativos confiables
for all $x_i \in U$ **do**
    $T_k \leftarrow k$ vecinos más similares a x_i según S
    $x_{\text{max\_sim}} \leftarrow \arg\max_{x_j \in D \setminus \{x_i\}} S_{ij}$ ▷ Vecino más similar
    if $\frac{|T_k \cap U|}{k} \geq t$ **y** $x_{\text{max\_sim}} \in U$ **then**
        $\mathcal{RN} \leftarrow \mathcal{RN} \cup \{x_i\}$ ▷ Añadir como negativo fiable
    end if
end for
return $\mathcal{RN}$

Este procedimiento permite reducir el ruido en U sin introducir etiquetas manuales, aprovechando únicamente información de similitud local. La precisión del método depende de una adecuada elección de los parámetros k y t, los cuales se ajustan posteriormente mediante validación cruzada jerárquica.

4.5 Esquema de validación cruzada jerárquica

Con el objetivo de evaluar de forma robusta el rendimiento del sistema propuesto, se implementa una estrategia de validación cruzada jerárquica. Esta estructura anidada permite ajustar los hiperparámetros del modelo sin incurrir en sesgos de sobreajuste, y facilita una estimación fiable del rendimiento sobre datos no vistos.

El esquema admite dos configuraciones: una basada únicamente en selección de características mediante *Fast-mRMR*, y otra que incorpora de forma complementaria la técnica *PU Learning* para identificar ejemplos negativos fiables dentro del conjunto no etiquetado.

Cabe destacar que, incluso cuando se activa la opción de PU Learning **todo el proceso se integra en una única validación cruzada jerárquica**, donde *Fast-mRMR* actúa sobre el espacio de características y *PU Learning* sobre el espacio de genes. No se realizan validaciones independientes para cada técnica, sino que ambas se encadenan dentro del mismo ciclo de entrenamiento y evaluación.

4.5.1 Estructura general y justificación metodológica

La Figura 4.4 muestra una representación esquemática del enfoque de validación cruzada jerárquica utilizado en este trabajo. Esta estructura modular permite integrar distintas técnicas (como selección de características o identificación de negativos fiables) dentro de un flujo robusto de entrenamiento y evaluación, minimizando el riesgo de sobreajuste.

El procedimiento se organiza en dos niveles anidados:

- Un **bucle externo**, que divide el conjunto de datos en particiones de entrenamiento y test, simulando evaluaciones independientes.
- Un **bucle interno**, responsable de seleccionar los mejores hiperparámetros (como el número de características, número de vecinos, umbrales de similitud, etc.) evaluando su rendimiento sobre subconjuntos de validación.

Una vez seleccionada la mejor combinación de hiperparámetros, se reentrena el modelo sobre todo el conjunto de entrenamiento externo y se evalúa en el fold de test correspondiente. Este proceso se repite en todos los folds del bucle externo, y los resultados se agregan para obtener una estimación robusta del rendimiento final.

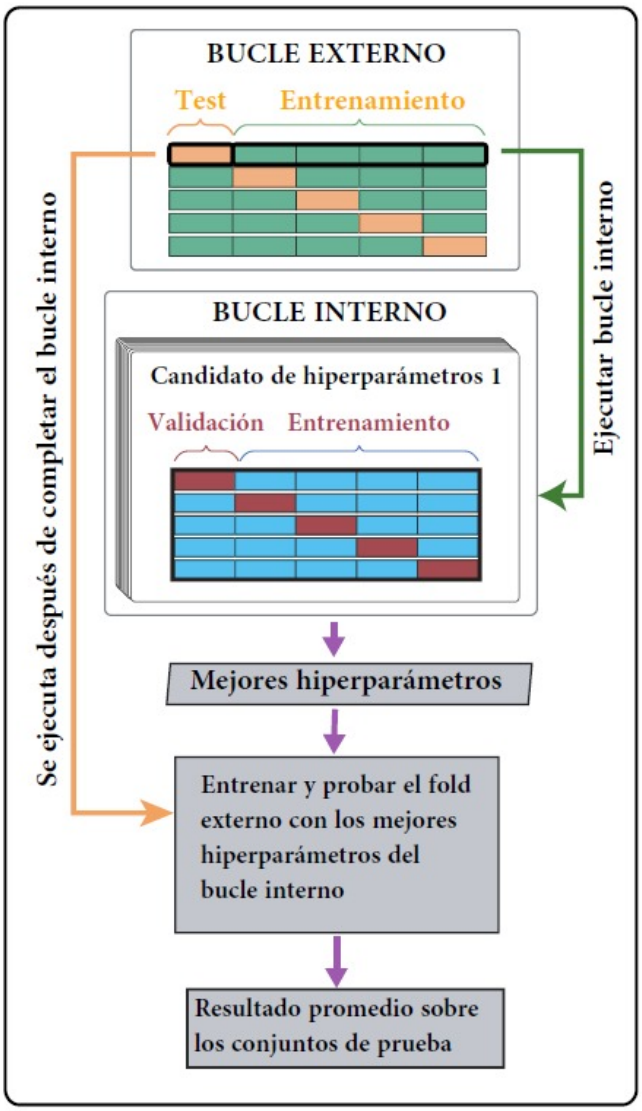


Figura 4.4: Esquema genérico de validación cruzada jerárquica. Cada fold externo actúa como test, mientras que los restantes se utilizan para el ajuste de hiperparámetros en un bucle interno.

Esta organización jerárquica sigue las recomendaciones metodológicas propuestas por Varma y Simon [16] y Cawley y Talbot [17], ampliamente aceptadas en contextos de validación con selección de características o ajuste de hiperparámetros, especialmente en entornos con alta dimensionalidad y escasez de positivos.

A continuación, se describe con mayor detalle cómo se integran *Fast-mRMR* y *PU Learning* dentro del esquema jerárquico, actuando respectivamente sobre el espacio de características y el de instancias.

4.5.2 Integración de Fast-mRMR en el pipeline jerárquico

Una vez implementado el algoritmo *Fast-mRMR* como técnica de selección de características, se integra dentro del esquema de validación cruzada jerárquica descrito en la Sección 4.5.1. Esta incorporación permite evaluar el rendimiento de los modelos de forma robusta, evitando *data leakage*, es decir, asegurando que ninguna información del conjunto de test influya indebidamente en el proceso de selección.

Durante cada iteración, Fast-mRMR se aplica exclusivamente sobre los subconjuntos de entrenamiento de cada fold, tanto interno como externo, garantizando que la elección de características se base únicamente en datos vistos por el modelo durante el aprendizaje. En la validación interna se exploran distintos valores de n_f, y la mejor configuración obtenida se utiliza posteriormente para generar una nueva selección sobre el conjunto completo de entrenamiento externo.

Esta estrategia, repetida en cada partición externa, permite adaptar dinámicamente el subconjunto de características a cada muestra, mejorando la capacidad de generalización del modelo y proporcionando un ranking de genes más robusto frente a variaciones en los datos.

El procedimiento completo puede esquematizarse mediante el siguiente pseudocódigo:

Algorithm 3 Validación cruzada jerárquica con selección de características mediante Fast-mRMR

Require: Dataset D, conjunto de clasificadores $\mathcal{C}$, valores candidatos de número de características n_f

Ensure: Métricas promedio de evaluación externa y ranking final

Dividir D en 10 subconjuntos (folds) externos $D_1, \ldots, D_{10}$
for $i = 1$ **to** 10 **do**
    $D_{\text{test}} \leftarrow D_i$ ▷ Fold externo para test
    $D_{\text{train}} \leftarrow D \setminus D_i$ ▷ Resto para entrenamiento
    for all combinaciones de hiperparámetros $(n_f, \text{params}_{\mathcal{C}})$ **do**
        Dividir D_{train} en 5 folds internos
        for $j = 1$ **to** 5 **do**
            $D_{\text{val}} \leftarrow$ fold j ▷ Validación interna
            $D_{\text{learn}} \leftarrow D_{\text{train}} \setminus D_{\text{val}}$
            Aplicar Fast-mRMR sobre D_{learn} para obtener características seleccionadas
            Transformar D_{learn} y D_{val} usando características seleccionadas
            Entrenar modelo M con $\text{params}_{\mathcal{C}}$ sobre D_{learn}
            Evaluar M sobre D_{val} y guardar métricas
        end for
        Calcular rendimiento promedio de la combinación actual
        Actualizar mejores hiperparámetros si mejora el rendimiento
    end for
    Aplicar Fast-mRMR sobre D_{train} con n_f^*
    Transformar D_{train} y D_{test} con las características seleccionadas
    Entrenar modelo final con los mejores parámetros sobre D_{train}
    Predecir sobre D_{test} y guardar probabilidades
end for
return Métricas promedio finales y ranking global de genes

Este esquema permite adaptar dinámicamente la selección de características sobre cada subconjunto de datos, mejorando la capacidad de generalización del modelo. Además, al aplicar *Fast-mRMR* de forma independiente en cada iteración externa, se obtiene un ranking más robusto y adaptado a diferentes particiones del conjunto de datos.

4.5.3 Aplicación de PU Learning en la validación jerárquica

La técnica *PU Learning* se integra igualmente dentro del esquema de validación cruzada jerárquica descrito en la Sección 4.5.1, permitiendo evaluar su impacto de forma controlada y evitando sesgos derivados de un ajuste inadecuado.

Mientras que *Fast-mRMR* actúa sobre el espacio de características, PU Learning opera sobre el espacio de instancias, con el objetivo de identificar ejemplos negativos fiables dentro del conjunto no etiquetado. Este proceso se basa en un criterio de similitud entre genes, calculado exclusivamente a partir de las características previamente seleccionadas por Fast-mRMR.

De este modo, se garantiza que la detección de negativos fiables se realice en un espacio más reducido e informativo, lo que contribuye a mejorar tanto la eficiencia como la precisión del modelo final.

La lógica del procedimiento sigue una estructura anidada:

- En el **bucle externo** (10 folds), se evalúa el rendimiento final del clasificador en conjuntos completamente independientes, y se recogen las probabilidades necesarias para construir el ranking final de genes candidatos.
- En el **bucle interno** (5 folds), se exploran combinaciones de hiperparámetros como el número de vecinos k y el umbral de similitud t, aplicados sobre las características ya seleccionadas por Fast-mRMR.

Algorithm 4 Validación cruzada jerárquica con PU Learning

Require: Conjunto D (genes positivos y no etiquetados), características seleccionadas F (por Fast-mRMR), valores candidatos k, t

Ensure: Métricas promedio en los folds de test

Dividir D en 10 folds externos $D_1, \ldots, D_{10}$
for $i = 1$ **to** 10 **do**
  $D_{\text{test}} \leftarrow D_i, \quad D_{\text{train}} \leftarrow D \setminus D_i$
  for all combinaciones (k, t) **do**
    Dividir D_{train} en 5 folds internos
    for $j = 1$ **to** 5 **do**
      $D_{\text{val}} \leftarrow$ fold j, $\quad D_{\text{learn}} \leftarrow D_{\text{train}} \setminus D_{\text{val}}$
      Separar P y U en D_{learn}
      $\mathcal{RN} \leftarrow$ ReliableNegatives(P, U, k, t, F) ▷ Usando características F
      Entrenar clasificador con $P \cup \mathcal{RN}$ sobre F
      Evaluar $F1$ sobre D_{val}
    end for
    Actualizar (k^*, t^*) si mejora $F1$
  end for
  Reentrenar clasificador sobre D_{train} con (k^*, t^*)
  Predecir sobre D_{test}, guardar probabilidades
end for
return Métricas promedio (F1, AUC, G-Mean) y ranking final de genes

Este pseudocódigo sigue el esquema propuesto por Jorge Paz [6], adaptado a las necesidades específicas del presente trabajo, en el que **PU Learning opera únicamente sobre las características previamente seleccionadas mediante el algoritmo Fast-mRMR**.

Finalmente, las predicciones obtenidas en los folds externos se combinan para construir un **ranking global de genes candidatos**.

4.5.4 Evaluación del rendimiento y generación del ranking

Una vez completada la validación cruzada externa, se agregan las predicciones probabilísticas generadas en cada fold para cada instancia. Dado que cada gen aparece exactamente una vez como ejemplo de test en cada iteración, se construye un conjunto de probabilidades finales $\hat{p}(x_i)$ mediante media simple:

$$\hat{p}(x_i) = \frac{1}{|J(x_i)|} \sum_{j \in J(x_i)} \hat{p}_j(x_i)$$

donde $J(x_i)$ representa el conjunto de repeticiones en las que x_i fue incluido como ejemplo de prueba, y $\hat{p}_j(x_i)$ la probabilidad estimada en dicha iteración.

