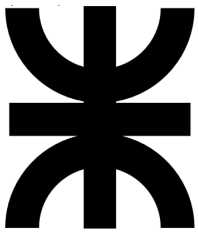

Universidad Tecnológica Nacional

Tesis de Maestría

Generación de datos sintéticos para selección de características con algoritmos genéticos

Autor:
Claudio Sebastian Castillo

Directores:
Matías Gerard y Leandro Vignolo

Tesis presentada en cumplimiento de los requisitos para el grado de Maestría en Minería de Datos

en

UTN
Seccional Paraná

Septiembre, 2024

A mi corazón de 4/4; Morella, Sofía, Joquín y Manuel. A mi musa Verónica.

UNIVERSIDAD TECNOLÓGICA NACIONAL

Seccional Paraná

Maestría en Minería de Datos

Resumen

Generación de datos sintéticos para selección de características con algoritmos genéticos

Claudio Sebastian Castillo

La disponibilidad de datos muestrales afecta los procesos de selección de características, y resulta particularmente condicionante en escenarios de alta dimensionalidad y bajo número de muestras. En el caso de selección de características mediante Algoritmos Genéticos la falta de datos impacta negativamente en la función de aptitud, y de esa forma limita la eficacia del algoritmo. Por eso, la técnica de aumentación de datos mediante Autocodificadores Variacionales plantea una posible solución a este problema, ofreciendo distintas alternativas de implementación en el contexto de los Algoritmos Genéticos.

Introducción

En el campo del aprendizaje automático, la selección de características es una tarea crítica que puede determinar el éxito o fracaso de un modelo predictivo. La alta dimensionalidad y la complejidad inherente a los conjuntos de datos reales hacen que la selección de un subconjunto óptimo de características sea, muchas veces, un paso ineludible para el aprendizaje efectivo.

En este contexto, los Algoritmos Genéticos (AGs) se han consolidado como una herramienta poderosa para resolver problemas de optimización complejos, incluida la selección de características. Estos algoritmos, inspirados en la evolución natural, son capaces de explorar grandes espacios de búsqueda de manera efectiva, proporcionando soluciones cercanas al óptimo en una variedad de escenarios. Por ello, los AGs han sido ampliamente utilizados en problemas de selección, demostrando su eficacia en la identificación de subconjuntos relevantes de características en datos de alta dimensionalidad.

Sin embargo, la eficacia de los AGs depende de la disponibilidad de suficientes datos para evaluar las soluciones en competencia. En contextos donde los datos son limitados, los AGs pueden verse afectados en su capacidad discriminativa, produciendo soluciones subóptimas o inestables. Esta limitación es especialmente crítica en problemas de alta dimensionalidad y bajo número de muestras, donde la función objetivo que guía la búsqueda de soluciones puede degradarse significativamente.

Por esta razón, la investigación de estrategias que mitiguen las limitaciones impuestas por la escasez de datos se ha convertido en un área de interés creciente en el subcampo de la selección de características. Una de las técnicas emergentes en este ámbito es la aumentación de datos mediante Autoencodificadores Variacionales (AVs). Los AVs, como modelos generativos, tienen la capacidad de crear muestras sintéticas que mantienen las propiedades fundamentales de los datos originales, conviertiéndolos en una herramienta prometedora para mejorar la capacidad de los AGs en la selección de características.

El problema central de la tesis que aquí presentamos gira, precísamente, en la restricciones que la escacez de datos impone a los AGs, y cómo superarlas usando AVs. Así, la pregunta central del trabajo es: ¿cómo puede la aumentación de datos mediante autoencodificadores variacionales mejorar el desempeño de los algoritmos genéticos en la selección de características?

Cabe destacar que este desafío y su eventual solución son importantes por varias razones. La selección de características no solo condiciona, como ya se mencionó, la precisión de los modelos predictivos, también afecta la eficiencia computacional y la interpretabilidad de los resultados. En problemas de alta dimensionalidad, la posibilidad de reducir el número de características relevantes sin perder información útil puede marcar la diferencia entre un modelo efectivo y uno ineficaz, entre uno interpretable y uno de caja negra. Por lo tanto, mejorar este proceso mediante la integración de técnicas de aumentación de datos puede tener un impacto significativo en diversas aplicaciones prácticas, desde la biología molecular hasta la ingeniería y las ciencias sociales.