A partir de estas puntuaciones se construye el **ranking global de genes candidatos**, ordenando los ejemplos en función de $\hat{p}(x_i)$ de forma descendente. Este ranking se evalúa mediante métricas propias de tareas de recuperación de información, como NDCG@10 [18], Precision@10 y Recall@10 [19], que priorizan la capacidad del sistema para situar genes relevantes en las primeras posiciones.

En paralelo, se calculan métricas de clasificación globales —como F1, AUC-ROC, AUC-PR y G-Mean— con el objetivo de obtener una evaluación complementaria de tipo binaria que permita valorar la capacidad general del sistema para discriminar entre clases [19, 20].

Este enfoque dual, que combina métricas de ordenación y clasificación, proporciona una evaluación integral, coherente con la naturaleza prioritaria del problema abordado.

Capítulo 5

Configuración Experimental

Este capítulo presenta el diseño experimental adoptado para validar el sistema de priorización génica desarrollado. Se describen detalladamente las fuentes de datos utilizadas, el flujo de preprocesamiento aplicado y el funcionamiento de los principales módulos del sistema, incluyendo la reducción de dimensionalidad, la selección de características, la identificación de negativos confiables mediante PU Learning (Aprendizaje positivo y sin etiquetas) y la clasificación supervisada.

Asimismo, se expone el esquema de validación empleado, basado en validación cruzada jerárquica, así como las diferentes configuraciones experimentales evaluadas. El objetivo es analizar de forma sistemática el impacto individual y combinado de cada componente del sistema, así como comparar su rendimiento frente a una línea base inspirada en trabajos anteriores.

5.1 Datos y Preprocesamiento

Para construir una representación robusta y multimodal de cada gen, se integraron múltiples fuentes de información biológica complementaria. Estas fuentes capturan distintos aspectos funcionales, estructurales y expresionales de los genes, y permiten enriquecer el proceso de priorización mediante una codificación informativa y diversificada. A continuación, se describen las representaciones empleadas, siguiendo el orden ilustrado en la Figura 5.1:

(A) PathDIP: se codifica cada gen como un vector binario indicando su pertenencia a rutas biológicas definidas. Integra rutas de diversas fuentes (Reactome, KEGG, WikiPathways, entre otras) y expande las asociaciones mediante predicciones por homología y anotaciones funcionales. Esta representación permite captar información de contexto funcional a nivel de sistemas.

B) GO Terms: cada gen se representa mediante un vector binario esparso, codificando su asociación con términos del dominio "proceso biológico" de la ontología génica. Dado el carácter jerárquico de GO, se aplica una expansión ascendente por la cual, si un gen está anotado con un término, se le asocia también con todos sus ancestros. Esta estrategia, empleada también por Vega Magdaleno et al. [1], busca mejorar la cobertura semántica de la representación funcional.

C) GTEx: se emplean vectores de características numéricas que recogen el nivel de expresión génica mediana en 55 tejidos humanos, medido en unidades TPM. Esta representación cuantitativa, basada en la base GTEx v8, proporciona una perspectiva transversal sobre la actividad transcripcional del gen a lo largo del cuerpo humano.

(D) Coexpresión génica: se construyen perfiles numéricos de alta dimensionalidad que codifican la correlación de expresión entre cada gen y el resto del transcriptoma, según los datos proporcionados por GeneFriends. Esta representación continua permite inferir relaciones funcionales a gran escala a partir de patrones de coexpresión.

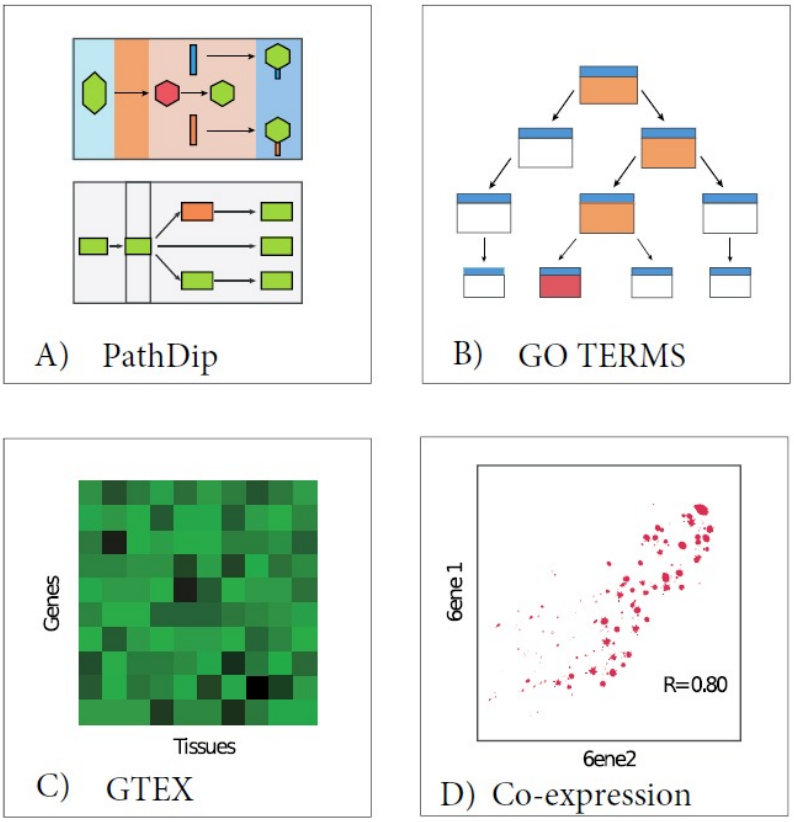


Figura 5.1: Esquema visual de las fuentes de datos utilizadas para caracterizar cada gen tal como fueron adaptadas del trabajo de Vega Magdaleno et al. [1].

5.1.1 Preprocesamiento

Antes de aplicar el algoritmo de **fast_mRMR**, se lleva a cabo un proceso de preprocesamiento para adaptar los distintos conjuntos de datos a un formato común y funcional. Cada archivo se carga desde su versión original en formato CSV, extrayendo las diversas características, las etiquetas (Gen y Class) y los genes.

En esta fase se realizan las siguientes operaciones:

- **Codificación de etiquetas**: Se asigna el valor 1 a los genes conocidos por estar asociados a restricción dietética, y -1 al resto. Esta codificación binaria se utiliza como variable objetivo en las fases posteriores.
- **Discretización**: En datasets con variables continuas (como GTEx o Coexpression), los valores se transforman en variables discretas mediante partición en cuantiles. Se utilizan 10 bins para asegurar una distribución equilibrada. Esto es necesario para aplicar el método de Fast-mRMR, que trabaja sobre variables discretas.
- **Variables ya binarias**: En otros conjuntos, como GO o PathDIP, las características ya son binarias (0/1), por lo que se emplean directamente sin modificación.
- **Estandarización del formato**: Una vez preprocesados, todos los datos se representan como matrices $n \times d$, donde n es el número de genes y d el número de características, manteniendo una estructura tabular homogénea entre conjuntos.

Este preprocesamiento permite aplicar de forma consistente los módulos de selección de características y clasificación, y la preparación de los datos para su conversión al formato específico que requiere Fast-mRMR como entrada.

5.2 Configuración del algoritmo *Fast-mRMR*

En coherencia con lo expuesto en la Sección 4.2, el algoritmo *Fast-mRMR* se utilizó como mecanismo de reducción de dimensionalidad previo al entrenamiento, con el objetivo de mejorar la eficiencia computacional y mitigar la redundancia entre características. Su aplicación fue cuidadosamente ajustada a las particularidades de cada conjunto de datos, con especial atención al coste computacional cuadrático asociado al número de características ($\mathcal{O}(d^2)$).

5.2.1 Ajustes adaptativos en función de la dimensionalidad

Dado el elevado rango de dimensionalidades presentes entre las distintas fuentes de datos, se implementó una estrategia adaptativa para la selección de características mediante *Fast-*

mRMR, ajustando el porcentaje de retención de características en función tanto de la escala del conjunto como del rendimiento observado durante la validación cruzada.

- **Fuentes de baja dimensionalidad (e.g., GTEx, PathDIP)**: Al contar con un número reducido de características, el algoritmo se ejecuta con rapidez y sin requisitos adicionales de recursos. En estos casos se seleccionaron proporciones elevadas (20%, 40%, hasta 60%) del total de características, preservando así una mayor riqueza informativa sin riesgo de sobreajuste significativo.

- **Fuentes de alta dimensionalidad (e.g., GO, Coexpression)**: En dominios con miles o decenas de miles de características, se aplicaron reducciones más agresivas (5%, 10%, 20%, 40%) e incluso inferiores al 5% en ciertos casos. Esta estrategia permitió aliviar el coste computacional y mejorar la capacidad de generalización del modelo, evitando el impacto negativo de la maldición de la dimensionalidad.

- **Selección basada en validación**: La elección final del porcentaje de características retenidas no fue fija ni arbitraria, sino guiada por un proceso empírico de validación cruzada. En cada fold, se monitorizaron métricas agregadas de rendimiento como **F1-score**, **AUC** y **G-mean**, priorizando aquellas que ofrecen una visión equilibrada del comportamiento global del clasificador. A diferencia de métricas puntuales como sensibilidad, especificidad o precisión, estas medidas reflejan mejor el compromiso general entre clases desbalanceadas. El umbral de selección se fijó en función del rendimiento promedio observado tanto por fold como a nivel global, lo que permitió ajustar de forma dinámica el grado de reducción para cada fuente de características según su complejidad estructural y su valor predictivo efectivo.

- **Casos extremos (e.g., Coexpression)**: En conjuntos extremadamente grandes como el de coexpresión (con más de 44,000 atributos), se aplicó un esquema combinado de particionado y reducción. Se dividió el espacio total en cinco grupos disjuntos del mismo tamaño y se aplicó *Fast-mRMR* por separado en cada uno, seleccionando un 20% de cada bloque. Esta técnica, inspirada en *feature bagging*[14, 15] , permitió reducir el espacio total a 8,986 características seleccionadas, facilitando su tratamiento posterior.

Este esquema flexible y validado experimentalmente asegura un balance adecuado entre capacidad predictiva, interpretabilidad y eficiencia computacional, adaptando la presión de selección a la estructura y complejidad de cada fuente de información.

5.3 Configuración experimental de PU Learning

La estrategia de PU Learning descrita en la Sección 4.4.2 se integró dentro del ciclo de validación cruzada jerárquica, con el objetivo de identificar genes no etiquetados con alta probabilidad de pertenecer a la clase negativa.

Los hiperparámetros principales del método, k (número de vecinos) y t (umbral de densidad local), se optimizaron mediante una búsqueda en cuadrícula dentro del bucle de validación interna. Para cada combinación, se construyó un subconjunto de negativos confiables $\mathcal{RN}$ y se entrenó un clasificador binario utilizando como conjunto de entrenamiento la unión $P \cup \mathcal{RN}$. Las combinaciones evaluadas fueron:

$$k \in \{5,\ 8\}, \quad t \in \{0.8,\ 0.875,\ 1.0\}$$

y la selección final se realizó en base al rendimiento promedio obtenido en validación interna, considerando métricas como F1-score, AUC-ROC y G-mean.

Todos los experimentos se llevaron a cabo sobre las características de entrada resultantes de la reducción dimensional previa realizada mediante *Fast-mRMR*, asegurando así un espacio de representación más informativo y eficiente. Sobre estas características se aplicaron clasificadores de tipo *ensemble*, concretamente *CatBoost* y *Balanced Random Forest*, seleccionados por su robustez frente a desbalance de clases y su buen desempeño en tareas bioinformáticas similares, los cuales se describen más en detalle a continuación.

5.4 Entrenamiento de Clasificadores

Una vez construido el conjunto de entrenamiento, se procedió a entrenar modelos supervisados capaces de estimar una probabilidad $\hat{p}(y = 1 \mid x)$ que refleje el grado de asociación de cada gen candidato con el proceso de interés.

5.4.1 Modelos utilizados

Se evaluaron diversos clasificadores, priorizando aquellos robustos ante desequilibrios de clase y que han demostrado alto rendimiento en tareas con datos tabulares y clases minoritarias. Finalmente, se seleccionaron dos modelos complementarios, siguiendo los precedentes establecidos en trabajos recientes sobre identificación de genes candidatos [1, 6]:

- **Balanced Random Forest (BRF)** [21]: variante del *Random Forest* que aplica muestreo equilibrado en cada árbol para atenuar el sesgo hacia la clase mayoritaria. Esta estrategia mejora la sensibilidad hacia la clase minoritaria, y permite además estimar la relevancia de cada característica en el modelo mediante la disminución promedio de impureza (*mean decrease in impurity*), una métrica basada en la capacidad de cada característica para dividir correctamente los datos en los nodos de decisión.
- **CatBoost** [22]: algoritmo de *boosting* basado en árboles de decisión, optimizado para escenarios con características categóricas o dispersas. Emplea ajuste interno de pesos de clase (*auto_class_weights*) y ha demostrado gran capacidad predictiva en tareas bioinformáticas similares. Su eficiencia y estabilidad lo convierten en una buena elección para este trabajo.

Ambos clasificadores se entrenaron con **500 estimadores** y **semillas aleatorias fijas** para garantizar la reproducibilidad de resultados. En concreto, se emplearon las siguientes semillas:

{14, 33, 39, 42, 727, 1312, 1337, 56709, 177013, 241543903}

En *BRF* se desactivó el muestreo con reemplazo para evitar redundancia en los subconjuntos balanceados. En *CatBoost* se prescindió de validación temprana o técnicas de codificación categórica, dado que todas las características fueron previamente discretizadas o binarizadas.

Los resultados reportados corresponden al promedio de las métricas obtenidas a lo largo de estas ejecuciones independientes.

5.4.2 Tratamiento del desbalance

El desbalance de clases inherente a los conjuntos $P \cup \mathcal{RN}$ fue abordado mediante estrategias internas de cada modelo. En BRF, la proporción equilibrada de clases en cada árbol garantiza un sesgo controlado hacia la clase positiva. En CatBoost, el peso relativo de cada clase se ajustó automáticamente durante la optimización de la función de pérdida.

De forma complementaria, se evaluó la incorporación de pesos manuales para reforzar la compensación entre clases, definidos como:

$$w_{\text{pos}} = \frac{n}{2 \cdot n_1}, \quad w_{\text{neg}} = \frac{n}{2 \cdot n_0}$$

donde n es el tamaño total del conjunto de entrenamiento, n_1 el número de ejemplos positivos y n_0 el de negativos confiables. Esta formulación garantiza que ambas clases contri-

buyan equitativamente a la función objetivo, especialmente en contextos con gran desbalance.

Ambos modelos se integraron en el pipeline de validación cruzada jerárquica y sus rendimientos comparativos se detallarán en la Sección 6.

5.5 Configuración experimental de la validación cruzada

El sistema se evaluó mediante un esquema de validación cruzada jerárquica implementado en *Python*, siguiendo el pipeline descrito en la Sección 4. Se utilizó un esquema 10×10 estratificado en el bucle externo, empleando las 10 semillas definidas en la Sección 5.4. En cada iteración, se entrenó un modelo con los 9 folds y se evaluó en el restante, generando predicciones probabilísticas para cada gen.

Dentro de cada iteración externa se integraron:

- **Fast-mRMR**: aplicado en cada fold, con o sin bagging según el tipo de fuente.
- **PU Learning** (si aplica): selección de RN con parámetros k y t explorados en validación interna.
- **Clasificadores**: BRF y CatBoost, con hiperparámetros definidos en la Tabla 5.5.1.

La selección de hiperparámetros se realizó mediante validación cruzada interna (10×5), aplicando una búsqueda en rejilla (*grid search*) sobre las combinaciones posibles de valores para cada parámetro. Esta técnica permite explorar sistemáticamente el espacio de configuraciones para identificar aquella que maximiza el rendimiento del modelo.

Como métrica objetivo se utilizó el **F1-score**, ya que ofrece un equilibrio entre precisión y exhaustividad, siendo especialmente adecuada en contextos de aprendizaje positivo-no etiquetado (PU). En este tipo de problemas, donde la clase negativa es incierta o subrepresentada, muchas métricas tradicionales (como la exactitud o incluso la AUC-ROC) pueden resultar engañosas o estar subestimadas. El F1-score, al enfocarse en los positivos detectados correctamente, proporciona una medida más realista del rendimiento bajo esta configuración asimétrica.

Las predicciones generadas en cada fold externo se almacenaron y promediaron por instancia:

$$\hat{p}(x_i) = \frac{1}{|J(x_i)|} \sum_{j \in J(x_i)} \hat{p}_j(x_i)$$

donde $J(x_i)$ es el conjunto de iteraciones donde x_i fue usado como test. Este promedio permitió construir un **ranking global** de candidatos más estable, acumulando un total de 100 modelos externos por configuración experimental.

5.5.1 Resumen de hiperparámetros del pipeline

En la Tabla 5.1, se presentan los principales hiperparámetros evaluados en cada componente del sistema durante la validación cruzada interna. Estos valores fueron ajustados de forma empírica y validados según la estructura jerárquica descrita.

Cuadro 5.1: Resumen de hiperparámetros ajustados por módulo del sistema

Componente	**Hiperparámetros evaluados**
Fast-mRMR (Selección de características)	• Porcentaje de retención: 2.5% – 60%, según la fuente y validación. • En **Coexpression**: bagging en subespacios (5 particiones), 20% retenido por bloque.
PU Learning (Selección de $\mathcal{RN}$)	• Vecinos k: {5, 8} • Umbral de densidad t: {0.8, 0.875, 1.0} • Similitud: índice de Jaccard
Clasificadores supervisados	**Balanced Random Forest (BRF)**: • Número de árboles: 500 • Muestreo equilibrado sin reemplazo • Pesos de clase: automáticos o manuales **CatBoost**: • Número de árboles: 500 • Pesos de clase: automáticos
Validación cruzada	• Externa: 10×10 estratificada (con semillas fijas) • Interna: 10×5 estratificada • Métricas objetivo: F1-score al trabajar sobre datos PU

5.6 Elección de la Evaluación

Las métricas de rendimiento se calcularon a partir de las predicciones finales generadas tras el proceso descrito en la Sección 5.5. En configuraciones con múltiples submodelos (p. ej., bagging o distintas particiones en selección de características), las probabilidades fueron agregadas por promedio antes de aplicar las métricas.

Se reportaron los resultados como el **promedio y desviación estándar** de cada métrica, obtenidos tras las 10 ejecuciones completas del procedimiento (una por semilla distinta), lo que permite estimar tanto el rendimiento medio como la estabilidad del sistema.

5.6.1 Métricas aplicadas

Se realiza una evaluación tanto de la capacidad de clasificación binaria como de la calidad del ranking generado por el sistema. A continuación, se describen las métricas empleadas:

- **F1-score**: mide el compromiso entre *precisión* y *sensibilidad* [23]. Es especialmente útil cuando se busca evaluar de forma equilibrada la capacidad del modelo para detectar correctamente los positivos (sensibilidad) y evitar falsas alarmas (precisión), sin favorecer ninguna de las dos. Se define como:

$$F1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}}, \quad \text{donde}$$

$$\text{Precisión} = \frac{VP}{VP + FP}, \quad \text{Sensibilidad} = \frac{VP}{VP + FN}$$

  (VP: verdaderos positivos, VN: verdaderos negativos, FP: falsos positivos, FN: falsos negativos)

  El F1-score se emplea como métrica principal durante la validación cruzada interna, ya que permite seleccionar subconjuntos de características que logran un buen equilibrio entre precisión y sensibilidad. Dado que Fast-mRMR no incorpora directamente información sobre la clase negativa —especialmente relevante en el contexto de datos PU—, esta métrica es adecuada para evaluar el rendimiento sin sesgos hacia ninguna clase.

- **G-mean**: refleja el rendimiento equilibrado del modelo en ambas clases, combinando sensibilidad y especificidad [20]:

$$G\text{-}mean = \sqrt{\text{Sensibilidad} \cdot \text{Especificidad}} = \sqrt{\frac{VP}{VP + FN} \cdot \frac{VN}{VN + FP}}$$

G-mean es útil para comprobar si la selección de características afecta de forma desproporcionada a la detección de una de las clases, permitiendo una evaluación equilibrada del rendimiento global.

- **AUC-ROC (Área bajo la curva ROC)**: esta métrica evalúa la capacidad del modelo para distinguir correctamente entre clases positivas y negativas, considerando todos los posibles umbrales de decisión. La curva ROC (*Receiver Operating Characteristic*) representa gráficamente la relación entre la **tasa de verdaderos positivos** (TVP o sensibilidad) y la **tasa de falsos positivos** (TFP), definida como:

$$\text{TFP} = \frac{FP}{FP + VN}$$

  El valor de AUC (Área Bajo la Curva) resume esta curva en un único número entre 0 y 1:

  - Un valor de 1 indica una separación perfecta entre clases.
  - Un valor de 0,5 indica un comportamiento equivalente al azar.

  Formalmente, el área bajo la curva ROC se define como:

$$\text{AUC-ROC} = \int_0^1 \text{Sensibilidad}(\text{TFP})\, d(\text{TFP})$$

  Esta fórmula expresa que el área bajo la curva se calcula como la integral de la función *Sensibilidad* (eje Y) en función de la *TFP* (eje X). El término $d(\text{TFP})$ representa un pequeño incremento en la tasa de falsos positivos, es decir, describe cómo evoluciona la sensibilidad a medida que cambia la TFP al modificar el umbral de decisión.

  Tom Fawcett [24] sistematizó su uso como métrica estándar en clasificación binaria. Se emplea para evaluar el rendimiento de los modelos tras la reducción de características con *Fast-mRMR*, ya que permite medir su capacidad de discriminación sin depender de un único umbral.

- **AUC-PR (Área bajo la curva Precisión-Recall)**: esta métrica mide el rendimiento del modelo enfocándose en su capacidad para identificar correctamente la clase positiva. La curva *Precisión-Recall* representa gráficamente la relación entre la **precisión** y la **sensibilidad** (*recall*) a lo largo de distintos umbrales de decisión.

  Jesse Davis y Mark Goadrich [19] demostraron que AUC-PR puede ofrecer una evaluación más informativa que AUC-ROC en escenarios donde los ejemplos positivos son minoritarios, al poner más énfasis en el rendimiento sobre esa clase.

Formalmente, se define como:

$$\text{AUC-PR} = \int_0^1 \text{Precisión}(r)\, dr$$

donde r representa la sensibilidad (recall), y $d(r)$ corresponde a un pequeño incremento en dicha sensibilidad a medida que se recorre la curva. Esta integral cuantifica el área bajo la curva trazada por la precisión en función del recall, dando mayor peso a las predicciones sobre los positivos.

Aunque en nuestro caso la proporción de positivos no es extremadamente baja, AUC-PR complementa a AUC-ROC al capturar de forma más directa la calidad de las predicciones positivas, aspecto especialmente relevante en tareas de priorización génica.

- **Precision@10**: mide la proporción de verdaderos positivos presentes entre las 10 primeras posiciones del ranking generado por el modelo. Es una métrica centrada en los primeros resultados, lo que la hace especialmente relevante en tareas de priorización, donde solo se validan o estudian unos pocos candidatos.

  Se calcula como:

  $$\text{Precision@10} = \frac{\text{Número de verdaderos positivos en el top-10}}{10}$$

  Esta métrica refleja cuántas de las predicciones más confiables del modelo (las 10 primeras) son realmente positivas. En nuestro caso, permite evaluar cuán útil sería el sistema en un escenario real donde solo se pudieran analizar experimentalmente los genes mejor clasificados. Su interpretación directa y enfoque en los mejores resultados la convierten en una medida práctica y comprensible para tareas de descubrimiento biomédico. Este tipo de métricas han sido empleadas, por ejemplo, en trabajos como el de Yang et al. [7], enfocados en la priorización génica mediante redes neuronales.

- **Recall@10**: estima la proporción de instancias positivas que se encuentran dentro de las 10 primeras posiciones del ranking. A diferencia de la *Precision@10*, que evalúa la calidad de los elementos recomendados, esta métrica se centra en la capacidad del sistema para recuperar los positivos reales de forma temprana.