La hipótesis que hemos llevado a prueba ha sido que la aumentación de datos mediante AVs mejora la capacidad discriminativa de los AGs, permitiendo la identificación de subconjuntos de características más relevantes y estables en contextos de escasez muestral. Para evaluarla, hemos propuesto un trabajo experimental que exploró la

integración de estas dos técnicas en un marco unificado. La idea de combinar la generación de datos sintéticos mediante AVs con la selección de características mediante AGs, estaba orientada a buscar combinaciones sinérgicas y eficaces entre modelos que mejorasen la selección de características. A estos fines, trabajamos con cinco conjuntos de datos de referencia, representativos de distintos contextos y niveles de complejidad, para evaluar el desempeño de los modelos propuestos.

A lo largo de este documento, describiremos el proceso de investigación llevado a cabo, desde los estudios iniciales hasta los experimentos finales, pasando por el diseño y construcción de un modelo genérico de AV, y su adaptación a los datasets elegidos y la creación de una estructura combinada de AV + AG para la selección de características. Los resultados obtenidos en cada etapa se analizarán y discutirán en detalle, con el objetivo de identificar las ventajas y limitaciones de la propuesta, así como posibles áreas de mejora y futuros trabajos.

Al finalizar el documento, esperamos poder justificar la eficacia de la aumentación de datos mediante AVs en la selección de características, demostrando que esta técnica puede mejorar significativamente el desempeño de los AGs en contextos de escasez. Además, esperamos identificar las condiciones y contextos en los que esta técnica es más efectiva.

El documento está estructurado de la siguiente manera: en el capítulo 1 se presenta el problema de la selección de características en contextos de alta dimensionalidad, desbalance y escacez de datos. En el capítulo 2 se hace una revisión de modelos clásicos aplicados a los datos que forman parte de nuestra investigación, su evaluación y resultados. En el capítulo 3 se presentan los modelos AVs, sus bases teóricas y los experimentos realizados para construir la arquitectura final empleada en nuestra investigación. En el capítulo 4 se aborda la integración de los modelos AVs y AGs en una estructura combinada, su configuración y los resultados experimentales obtenidos de su aplicación. Finalmente, en el capítulo 5, se presentan las conclusiones de la investigación, así como posibles líneas futuras de trabajo.

Por último, en tiempos de grandes debates sobre los riesgos de los modelos generativos, esperamos poder brindar a nuestros lectores un ejemplo funcional de su estructura, iluminar ciertos principios y evidenciar sus límites. Si nuestro trabajo permite resaltar la sobriedad de sus mecanismos internos, habremos cumplido el anhelo de hacer menos opaca su belleza y más clara sus posibilidades.

Índice

Lista de figuras

Lista de tablas

Capítulo 1

El problema de la selección de características

En el presente capítulo abordamos el problema de la selección de características, su importancia y desafíos. En ese marco, repasaremos ciertas dificultades asociadas a los datos, como por ejemplo la alta dimensionalidad y el desbalance de clases, y veremos cómo ellas impactan en la selección de características. Luego describiremos brevemente distintos enfoques para la selección de características, como así también ventajas y desventajas de cada uno. Finalmente, vincularemos la selección de características con el aporte fundamental de nuestro trabajo asociado a la generación sintética de datos.

1.1. Estrés, ignorancia y selección de características

Un punto de partida difícil de controvertir en el mundo actual del aprendizaje automático es que la cantidad de información disponible para investigar ha crecido dramáticamente en los últimos veinte años (Li and Zhang 2023). Conforme la vida se digitaliza, y la información almacenada en los sistemas aumenta, la generación de conocimiento parece menos determinada por el viejo problema de la escasez de factores y más por su nueva situación de abundancia.

En este contexto, el desafío de trabajar con datos de alta dimensionalidad ha motivado la aparición de una serie de técnicas dirigidas a generar conocimiento y resultados precisos en escenarios complejos debido a la abundancia de información. Así, bajo el nombre de *selección de características* nació un area de estudio que busca resolver, precisamente, el problema de discernir *sistemáticamente* la información relevante de aquella que no lo es cuando se trabaja con muchas variables.

Según Bolón-Canedo y ot., el origen de este campo se remonta a los años '60, cuando Hughes, usando un modelo paramétrico, estudió la precisión de un clasificador bayesiano en función del número de características utilizado para predecir una variable objetivo (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). Por nuestra parte, entendemos que incluso se puede ir más lejos, a los años '30, cuando Fisher reprochaba el *estrés* con el que se buscaba conocimiento manipulando variables aisladas en lugar de mirar sus interacciones, proponiendo una *investigación experimental* más *abarcativa* y *eficiente* que permitiera obtener mayor conocimiento con la misma cantidad de observaciones (Fisher 1935).

Más allá de su origen, el poderoso impulso de considerar espacios de búsqueda cada vez más amplios y problemas cada vez más complejos, ha contribuido a que la selección de características se convierta en un campo de estudio activo e importante.