  Se define como:

  $$\text{Recall@10} = \frac{\text{Número de verdaderos positivos en el top-10}}{\text{Total de verdaderos positivos}}$$

  Recall@10 permite cuantificar qué fracción de los genes positivos conocidos aparecen entre los candidatos mejor puntuados por el modelo. Es especialmente útil para valorar

el grado de cobertura del sistema en tareas donde no solo importa la precisión, sino también la capacidad de recuperar tantos elementos relevantes como sea posible desde el inicio. Este enfoque ha sido utilizado también en estudios como el de Gligorijević et al. [25], donde se evalúa el ranking de genes relevantes mediante modelos de aprendizaje profundo.

- **NDCG@10 (Normalized Discounted Cumulative Gain)**: evalúa la calidad del ranking generado, teniendo en cuenta no solo cuántos elementos relevantes (positivos) aparecen en las primeras posiciones, sino también en qué lugar exacto se encuentran. Esta métrica, propuesta por Kalervo Järvelin y Jaana Kekäläinen [18], asigna más valor a los aciertos en posiciones altas del ranking.

  Se define como:

$$NDCG@10 = \frac{DCG@10}{IDCG@10}, \quad \text{donde } DCG@10 = \sum_{i=1}^{10} \frac{rel_i}{\log_2(i+1)}$$

  Aquí, rel_i representa la relevancia del elemento en la posición i (en nuestro caso, 1 si es un gen positivo, 0 en caso contrario), y $IDCG@10$ corresponde al valor ideal del DCG si todos los positivos estuvieran ordenados en las primeras posiciones posibles.

  NDCG@10 se utiliza en este trabajo como métrica complementaria a *Precision@10* y *Recall@10*, ya que ofrece una evaluación más matizada del ranking, penalizando menos los aciertos que aparecen más abajo en el top-10, pero premiando fuertemente aquellos que se sitúan en los primeros lugares.

5.7 Pruebas estadísticas

Para validar la significancia de las diferencias observadas entre las configuraciones evaluadas, se aplican pruebas estadísticas siguiendo la metodología descrita en trabajos previos como el de Paz-Ruza et al. [6]. En concreto, se emplea un **t-test pareado bilateral** con un nivel de significancia $\alpha = 0.05$.

Este tipo de prueba permite determinar si las diferencias en el rendimiento entre dos enfoques son estadísticamente significativas o pueden atribuirse al azar. El diseño *pareado* se justifica porque las comparaciones se realizan sobre los mismos subconjuntos de datos en cada ejecución, lo que reduce la variabilidad no controlada. El enfoque *bilateral*, por su parte, considera tanto posibles mejoras como empeoramientos entre métodos, sin asumir una dirección preferente.

Tal como se propone en el trabajo de referencia, las comparaciones se basan en las métricas medias obtenidas tras **10 ejecuciones independientes** (*runs*) por configuración, cada una con su propia validación cruzada. Este enfoque permite capturar la variabilidad del proceso y realizar contrastes estadísticos sólidos sobre las distribuciones resultantes.

Cuando se detecta una diferencia significativa ($p < 0.05$), se indica explícitamente en las tablas del Capítulo 6 con el símbolo † junto al valor correspondiente. Esto permite identificar no solo qué configuraciones ofrecen mejor rendimiento aparente, sino también cuáles presentan mejoras consistentes desde un punto de vista estadístico.

5.8 Descripción del entorno de pruebas

Los experimentos se realizaron en un servidor proporcionado por el **CITIC** (Centro de Investigación en Tecnologías de la Información y las Comunicaciones). El acceso al entorno de pruebas se realizaba de forma remota mediante conexión VPN y el uso de la herramienta **NoMachine**, que permitía interactuar con un escritorio virtual del servidor de manera fluida.

Las características técnicas del entorno eran las siguientes:

- **Procesador:** Intel Core i7-11700 (11ª generación), 8 núcleos físicos y 16 hilos, con frecuencia base de 2.50 GHz y frecuencia máxima de hasta 4.90 GHz.
- **Memoria RAM:** 16 GB DDR4, con aproximadamente 10 GB disponibles durante la ejecución de los experimentos.
- **Almacenamiento:** Unidad SSD NVMe de 931 GB, utilizada como sistema principal de archivos.
- **Sistema operativo:** Ubuntu 20.04.6 LTS (arquitectura *x86_64*).
- **Número de hilos lógicos:** 16 necesarios para la ejecución de Fast_mRMR

Este entorno permitió ejecutar el pipeline completo con tiempos razonables incluso en tareas exigentes, como la reducción de dimensionalidad sobre el dataset *Coexpression*, que contiene más de 44,000 características.

5.9 Librerías y dependencias utilizadas

El desarrollo del sistema se llevó a cabo utilizando Python 3.10 dentro de un entorno virtual gestionado mediante *venv*. Esta configuración garantiza la reproducibilidad del expe-

rimento y evita conflictos con otras configuraciones del sistema.

A continuación se detallan las principales librerías empleadas, su funcionalidad dentro del proyecto y las versiones específicas utilizadas:

- **Manipulación y procesamiento de datos:**
  - *numpy* (v1.23.0): operaciones numéricas vectorizadas y manejo eficiente de arrays.
  - *pandas* (v1.5.3): estructuras de datos tipo DataFrame para manipulación tabular y lectura/escritura de archivos.
- **Preprocesamiento y discretización:**
  - *scikit-learn.preprocessing* (v1.1.3): normalización de características y discretización mediante *KBinsDiscretizer*.
- **Modelado y algoritmos de clasificación:**
  - *imblearn.ensemble.BalancedRandomForestClassifier* (v0.12.0): clasificador basado en random forest diseñado para manejar conjuntos de datos desbalanceados.
  - *catboost.CatBoostClassifier* (v1.2.2): algoritmo de boosting optimizado para rendimiento y adecuado para variables categóricas o datos tabulares.
- **Evaluación y métricas de rendimiento:**
  - *scikit-learn.metrics* (v1.1.3): cálculo de métricas como *f1_score*, *Precisión*, *Auc roc*, *Ndcg*, entre otras.
  - *imblearn.metrics* (v0.12.0): métricas específicas como *Gmean*, *Sensibilidad*, *Especifidad*.
- **Análisis estadístico:**
  - *scipy.stats* (v1.9.3): implementación del *t-test* pareado bilateral para evaluar la significancia estadística entre configuraciones.
- **Gestión de experimentos:**
  - *neptune* (v1.10.0): plataforma de gestión y seguimiento de experimentos, utilizada para registrar métricas, parámetros y resultados de múltiples ejecuciones.

El código fuente completo que integra estas librerías y permite reproducir los experimentos descritos en este trabajo se encuentra disponible públicamente en el repositorio de GitHub https://github.com/ru-farelo/TFG-DR-Feature-Selection..

Capítulo 6

Resultados experimentales

Este capítulo se organiza en cuatro bloques principales, siguiendo la metodología propuesta. En primer lugar, se evalúa el rendimiento del algoritmo *Fast-mRMR* aplicado individualmente sobre diferentes conjuntos funcionales (GO, PathDIP y GTEx), analizando tanto aquellos con buen desempeño como los que presentaron limitaciones en trabajos anteriores.

A continuación, se estudia su aplicación sobre conjuntos combinados de datos con intersección de genes y unión de características, donde se comprueba que la selección de características mejora notablemente los resultados frente a estrategias previas planteadas en el trabajo de referencia de Vega Magdaleno [1]. El tercer bloque incluye propuestas complementarias, como la aplicación de *bagging* en escenarios con alta dimensionalidad. Finalmente, se analiza la integración de *Fast-mRMR* con técnicas de aprendizaje *Positive-Unlabeled* (PU), valorando su efecto en conjunto y por separado.

Todos los experimentos se han llevado a cabo utilizando los clasificadores *Balanced Random Forest* (BRF) y *CatBoost*, y las métricas se reportan como promedio, junto con su desviación estándar, tras múltiples repeticiones de validación cruzada. Es importante señalar que el valor del 100% de características equivale a no aplicar ninguna selección, representando así el rendimiento base de cada clasificador.

6.1 Evaluación de Fast-mRMR

Esta sección presenta los resultados de nuestro enfoque basado en la selección de características mediante *Fast-mRMR*, aplicado a los conjuntos funcionales **PathDIP** y **Gene Ontology (GO)**, ampliamente utilizados en estudios previos. Evaluamos su rendimiento en la tarea de priorización de genes relacionados con la restricción dietética (RD), comparándolo con dos referencias: el método base sin selección de características y el enfoque PU Learning [6].

Los experimentos se realizaron empleando los clasificadores *Balanced Random Forest (BRF)* y *CatBoost*. Se reportan cuatro métricas estándar: *AUC-ROC*, *AUC-PR*, *G-Mean* y *F1-Score*, promediadas a lo largo de 10 ejecuciones independientes con validación cruzada anidada.

Cuadro 6.1: Comparación del rendimiento predictivo entre el método original sin selección, el enfoque PU Learning *(Jorge Paz et al.)* y el método propuesto basado en *Fast-mRMR*. Se muestran resultados sobre los conjuntos PathDIP y GO con dos clasificadores. Los valores se expresan como *media ± desviación típica*, y entre paréntesis se indica el porcentaje de características seleccionadas en el método propuesto. En negrita, los mejores resultados por métrica. El símbolo † indica significancia estadística ($p < 0.05$) frente al resto de métodos para esa métrica.

Método	**Conjunto / Clasificador**	***AUC-ROC***	***AUC-PR***	***G-Mean***	***F1-Score***
Original (sin selección)	PathDIP / CAT	0.813 ± 0.010	0.557 ± 0.011	0.702 ± 0.012	0.505 ± 0.015
	PathDIP / BRF	0.827 ± 0.007	0.563 ± 0.013	0.771 ± 0.011	0.476 ± 0.013
	GO / CAT	0.838 ± 0.012	0.499 ± 0.020	0.684 ± 0.015	0.498 ± 0.020
	GO / BRF	0.835 ± 0.011	0.429 ± 0.008	0.763 ± 0.010	0.389 ± 0.010
PU Learning (Jorge Paz et al.)	PathDIP / CAT	0.829 ± 0.011	0.526 ± 0.020	0.749 ± 0.007	0.531 ± 0.011
	PathDIP / BRF	0.827 ± 0.006	0.549 ± 0.016	0.768 ± 0.010	0.461 ± 0.011
	GO / CAT	0.839 ± 0.012	0.438 ± 0.015	0.729 ± 0.011	0.496 ± 0.007
	GO / BRF	0.829 ± 0.011	0.390 ± 0.008	0.762 ± 0.011	0.380 ± 0.009
Fast-mRMR (propuesto)	PathDIP / CAT	0.830 ± 0.009 (80%)	**0.588 ± 0.015** † (17.5%)	0.732 ± 0.010 (17.5%)	0.523 ± 0.020 (45%)
	PathDIP / BRF	0.841 ± 0.006 (45%)	0.562 ± 0.016 (80%)	**0.779 ± 0.009** † (50%)	0.479 ± 0.016 (2.5%)
	GO / CAT	**0.852 ± 0.009** † (5%)	0.533 ± 0.022 (5%)	0.738 ± 0.016 (5%)	**0.532 ± 0.014** (5%)
	GO / BRF	0.842 ± 0.009 (40%)	0.479 ± 0.015 (5%)	0.766 ± 0.007 (5%)	0.450 ± 0.010 (5%)

Como puede observarse en la Tabla 6.1, el mejor resultado de cada métrica se encuentra en el bloque de **Fast-mRMR (propuesto)**, indicado en negrita. Para validar estos resultados, se aplicó un análisis de significancia estadística mediante un *t-test* pareado bilateral, enfrentando —para cada métrica— el valor más alto contra todas las demás configuraciones de esa columna (un total de 11 comparaciones por métrica). El símbolo † indica aquellos valores cuya diferencia es **estadísticamente significativa** ($p < 0.05$), lo cual refuerza que las mejoras observadas no son atribuibles al azar.

Este análisis revela que, aunque el mejor **F1-score** (0.532) alcanzado en GO con CatBoost no resultó estadísticamente significativo, el método propuesto sí logra mejoras significativas

en el resto de métricas clave: **AUC-ROC**, **AUC-PR** y **G-Mean**. Estos resultados reflejan de forma robusta la eficacia de *Fast-mRMR* como técnica de selección de características.

Además, el enfoque permite reducir drásticamente el número de variables utilizadas —en muchos casos, por debajo del 20%— concentrándose en las más informativas. Esto no solo mejora la capacidad predictiva, sino que también aporta ventajas prácticas en cuanto a eficiencia computacional y simplicidad del modelo.

En particular, destacan las siguientes configuraciones:

- **GO + CatBoost**: AUC-ROC de **0.852**†.
- **PathDIP + CatBoost**: AUC-PR de **0.588**†.
- **PathDIP + BRF**: G-Mean de **0.779**†.

El **F1-score** se presenta como la métrica central en este análisis por ser especialmente adecuada en entornos con datos *Positive-Unlabeled (PU)*, como es el caso de este trabajo. En este contexto, no se dispone de ejemplos negativos verificados, lo que impacta directamente en la evaluación del rendimiento. Concretamente, la métrica de *Precision* tiende a estar sistemáticamente subestimada, ya que muchos ejemplos clasificados como falsos positivos podrían ser en realidad positivos verdaderos que carecen de etiqueta. Aunque el *Recall* no se ve afectado por este sesgo, el **F1-score**, al ser su media armónica, hereda la subestimación de la *Precision*. Este hecho, demostrado por Elkan y Noto (2008) [26],lo que hace que el F1-score sea una medida de rendimiento conservadora pero robusta en escenarios PU.

Por tanto, valores absolutos aparentemente bajos (como 0.47 o 0.53 en F1) no deben interpretarse como un mal rendimiento, sino como el reflejo de esta incertidumbre estructural en los datos. Lo importante, en este contexto, no es tanto el valor en sí, sino su **comparación relativa entre métodos bajo las mismas condiciones**. Si un enfoque obtiene consistentemente un mayor F1-score que otro, se puede afirmar con confianza que su rendimiento es superior, independientemente de la subestimación de base.

6.1.1 Influencia del número de características en el rendimiento

En esta subsección se analiza cómo varía el rendimiento del sistema en función del porcentaje de características seleccionadas mediante Fast-mRMR, evaluando distintos porcentajes de selección. Este análisis se centra en el conjunto GO, donde el impacto fue más evidente.

Las Figuras 6.1 y 6.2 muestran la evolución de las métricas clave (AUC-ROC, AUC-PR, F1-score y G-Mean) al aplicar CatBoost y BRF, respectivamente. En ambos casos se observa que es posible alcanzar —e incluso superar— el rendimiento original sin selección utilizando solo entre el 5% y el 20% de las características, lo que confirma el beneficio de reducir la dimensionalidad. Además, a partir de cierto punto, incluir más características no aporta mejoras sustanciales e incluso puede degradar el rendimiento, lo que sugiere que añadir información irrelevante introduce ruido en el modelo.

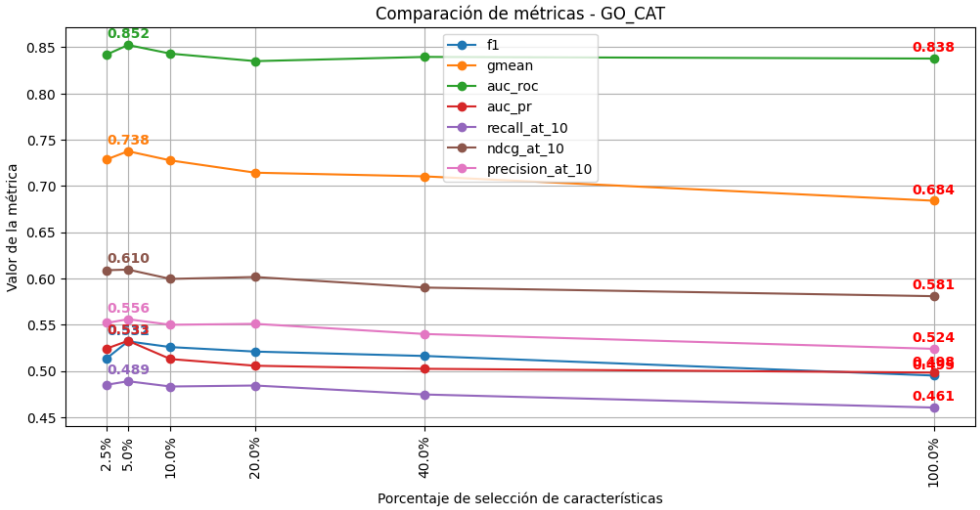


Figura 6.1: Evolución de las métricas sobre el conjunto GO con el clasificador CatBoost para distintos porcentajes de características seleccionadas mediante Fast-mRMR.

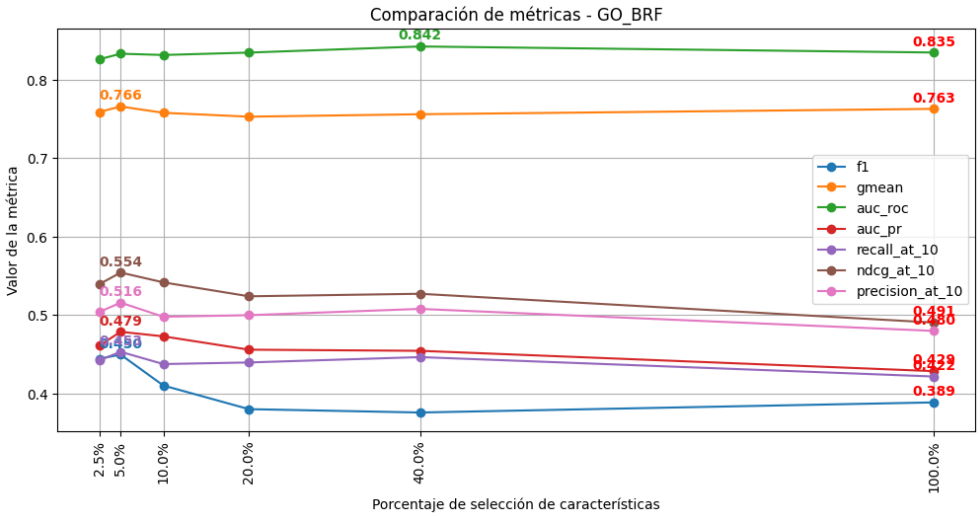


Figura 6.2: Evolución de las métricas sobre el conjunto GO con el clasificador BRF para distintos porcentajes de características seleccionadas mediante Fast-mRMR.

6.1.2 Estudio de caso: conjunto funcional GTEx

El conjunto **GTEx** se incluye como caso de estudio especial por dos razones fundamentales: (i) presenta diferencias sustanciales respecto a GO y PathDIP en cuanto a composición génica, incluyendo más de un centenar de genes exclusivos, y (ii) mostró un rendimiento notablemente bajo en el trabajo original de referencia [1]. Este conjunto, basado en perfiles de expresión génica continua, representa un entorno especialmente desafiante para la priorización de genes, tanto por la escasa estructuración como por el reducido número de características disponibles (aproximadamente 25).

Cuadro 6.2: Comparación del rendimiento predictivo en el conjunto **GTEx** entre el método original (sin selección) y nuestro enfoque basado en *Fast-mRMR*, usando dos clasificadores. Los resultados se presentan como *media ± desviación típica*, y entre paréntesis se indica el porcentaje de características seleccionadas en el método propuesto. En negrita se resaltan los mejores resultados por métrica. El símbolo † indica significancia estadística ($p < 0.05$).

Método	**Clasificador**	***AUC-ROC***	***AUC-PR***	***G-Mean***	***F1-Score***
Original (sin selección)	BRF	0.535 ± 0.007	0.135 ± 0.005	0.512 ± 0.015	0.184 ± 0.009
	CatBoost	0.523 ± 0.011	0.125 ± 0.011	0.145 ± 0.041	0.067 ± 0.022
Fast-mRMR (propuesto)	BRF	**0.568 ± 0.020 †** (15%)	**0.164 ± 0.007 †** (5%)	**0.540 ± 0.021 †** (15%)	**0.199 ± 0.013 †** (15%)
	CatBoost	0.561 ± 0.022 (15%)	0.148 ± 0.010 (10%)	0.496 ± 0.012 (5%)	0.173 ± 0.011 (5%)

Como se muestra en la Tabla 6.2, el método **Fast-mRMR** supera consistentemente al modelo base sin selección en todas las métricas y clasificadores evaluados. Estas mejoras han sido validadas estadísticamente mediante un *t-test* pareado bilateral: en cada métrica se comparó el mejor resultado frente a los otros tres valores de la misma columna (tres comprobaciones por métrica). En todos los casos, las diferencias fueron estadísticamente significativas ($p < 0.05$), lo que confirma la capacidad del enfoque propuesto para extraer información útil incluso en conjuntos con baja estructura y escasa dimensionalidad.

En particular, los mayores saltos se observan con el clasificador *CatBoost*, donde:

- **G-Mean** en CatBoost pasa de 0.145 a **0.496** , más del triple de su valor inicial.
- **F1-Score** en CatBoost mejora de 0.067 a **0.173**, destacando especialmente en un entorno PU.
- **AUC-PR** en BRF sube de 0.125 a **0.164** (†), una ganancia significativa dado el contexto reducido de características.

Tal como refleja la Figura 6.3, la mejora en **G-Mean** alcanza su punto máximo con apenas un 5% de las características, lo que evidencia que una buena selección de variables puede reducir el ruido y estabilizar el aprendizaje, incluso en condiciones adversas.

Es importante destacar que, a diferencia de los conjuntos GO y PathDIP —de naturaleza más categórica—, GTEx se basa en variables continuas. Aunque se aplicó discretización para homogeneizar el tratamiento, esta transformación no fue suficiente por sí sola para mejorar el rendimiento. Sin embargo, **Fast-mRMR logra superar esta barrera**, lo que sugiere su robustez ante distintos tipos de representación de datos.

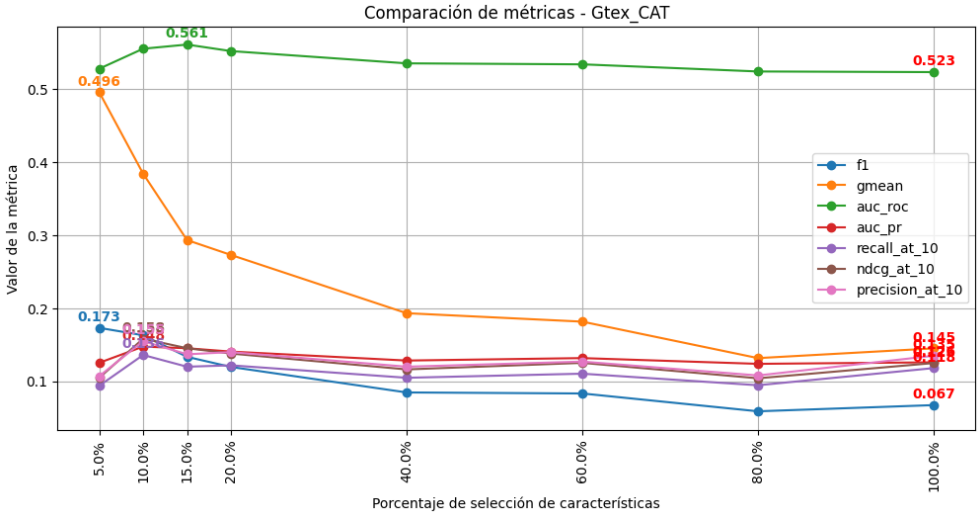


Figura 6.3: Evolución de las métricas sobre el conjunto GTEx con CatBoost al aplicar Fast-mRMR. El 100% representa el modelo sin selección (baseline original).

Aunque los valores absolutos siguen siendo moderados debido a la dificultad intrínseca del conjunto, el enfoque propuesto demuestra una mejora sólida, consistente y estadísticamente significativa. Esto refuerza su utilidad como herramienta de preprocesamiento, incluso en dominios con información limitada y estructura débil.

6.2 Evaluación sobre conjuntos combinados

Esta sección analiza el rendimiento del pipeline propuesto al aplicar *Fast-mRMR* sobre configuraciones combinadas de conjuntos de datos, específicamente **Gene Ontology (GO)** y **PathDIP**. A diferencia de trabajos anteriores como el de Vega Magdaleno et al. [1], donde la combinación directa de conjuntos empeoraba el rendimiento debido a la alta dimensionalidad y ruido, aquí se analiza si la selección de características puede contrarrestar dicho efecto negativo.

Se consideran dos estrategias de combinación, siempre tomando como base los **genes presentes en ambos conjuntos (intersección)**:

- **Fusión de características (GO + PathDIP)**: se combinan las características de ambos conjuntos funcionales en un único vector por gen.
- **Conjuntos individuales (solo GO o solo PathDIP)**: se utilizan únicamente las características de uno de los dos conjuntos, restringidas a los genes comunes.