En efecto, hoy no es extraño encontrar investigaciones donde los objetos de estudio superen las decenas de miles de dimensiones, como sucede en la investigación biomédica. Allí, los microarrays de ADN son ejemplos representativos de datos de alta dimensionalidad, que constituyen fuentes vitales de información en problemas que involucran expresión génica (Almugren and Alshamlan 2019). En sentido similar, datos de video -presentes en multiples aplicaciones-, datos financieros, información vinculada a interacciones sociales, entre otros, son ejemplos del tipo de problema que justifican el uso de técnicas de selección de características (El-Hasnony et al. 2020).

Podemos definir a la *selección de características* como *el proceso de detectar las características relevantes y descartar aquellas irrelevantes o redundantes, con el objetivo de obtener un subconjunto que describa adecuadamente un problema dado con una degradación mínima del rendimiento* (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). Aquí rendimiento se refiere a la medida de evaluación empleada para juzgar los resultados del proceso de selección. Los distintos enfoques para selección de características adhieren a la premisa de que podemos separar la información relevante de aquella que no lo es para predecir una variable objetivo, y por ende mejorar el desempeño de los modelos de aprendizaje.

Nótese aquí que el *proceso* de selección de características es un proceso sistemático. Gran parte de su esfuerzo se centra en disponer de un método para discriminar, de manera precisa y controlada, la información relevante de la irrelevante o redundante. La precisión y el control destacan como dos características fundamentales debido a que la selección de características tiene lugar en un escenarios donde se ignora cual es el aporte de cada variable a la resolución del problema. Por eso, resulta necesario contar con un criterio de evaluación claramente definido que permitan examinar de forma objetiva y cuantificable el valor real de cada variable y sus interacciones.

Bajo esa perspectiva, la selección de características busca determinar un subconjunto de atributos que satisfaga uno de los siguientes criterios(Vignolo and Gerard 2017):

1. El subconjunto con un tamaño especificado que maximice la precisión de la predicción.
2. El subconjunto de menor tamaño que satisfaga un requisito de precisión mínima.
3. El subconjunto que logre el mejor compromiso entre dimensionalidad y precisión.

El criterio a elegir dependerá de los objetivos del estudio y de las características del problema. Mas allá de eso, es fácil advertir que la selección de características supone como ventaja la reducción del espacio de búsqueda, y en cierto sentido es un remedio a la alta dimensionalidad.

En efecto, en el contexto del aprendizaje automático es común enfrentar problemas representados por grandes conjuntos de variables, vicisitud que se asocia con la *maldición de la dimensionalidad* (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). En general, este fenómeno se presenta por la alta demanda computacional y costos asociados con la optimización de dichos espacios, volviendo a ciertos problemas intratables desde el punto de vista práctico. Para abordarlo, una alternativa posible es la reducción de dimensionalidad, que consiste en encontrar o construir matrices con menor número de columnas que las originales pero la misma carga de información relevante. Esas matrices de menor dimensión, puede usarse de manera más eficiente para modelar el problema objetivo, facilitando su interpretación y comunicación. El proceso de encontrar estas matrices se llama *reducción de dimensionalidad*, y uno de los métodos para lograrlo es la selección de características.

Dicho lo anterior, es preciso reconocer que la abundancia de información trae consigo una serie de desafíos que impactan en la selección de características. En el siguiente apartado describiremos aquellos más relevantes para nuestro estudio.

1.2. Las multiples cara de los datos problemáticos

1.2.1. Alta dimensionalidad y escasez muestral

En primer lugar, un problema frecuente en el aprendizaje automático se presenta cuando disponemos de pocas muestras en un espacio vectorial de alta dimensionalidad. Es decir, cuando el número de muestras (m) es menor que el número de características (n), siendo n particularmente grande. Como vimos, esta situación es común en el campo del análisis genético, el procesamiento de información médica, procesamiento de video, entre otros. Como veremos mas adelante, tres de los datasets elegidos para nuestro estudio -vinculados al ámbito biomédico- se encuentran en esta situación.

La escasez de muestras dificulta el tratamiento de la información, ya que la cantidad de datos disponibles para entrenar un modelo es insuficiente para capturar la variabilidad inherente a los datos. En tales circunstancias, el proceso de aprendizaje tiende a sufrir severas limitaciones, bien sea sobreajustándose a las observaciones disponibles (llevando a modelos poco generalizables), o bien resultando incapaz de capturar la estructura subyacente de los datos (lo que puede llevar a modelos poco precisos).