Ambas configuraciones se evalúan utilizando los clasificadores *CatBoost* y *Balanced Random Forest (BRF)*, y se comparan en cuanto a rendimiento predictivo y simplicidad del modelo.

6.2.1 Comparativa de configuraciones funcionales combinadas

La Tabla 6.3 resume los mejores resultados obtenidos en cada caso, reportando *F1-score*, *G-Mean*, *AUC-ROC* y *AUC-PR*. En todos los experimentos se utilizó el conjunto de genes comunes entre GO y PathDIP, variando solo el conjunto de características utilizadas.

Cuadro 6.3: Comparación del rendimiento predictivo al combinar características de **GO** y **PathDIP** frente a su uso por separado, utilizando dos clasificadores. Los resultados se expresan como *media ± desviación típica*, e incluyen entre paréntesis el porcentaje de características seleccionadas en cada configuración. En negrita se indican los mejores resultados por métrica. El mejor valor en la configuración combinada se compara con las otras cinco configuraciones mediante un *t-test* pareado bilateral ($p < 0.05$). El símbolo † indica que la diferencia es estadísticamente significativa frente al resto de combinaciones en esa métrica.

Configuración de características	Clasificador	*F1-score*	*G-Mean*	*AUC-ROC*	*AUC-PR*
GO + PathDIP (fusionadas)	BRF	0.503 ± 0.010 (5%)	**0.798 ± 0.007** † (5%)	0.869 ± 0.007 (10%)	0.580 ± 0.018 (10%)
GO + PathDIP (fusionadas)	CAT	**0.560 ± 0.023** (5%)	0.750 ± 0.018 (5%)	**0.873 ± 0.003** (5%)	**0.608 ± 0.018** † (5%)
GO (solo)	BRF	0.485 ± 0.020 (5%)	0.777 ± 0.015 (5%)	0.841 ± 0.008 (5%)	0.513 ± 0.019 (5%)
PathDIP (solo)	BRF	0.472 ± 0.015 (5%)	0.781 ± 0.010 (40%)	0.835 ± 0.008 (40%)	0.569 ± 0.018 (15%)
GO (solo)	CAT	0.533 ± 0.024 (10%)	0.728 ± 0.019 (2.5%)	0.858 ± 0.010 (5%)	0.551 ± 0.015 (5%)
PathDIP (solo)	CAT	0.526 ± 0.025 (10%)	0.736 ± 0.010 (25%)	0.828 ± 0.012 (60%)	0.578 ± 0.016 (25%)

Los resultados reflejados en la Tabla 6.3 muestran que la combinación de características funcionales de GO y PathDIP, aplicada exclusivamente sobre genes comunes, permite mejorar el

rendimiento predictivo frente al uso de cada conjunto por separado. Este hallazgo contrasta con trabajos anteriores, como el de Magdaleno [1], donde la fusión directa de conjuntos funcionales —sin aplicar ningún método de selección— tendía a degradar la calidad del modelo debido a la alta dimensionalidad y a la introducción de ruido irrelevante.

Gracias al uso de *Fast-mRMR*, es posible reducir la dimensionalidad conservando solo las características más relevantes, lo cual permite explotar de forma eficaz la complementariedad entre fuentes funcionales. En este escenario, se observan mejoras aparentes en todas las métricas, aunque solo dos de ellas alcanzan **significancia estadística** ($p < 0.05$):

- **BRF**: se alcanza el mayor valor de **G-Mean** (**0.798**), con una diferencia estadísticamente significativa ($p < 0.05$) respecto a las otras cinco configuraciones evaluadas para esta métrica.
- **CatBoost**: se obtiene el valor más alto en **AUC-PR** (**0.608**), también con significancia estadística frente a todas las configuraciones alternativas en esa misma métrica.

En contraste, aunque el **F1-score** y el **AUC-ROC** presentaron sus valores más altos en la configuración combinada, estas diferencias no fueron estadísticamente significativas en las comprobaciones realizadas. Aun así, la tendencia general sigue siendo favorable para el enfoque combinado.

Este comportamiento respalda la hipótesis de que una selección adecuada de características puede potenciar sinergias entre fuentes funcionales, especialmente en contextos con ruido y alta dimensionalidad. Además, en tareas con datos *Positive-Unlabeled (PU)*, donde las métricas pueden estar sesgadas por la falta de negativos fiables, una comparación relativa como esta —basada en tests estadísticos pareados— ofrece una validación más robusta del beneficio obtenido.

6.2.2 Influencia del número de características en la configuración combinada

Las Figuras 6.4 y 6.5 ilustran la evolución de las métricas clave al aplicar *Fast-mRMR* sobre la configuración combinada. Se observa un comportamiento estable y robusto a partir del 5% de características seleccionadas.

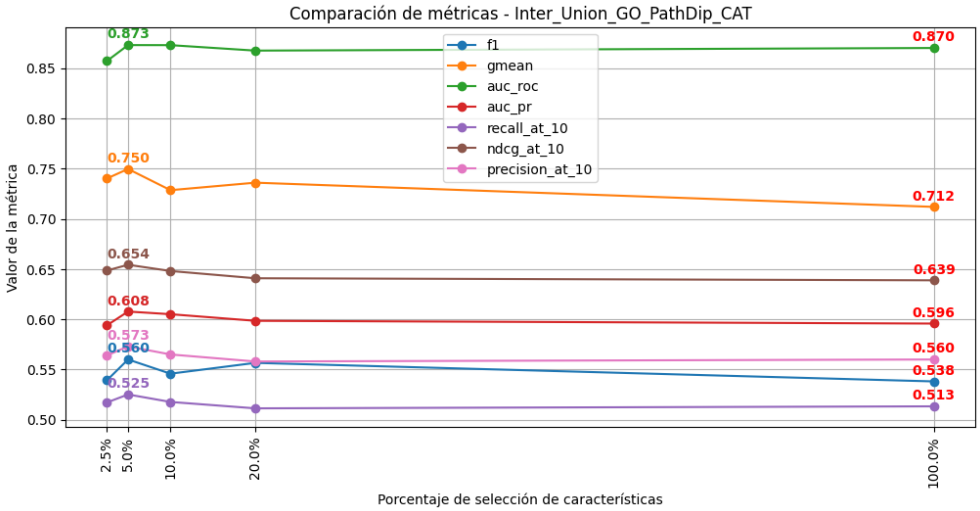


Figura 6.4: Rendimiento del clasificador CatBoost sobre la configuración de unión de características (GO + PathDIP).

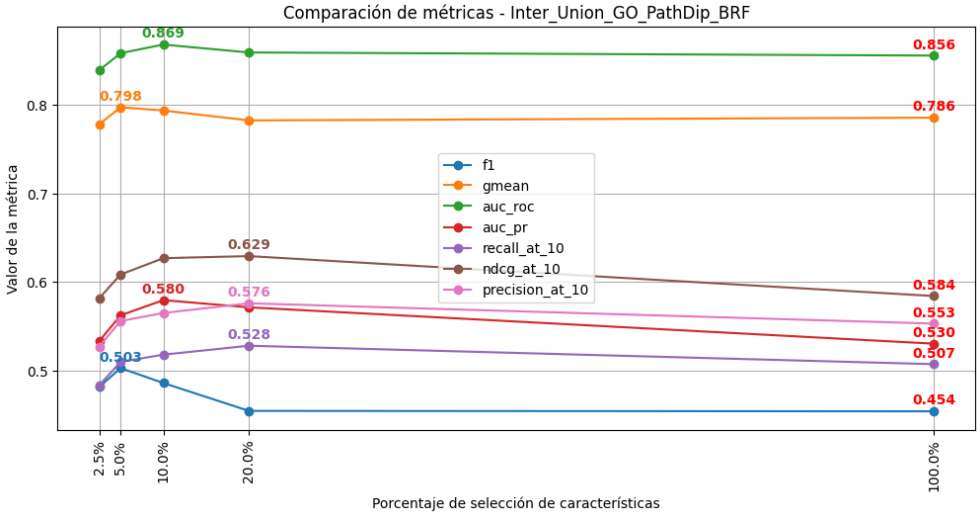


Figura 6.5: Rendimiento del clasificador BRF sobre la configuración de unión de características (GO + PathDIP).

Ambas gráficas refuerzan la conclusión de que es posible **mejorar el rendimiento incluso con menos información**, siempre que esta sea más relevante y de mayor calidad.

6.2.3 Discusión final sobre conjuntos combinados

En el trabajo original [1], ya se indicaba que combinar conjuntos como GO y PathDIP no mejoraba el rendimiento si no se realizaba una selección adecuada. De hecho, el apilamiento de características empeoraba las predicciones por la alta dimensionalidad y el ruido.

En contraste, los resultados presentados aquí demuestran que:

- La combinación de conjuntos **sí es efectiva** si se acompaña de una estrategia de reducción previa como *Fast-mRMR*.
- Se extrae información complementaria entre GO y PathDIP, mejorando los resultados obtenidos por separado.
- La reducción de complejidad computacional, al trabajar con menos características relevantes, facilita pipelines más eficientes y reproducibles.

Por tanto, esta sección refuerza uno de los mensajes clave de este trabajo: no es solo el modelo lo que importa, sino también la calidad y la selección de las características utilizadas. En problemas de priorización génica con datos *Positive-Unlabeled*, esta combinación puede marcar la diferencia entre modelos funcionales y modelos inútiles.

6.3 Evaluación sobre conjuntos de alta dimensionalidad

En esta sección se analiza el comportamiento del pipeline propuesto al aplicarlo sobre un conjunto especialmente complejo: uno basado en **coexpresión génica**, derivado de perfiles transcriptómicos. Este tipo de datos plantea un reto importante por su extrema dimensionalidad (más de 44.000 características) y bajo nivel de estructuración, lo que históricamente ha dificultado la obtención de buenos resultados [1].

Debido al coste computacional cuadrático de *Fast-mRMR*, se aplicó una estrategia de **bagging** de características —agrupamiento aleatorio y reducción dimensional previa— para estabilizar la ejecución del algoritmo. Además, al tratarse de datos continuos, se realizó una **discretización** previa para asegurar la correcta ejecución del método.

6.3.1 Resultados y análisis sobre Coexpression

El conjunto de coexpresión representa un entorno especialmente complejo por su extremada dimensionalidad y escasa estructuración. A pesar de ello, los resultados obtenidos tras aplicar *Fast-mRMR* son prometedores.

La Tabla 6.4 muestra una comparativa directa entre el rendimiento del modelo original (sin selección de características) y nuestro enfoque basado en *Fast-mRMR*, empleando dos clasificadores: BRF y CatBoost. Aunque los valores absolutos de las métricas son inferiores respecto a conjuntos más estructurados como GO o PathDIP, se aprecian mejoras consistentes tras aplicar la selección.

En particular, se observan incrementos claros en **AUC-ROC** y **AUC-PR**, con ligeras mejoras también en **F1-score**, especialmente al emplear BRF. Esto demuestra que, incluso en un escenario tan adverso, la reducción de dimensionalidad puede traducirse en beneficios reales de rendimiento.

Cuadro 6.4: Comparación del rendimiento predictivo en el conjunto **Coexpression** entre el método original sin selección y nuestro enfoque basado en *Fast-mRMR*, evaluados con dos clasificadores. Los resultados se presentan como *media ± desviación típica*, y en el caso del método propuesto se incluye entre paréntesis el porcentaje de características seleccionadas. En negrita se destacan los mejores resultados por métrica. El símbolo † indica que la diferencia respecto al modelo base (sin selección) es **estadísticamente significativa** ($p < 0.05$), según un *t-test* pareado bilateral.

Método	**Clasificador**	***AUC-ROC***	***AUC-PR***	***G-Mean***	***F1-Score***
Original (sin selección)	BRF	0.504 ± 0.015	0.126 ± 0.006	0.502 ± 0.017	0.179 ± 0.010
	CatBoost	0.510 ± 0.021	0.130 ± 0.007	0.070 ± 0.029	0.038 ± 0.016
Fast-mRMR (propuesto)	BRF	0.542 ± 0.012 (2.5%)	0.150 ± 0.036 (2.5%)	**0.512** ± 0.031 (2.5%)	**0.185** ± 0.013 (2.5%)
	CatBoost	**0.547** ± 0.020 (5%)	**0.153** ± 0.036 (2.5%)	0.065 ± 0.009 (10%)	0.035 ± 0.005 (10%)

Como se observa en la Tabla 6.4, el enfoque **Fast-mRMR** obtiene los mejores valores en varias métricas en comparación con el modelo original sin selección de características, en particular con **BRF para F1-score y G-Mean**, y con **CatBoost para AUC-ROC y AUC-PR**.No obstante, tras realizar un *t-test* pareado bilateral para cada métrica —comparando el mejor valor con los otros tres resultados de la misma columna (tres pruebas por métrica)—, **no se encontraron diferencias estadísticamente significativas** ($p \geq 0.05$).

Aunque estas mejoras no alcanzan significación estadística, sí reflejan una tendencia consistente a favor de *Fast-mRMR*, lo que sugiere que su aplicación puede ser beneficiosa incluso en conjuntos de datos complejos y con alto número de características como **Coexpression**.

Para analizar en mayor profundidad el efecto del número de características seleccionadas, se presentan a continuación las gráficas correspondientes a cada clasificador. En ellas se visualiza cómo evolucionan las principales métricas a medida que se incrementa el porcentaje de selección.

Clasificador: BRF. En la Figura 6.6 se observa un patrón claro: las mejoras más notables se concentran al utilizar únicamente entre el 1.5% y el 5% de las características. En ese intervalo se alcanza el mejor equilibrio entre simplicidad y rendimiento, mejorando tanto *F1-Score* como *G-Mean* respecto al modelo base.

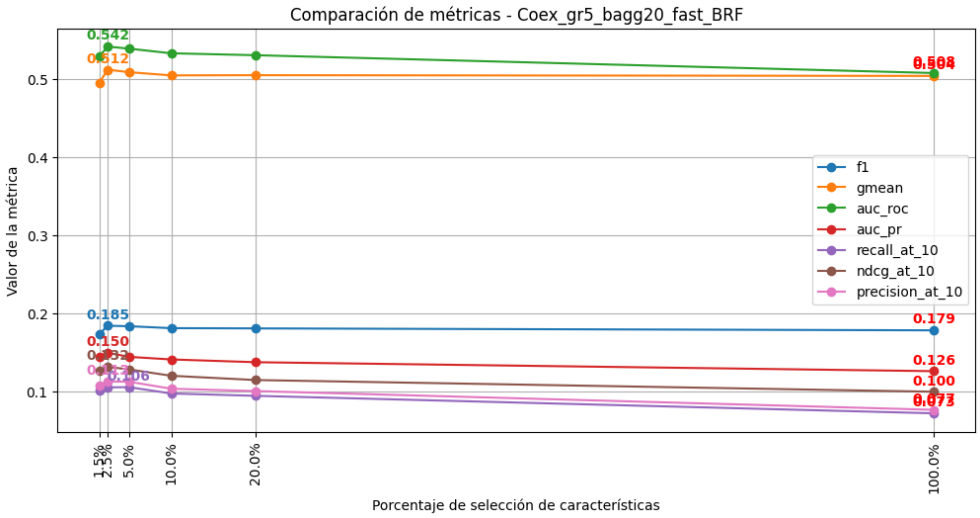


Figura 6.6: Evolución de las métricas clave al aplicar *Fast-mRMR* sobre el conjunto de coexpresión con el clasificador BRF.

Clasificador: CatBoost. La Figura 6.7 refleja un comportamiento similar. Aunque el clasificador CatBoost obtiene valores absolutos más bajos —especialmente en *G-Mean*—, sí se registran incrementos progresivos en *AUC-PR* y *AUC-ROC* tras aplicar *Fast-mRMR*, lo que confirma que el modelo se beneficia de una representación más compacta.

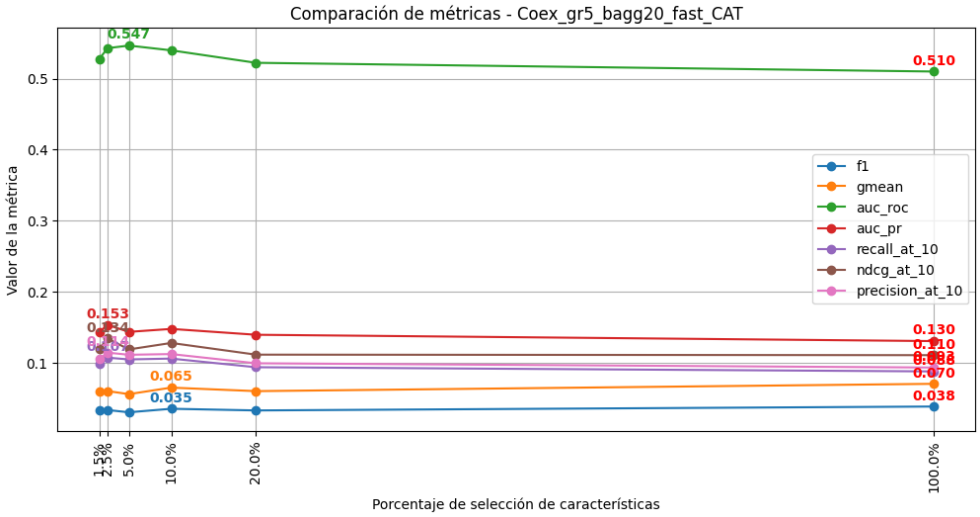


Figura 6.7: Evolución de las métricas clave al aplicar *Fast-mRMR* sobre el conjunto de coexpresión con el clasificador CatBoost.

En conjunto, estos resultados confirman que, incluso en condiciones desfavorables, la selección de características aporta mejoras tangibles. Es especialmente relevante destacar que se consigue:

- Mejorar sistemáticamente el rendimiento frente al baseline sin selección.
- Reducir el número de variables utilizadas en más de un 98%, lo que disminuye drásticamente el coste computacional.
- Superar los resultados de estudios previos [1], donde este conjunto no mostraba mejoras sustanciales ni siquiera tras procesos de discretización.

Por tanto, este estudio reafirma la utilidad de *Fast-mRMR* como herramienta robusta incluso en dominios complejos como la coexpresión génica. Su capacidad para identificar subconjuntos relevantes dentro de una estructura caótica y ruidosa demuestra su potencial como etapa clave en pipelines de priorización génica en contextos biomédicos de alta dimensionalidad.

6.4 Evaluación de la combinación de selección de características y de instancias: Fast-mRMR + PU Learning

En esta sección se evalúa el impacto de combinar dos estrategias de preprocesamiento: la selección de características mediante *Fast-mRMR* y la selección de instancias basada en *Positive-Unlabeled (PU) Learning*. El objetivo es comprobar si esta combinación mejora el rendimiento predictivo respecto a utilizar cada técnica de forma aislada o frente al modelo original sin tratamiento previo. Los experimentos se realizaron sobre los conjuntos funcionales **GO** y **PathDIP**, aplicando ambos enfoques con los clasificadores *Balanced Random Forest (BRF)* y *CatBoost*.

6.4.1 Comparativa de enfoques: Fast mRMR + PU learning vs enfoque propuesto vs trabajos previos

La Tabla 6.5 recoge los resultados obtenidos al aplicar la combinación de *Fast-mRMR* con *PU Learning*, comparados con el enfoque base sin selección, el método de Jorge Paz [1] y nuestro método propuesto sin PU. Para cada métrica, se selecciona el mejor valor en el bloque de **Fast-mRMR + PU** y se compara estadísticamente con el resto de configuraciones de esa misma métrica (catorce comprobaciones por métrica). Se aplica un *t-test* pareado bilateral ($p < 0.05$), indicando con el símbolo † los casos donde la diferencia es estadísticamente significativa.

Cuadro 6.5: Comparación del rendimiento predictivo entre el método original, PU Learning, Fast-mRMR y Fast-mRMR + PU. Los resultados se expresan como *media ± desviación típica*. En los métodos **Fast-mRMR** y **Fast-mRMR + PU** se indica entre paréntesis el porcentaje de características seleccionadas. En negrita se destacan los mejores resultados por métrica. El símbolo † indica que el mejor valor del bloque **Fast-mRMR + PU** presenta una diferencia estadísticamente significativa ($p < 0.05$) frente al resto de configuraciones en esa métrica.

Método	**Conjunto / Clasificador**	***AUC-ROC***	***AUC-PR***	***G-Mean***	***F1-Score***
Original (sin selección)	PathDIP / CAT	0.813 ± 0.010	0.557 ± 0.011	0.702 ± 0.012	0.505 ± 0.015
	PathDIP / BRF	0.827 ± 0.007	0.563 ± 0.013	0.771 ± 0.011	0.476 ± 0.013
	GO / CAT	0.838 ± 0.012	0.499 ± 0.020	0.684 ± 0.015	0.498 ± 0.020
	GO / BRF	0.835 ± 0.011	0.429 ± 0.008	0.763 ± 0.010	0.389 ± 0.010
PU Learning (Jorge Paz et al.)	PathDIP / CAT	0.829 ± 0.011	0.526 ± 0.020	0.749 ± 0.007	0.531 ± 0.011
	PathDIP / BRF	0.827 ± 0.006	0.549 ± 0.016	0.768 ± 0.010	0.461 ± 0.011
	GO / CAT	0.839 ± 0.012	0.438 ± 0.015	0.729 ± 0.011	0.496 ± 0.007
	GO / BRF	0.829 ± 0.011	0.390 ± 0.008	0.762 ± 0.011	0.380 ± 0.009
Fast-mRMR (propuesto)	PathDIP / CAT	0.830 ± 0.009 (80%)	**0.588 ± 0.015 †** (17.5%)	0.732 ± 0.010 (17.5%)	0.523 ± 0.020 (45%)
	PathDIP / BRF	0.841 ± 0.006 (45%)	0.562 ± 0.016 (80%)	**0.779 ± 0.009 †** (50%)	0.479 ± 0.016 (2.5%)
	GO / CAT	0.852 ± 0.009 (5%)	0.533 ± 0.022 (5%)	0.738 ± 0.016 (5%)	**0.532 ± 0.014** (5%)
	GO / BRF	0.842 ± 0.009 (40%)	0.479 ± 0.015 (5%)	0.766 ± 0.007 (5%)	0.450 ± 0.010 (5%)
Fast-mRMR + PU	PathDIP / CAT	0.834 ± 0.011 (20%)	0.532 ± 0.016 (20%)	0.768 ± 0.010 (40%)	0.525 ± 0.017 (40%)
	PathDIP / BRF	0.832 ± 0.006 (40%)	0.523 ± 0.022 (10%)	0.772 ± 0.007 (40%)	0.462 ± 0.014 (5%)
	GO / CAT	**0.860 ± 0.010** (5%)	0.483 ± 0.020 (5%)	0.759 ± 0.010 (5%)	0.522 ± 0.010 (5%)
	GO / BRF	0.831 ± 0.009 (5%)	0.454 ± 0.018 (5%)	0.764 ± 0.007 (5%)	0.437 ± 0.008 (5%)

Del análisis se desprenden las siguientes observaciones clave:

- **Fast-mRMR de forma individual** proporciona los mejores resultados en la mayoría de escenarios. Esto se cumple particularmente en PathDIP con ambos clasificadores y en GO con BRF, donde supera tanto al modelo base como al enfoque PU.
- La combinación **Fast-mRMR + PU Learning** no aporta mejoras sistemáticas. Si bien los resultados son competitivos, no se observan beneficios generalizados respecto a aplicar Fast-mRMR por separado.

- El único caso donde la combinación **Fast-mRMR + PU** alcanza el mayor valor absoluto es en **GO con CatBoost**, con un *AUC-ROC* de **0.860**. No obstante, tras realizar 14 comprobaciones estadísticas en esa métrica —comparando dicho valor con el resto de configuraciones de la columna—, no se identificaron diferencias **estadísticamente significativas** ($p \geq 0.05$). Por tanto, esta mejora debe interpretarse con cautela, ya que podría deberse al azar. Aun así, el resultado apunta a un posible comportamiento prometedor que podría explorarse más a fondo en trabajos futuros, ajustando mejor los hiperparámetros o utilizando modelos más sofisticados.

6.4.2 Influencia del número de características en GO con Fast-mRMR + PU Learning

Las Figuras 6.8 y 6.9 muestran la evolución de métricas al aplicar Fast-mRMR + PU sobre el conjunto GO. En general, las curvas confirman que existe un crecimiento en ciertas métricas con respecto a usar solo PU learning, pero realmente no hay una mejora clara y diferencial con respecto a usar Fast-mRMR en solitario.