Por ejemplo, en el campo de la clasificación de perfiles de expresión génica, el problema se considera difícil de resolver debido a la complejidad que supone identificar los genes que contribuyen a la aparición de una enfermedad (Almugren and Alshamlan 2019). Dada la gran cantidad de genes presentes en estos supuestos (muchos de ellos irrelevantes), entrenar un modelo con todos ellos puede conducir a resultados erróneos. Este desafío, comúnmente se ve acentuado por la baja cantidad de observaciones disponibles y por el hecho de que los genes se encuentran altamente correlacionados. Dichas características dificultan el adecuado funcionamiento de los métodos clásicos de aprendizaje automático, cuyo rendimiento se asocia a la cantidad de muestras disponibles.

Ante estos problemas, la selección de características tiene severas limitaciones, ya que la alta dimensionalidad y bajo número de muestras dificultan la distinción entre información relevante y irrelevante.

1.2.2. Desbalance de clases

El desbalance de clases constituye otro problema importante y frecuente en los desafíos actuales que se presentan para el aprendizaje automático. El mismo sucede cuando la distribución de las clases en el conjunto de datos es altamente desigual: típicamente una clase mayoritaria supera ampliamente las observaciones de una o más clases minoritarias. Este problema puede darse en combinación con alta dimensionalidad y escacez muestral, agravando la dificultad de selección de características.

El desbalance de clases es un problema relevante en aplicaciones donde la o las clases minoritarias son precisamente las de mayor interés, como por ejemplo: el diagnóstico de enfermedades raras, la detección de fraudes bancarios, la identificación de intrusiones en redes y la predicción de fallos en equipos técnicos. En supuestos como estos, los clasificadores que no contemplen un tratamiento explícito del problema tienden

a sesgarse hacia la clase mayoritaria, pudiendo alcanzar una alta precisión global mientras fallan en la detección de los casos más importantes pero menos frecuentes.

Para atacar este problema, se han propuesto diversas soluciones, que pueden categorizarse en tres grupos:

1. **Muestreo de datos**: Este enfoque modifica las muestras de entrenamiento para producir una distribución de clases más balanceada. Las técnicas tradicionales incluyen: submuestreo (*undersampling*) donde se cre un subconjunto del conjunto original eliminando instancias; sobremuestreo (*oversampling*) donde se gnera un superconjunto replicando instancias existentes o creando nuevas, y finalmente, métodos híbridos que combinan ambas técnicas de muestreo.
2. **Modificación algorítmica**: Este enfoque adapta los métodos de aprendizaje para que sean más sensibles a los problemas de desbalance. Por ejemplo, el uso de la distancia de Hellinger como criterio de división en árboles de decisión, que ha demostrado ser insensible al sesgo, capturando la divergencia en las distribuciones sin ser dominada por las probabilidades previas de clase.
3. **Aprendizaje sensible al costo**: Esta solución puede incorporar elementos a nivel de datos, a nivel de algoritmos, o ambos a la vez, asignando costos más altos a la clasificación errónea de ejemplos de la clase positiva (minoritaria). Así, en el diagnóstico médico, resulta más importante reconocer la presencia de una enfermedad rara que su ausencia, por ello el costo de un falso negativo es mayor que el de un falso positivo.

El desbalance de clases también supone un desafío para la selección de características. Ello es así, en la medida que los métodos de selección estarán sesgados a favorecer aquellas que maximicen la separación de clases en su conjunto, sin ponderar sus diferencias para capturar información relevante para todas las clases presentes. Esto ocurre porque, en escenarios con distribución fuertemente desequilibrada, las técnicas de evaluación tienden a priorizar la identificación de la clase mayoritaria, reduciendo la relevancia de las características que serían valiosas para discriminar a las minoritarias. Como resultado, se descarta información relevante para la predicción de los casos poco frecuentes, y se seleccionan características que no son representativas para todas las clases.

Siguiendo a Bolón-Canedo y ot., junto a los problemas mencionados sobre alta dimensionalidad y desbalance de clases, la complejidad y el ruido de los datos también representan desafíos importantes en el tratamiento de la información que debemos considerar en el presente estudio.

1.2.3. Complejidad

Respecto a la complejidad, se refiere a la dificultad de identificar las fronteras de decisión entre clases, que pueden manifestarse en tres aspectos fundamentales (todos ellos pueden coexistir en un mismo problema):

1. **Ambigüedad de clases**: Surge cuando los casos no pueden distinguirse utilizando las características disponibles, ya sea porque los conceptos de clase están mal definidos y por lo tanto son intrínsecamente indistinguibles, o porque las características seleccionadas son insuficientes para discriminar las clases.
2. **Complejidad de fronteras**: Se refiere a situaciones donde los límites entre clases requieren una descripción extensa de la frontera de decisión, llegando a

demandar, en el caso extremos de complejidad, la enumeración exhaustiva de todos los puntos con sus etiquetas de clase. Este aspecto de dificultad se debe a la naturaleza del problema y no a la muestra o selección de características, indicando que una completa separación entre clases es un problema difícil de resolver.