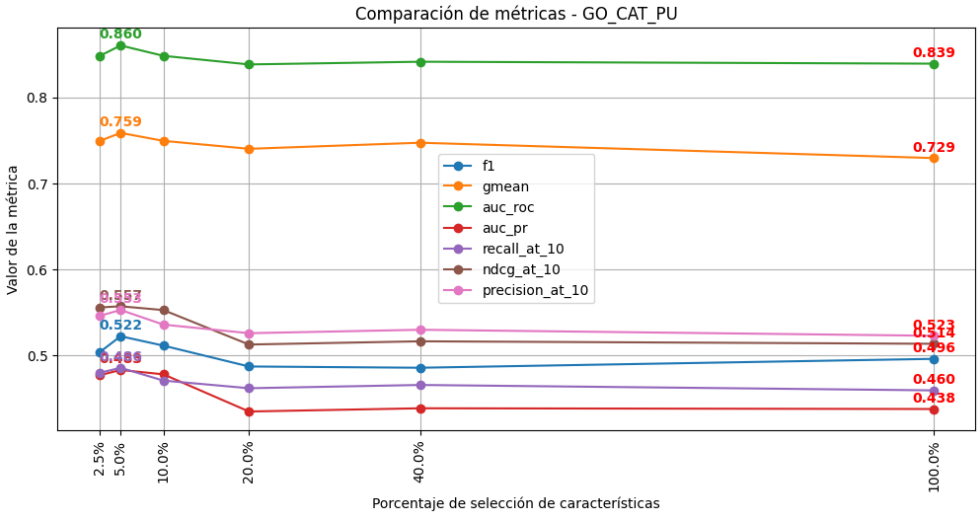


Figura 6.8: Evolución de métricas en el conjunto GO con CatBoost al aplicar Fast-mRMR + PU Learning.

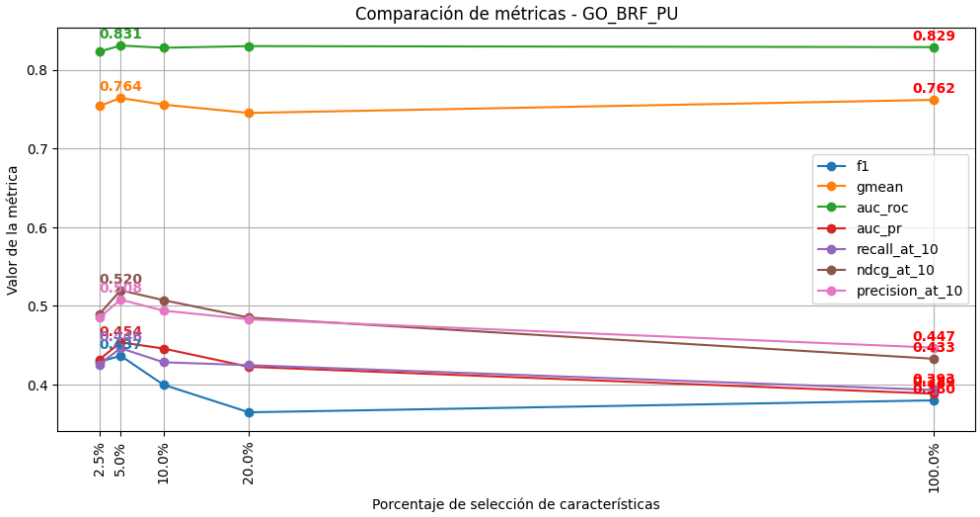


Figura 6.9: Evolución de métricas en el conjunto GO con BRF al aplicar Fast-mRMR + PU Learning.

Desde un punto de vista metodológico, los resultados indican que la combinación de Fast-mRMR con PU Learning no siempre es beneficiosa. Esto puede deberse a varios factores:

- **Falta de ajuste fino** en los parámetros del algoritmo PU, que podría limitar su contribución al modelo combinado.
- **Incompatibilidad entre instancias seleccionadas por PU** y las características priorizadas por Fast-mRMR, lo que puede introducir ruido o sesgo en el proceso de aprendizaje.
- **Complejidad añadida**, que no siempre se traduce en mejora, especialmente si no hay un mecanismo de coordinación entre selección de instancias y selección de variables.

Pese a ello, la integración no degrada drásticamente el rendimiento, y abre la puerta a explorar arquitecturas híbridas más sofisticadas.

Capítulo 7

Conclusiones

A lo largo de este trabajo se ha planteado un enfoque centrado en la selección de características como herramienta clave para mejorar el rendimiento predictivo en problemas de clasificación con datos biomédicos de alta dimensionalidad. En este tipo de contextos, donde la complejidad de los datos puede dificultar la generalización de los modelos, reducir el número de características permite no solo simplificar el proceso de entrenamiento, sino también aumentar la eficiencia computacional y mitigar el impacto del ruido o la redundancia en los datos.

En particular, se ha aplicado el algoritmo *Fast-mRMR* como técnica de selección dentro de un pipeline diseñado para la priorización de genes asociados a restricción dietética (DR). Los resultados obtenidos muestran mejoras consistentes respecto a enfoques previos, tanto en rendimiento predictivo como en simplicidad del modelo. En especial, conjuntos funcionales como GO y PathDIP han mostrado una mejora clara, y se ha comprobado que incluso fuentes más complejas como GTEx o los datos de coexpresión pueden beneficiarse de una estrategia de selección adecuada. La aplicación combinada de fuentes funcionales (GO + PathDIP) ha reforzado esta conclusión, mostrando que una fusión bien diseñada —junto a un proceso de filtrado informativo— puede superar limitaciones estructurales observadas en estudios anteriores.

En definitiva, este trabajo subraya que una IA más eficiente y robusta no depende únicamente del modelo de clasificación, sino de cómo se prepara y selecciona la información de entrada. La combinación adecuada de reducción de dimensionalidad y selección de características se consolida como una herramienta clave para abordar problemas biomédicos complejos. Además, trabajar con un subconjunto de características más relevantes implica una mayor interpretabilidad de los resultados, lo que permite a investigadores y expertos centrar su análisis en los factores con mayor influencia biológica o clínica.

7.1 Competencias adquiridas

Durante el desarrollo del proyecto, se han adquirido y afianzado numerosas competencias técnicas y metodológicas. En particular, cabe destacar:

- La capacidad para diseñar y evaluar experimentalmente pipelines de aprendizaje automático aplicados a datos reales, con un enfoque crítico sobre la calidad de los datos y la evaluación.
- La comprensión profunda de métodos de selección de características, técnicas de aprendizaje *PU*, y validación cruzada anidada como forma robusta de estimar el rendimiento.
- El manejo de herramientas avanzadas para el procesamiento de datos biomédicos, incluyendo preprocesamiento, discretización, bagging y análisis visual de resultados.
- La aplicación de metodologías ágiles de trabajo (como Kanban y Scrum), la organización modular del código y la documentación rigurosa del proceso.
- Además, se ha mejorado la capacidad de redactar textos científicos, elaborar figuras interpretables y extraer conclusiones fundamentadas sobre resultados experimentales.

Desde una perspectiva más personal, el trabajo ha supuesto una oportunidad para enfrentarse a un problema de investigación real, aprender a desenvolverse con grandes volúmenes de datos y evaluar críticamente la utilidad de cada técnica desde un punto de vista práctico y aplicado.

7.2 Trabajo futuro

Como líneas futuras de trabajo, se identifican múltiples direcciones prometedoras:

- **Optimización de la combinación Fast mRMR + PU:** Aunque la integración conjunta no ha producido mejoras sistemáticas, sería interesante explorar ajustes más finos de hiperparámetros o emplear clasificadores más adaptativos que integren de forma más eficaz ambas estrategias.
- **Exploración de otros métodos de selección:** Podrían compararse los resultados de Fast-mRMR con otros enfoques de selección multivariada, para evaluar su estabilidad y robustez.
- **Incorporación de nuevas fuentes funcionales:** Ampliar el estudio a otros tipos de datos ómicos (proteómica, epigenética) permitiría comprobar la generalización del enfoque propuesto.

- **Evaluación de interpretabilidad:** Incluir análisis de interpretabilidad para entender mejor el impacto biológico de las características seleccionadas y su relación con el gen objetivo.
- **Medición de complejidad computacional:** Un estudio detallado del coste de cada etapa permitiría optimizar aún más la eficiencia del pipeline, especialmente para entornos de computación limitada.

En conjunto, este trabajo sienta las bases para una metodología más informada, robusta y eficiente en el ámbito de la priorización génica y puede servir como referencia para futuras investigaciones en el campo del aprendizaje automático aplicado a datos biomédicos.

Bibliografía

[1] G. D. Vega Magdaleno, V. Bespalov, Y. Zheng, A. A. Freitas, and J. P. de Magalhães, "Machine learning-based predictions of dietary restriction associations across ageing-related genes," *BMC Bioinformatics*, vol. 23, no. 1, p. 10, 2022.

[2] L. Fontana and L. Partridge, "Promoting health and longevity through diet: from model organisms to humans," *Cell*, vol. 161, no. 1, pp. 106–118, 2015.

[3] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[4] S. Ramírez-Gallego, I. Lastra, D. Martínez-Rego, V. Bolón-Canedo, J. M. Benítez, F. Herrera, and A. Alonso-Betanzos, "Fast-mrmr: Fast minimum redundancy maximum relevance algorithm for high-dimensional big data," *International Journal of Intelligent Systems*, vol. 32, no. 2, pp. 134–152, 2017.

[5] J. Bekker and J. Davis, "Learning from positive and unlabeled data: A survey," *Machine Learning*, vol. 109, no. 4, pp. 719–760, 2020.

[6] J. Paz-Ruza, A. A. Freitas, A. Alonso-Betanzos, and B. Guijarro-Berdiñas, "Positive-unlabelled learning for identifying new candidate dietary restriction-related genes among ageing-related genes," *Computers in Biology and Medicine*, vol. 180, p. 108999, 2024.

[7] H. Yang, P. N. Robinson, and K. Wang, "Gene prioritization for complex traits using deep neural network," *BMC Bioinformatics*, vol. 19, no. 1, pp. 1–13, 2018.

[8] V. Kumar and S. Minz, "Feature selection: A literature review," *SmartCR*, vol. 4, no. 3, pp. 211–229, 2014.

[9] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, J. A. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig *et al.*, "Gene ontology: tool for the unification of biology," *Nature genetics*, vol. 25, no. 1, pp. 25–29, 2000.

[10] P. Rahmati, M. Abovsky, C. Pastrello, and I. Jurisica, "Pathdip: an annotated resource for known and predicted human gene-pathway associations and pathway enrichment analysis," *Nucleic Acids Research*, vol. 45, no. D1, pp. D419–D426, 2017.

[11] G. Consortium, "The genotype-tissue expression (gtex) project," *Nature Genetics*, vol. 45, no. 6, pp. 580–585, 2013.

[12] M. Kanehisa and et al., "Kegg: new perspectives on genomes, pathways, diseases and drugs," *Nucleic Acids Research*, vol. 45, no. D1, pp. D353–D361, 2017.

[13] D. Szklarczyk and et al., "String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets," *Nucleic Acids Research*, vol. 47, no. D1, pp. D607–D613, 2019.

[14] J. Wu and et al., "Evaluation of feature selection methods using bagging and boosting ensemble techniques on high throughput biological data," in *Proceedings of the 12th International Conference on Bioinformatics and Biomedical Technology (ICBET 2020)*. ACM, 2020, pp. 170–175, available at: https://doi.org/10.1145/3397391.3397403.

[15] X. Z. Fern and C. E. Brodley, "Random projection for high dimensional data clustering: A cluster ensemble approach," in *ICML*, 2003.

[16] S. Varma and R. Simon, "Bias in error estimation when using cross-validation for model selection," *BMC bioinformatics*, vol. 7, no. 1, p. 91, 2006.

[17] G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *Journal of Machine Learning Research*, vol. 11, pp. 2079–2107, 2010.

[18] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, 2002.

[19] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.

[20] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[21] C. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data," University of California, Berkeley, Tech. Rep. Technical Report 666, 2004. [Online]. Available: https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf

[22] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: gradient boosting with categorical features support," in *NeurIPS*, 2018.

[23] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[24] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[25] V. Gligorijević, M. Barot, and R. Bonneau, "Deepnf: deep network fusion for protein function prediction," *Bioinformatics*, vol. 34, no. 22, pp. 3873–3881, 2018.

[26] C. Elkan and K. Noto, "Learning classifiers from only positive and unlabeled data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 213–220.