3. **Dispersión de muestras**: La combinación de pocas muestras y alta dimensionalidad genera una dispersión que dificulta la generación de fronteras de decisión.

Anque los casos 1 y 2 son problemas ante los cuales la selección de características no puede ofrecer una solución efectiva, el caso 3, asociado con alta dimensionalidad y escasez muestral, podría ser mitigado por estrategias de aumentación de datos.

1.2.4. Ruido

Los datos del mundo real siempre están expuestos a imperfecciones, ruido, proveniente de entornos dinámicos donde las dimensiones de interés de un fenómeno coexisten en espacios de interacciones permanentes. Así, aún considerando un escenario artificial, completamente libre de interferencia, la imperfección puede provenir de diversas fuentes como dispositivos de medición defectuosos o limitados, errores de transcripción o irregularidades en la transmisión de información.

Ante tales circunstancias, existen cuatro enfoques principales para abordar estas imperfecciones en los conjuntos de datos:

1. **Conservación del ruido**: En este enfoque, se mantiene el conjunto de datos tal como está, con sus instancias ruidosas. Los algoritmos que utilizan los datos se diseñan para ser robustos, es decir, capaces de tolerar cierta cantidad de ruido. Una estrategia común es desarrollar algoritmos que eviten el sobreajuste al modelo (como sucede en el caso de los árboles de decisión) mediante técnicas de poda.

2. **Limpieza de datos**: Este método implica descartar las instancias que se consideran ruidosas según ciertos criterios de evaluación. El clasificador se construye utilizando únicamente las instancias retenidas en un conjunto de datos más pequeño pero más limpio. Sin embargo, este enfoque presenta dos debilidades significativas:

   - Al eliminar instancias completas, se puede descartar información potencialmente útil, como valores de características no corrompidos.
   - Cuando existe una gran cantidad de ruido, la información restante en el conjunto de datos limpio puede resultar insuficiente para construir un clasificador eficaz.

3. **Transformación de datos**: Este enfoque busca corregir las instancias ruidosas en lugar de eliminarlas. Las instancias identificadas como ruidosas se reparan reemplazando los valores corrompidos por otros más apropiados, y luego se reintroducen en el conjunto de datos.

4. **Reducción de datos**: Esta estrategia implica reducir la cantidad de datos mediante la agregación de valores o la eliminación y agrupación de atributos redundantes. La reducción de dimensionalidad -y consecuentemente la selección de características- es una de las técnicas más populares para eliminar características ruidosas (es decir, irrelevantes) y redundantes.

Vistos los problemas que enfrentamos con los datos, pasemos a describir, brevemente, los distintos enfoques que se han propuesto para sortearlos empleando estrategias de selección de características.

1.3. Distintos enfoques para la selección de características

Podemos agrupar los métodos de selección de características en tres grandes grupos:

1. **Filtros**: Se basan en las características generales de los datos de entrenamiento y realizan la selección de características como un paso de preprocesamiento independiente del algoritmo de aprendizaje. El análisis de relevancia de una característica se realiza considerando sus propiedades intrínsecas, sin determinar sus relaciones posibles con otras. El resultado de dicho análisis es una lista de características ordenadas por relevancia (ranking), donde el subconjunto final de características se selecciona según ese orden. Este enfoque es ventajoso por su bajo costo computacional, rapidez y escalabilidad, pero deja sin resolver el problema apuntado por Fisher sobre la interacción entre variables. Los metodos de filtro se pueden sub-clasificar en univariados y multivariados (Solorio-Fernández, Carrasco-Ochoa, and Martínez-Trinidad 2020). Los primeros analizan cada característica de manera independiente con el fin de obtener una lista ordenada. Este tipo de métodos puede identificar y eliminar eficazmente características irrelevantes, pero no son capaces de eliminar las redundantes, ya que no consideran posibles dependencias entre las características. Ejemplos de estos métodos son: la evaluación utilizando la distribución *chi-cuadrado* , la *ganancia de información*, entre muchos otros (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). Los métodos filtro multivariados evalúan la relevancia de las características de forma conjunta en lugar de hacerlo individualmente. Por ejemplo, el método de selección hacia adelante (forward selection) y hacia atrás (backward selection) son métodos de filtro multivariados que sucesivamente agregan y eliminan características para obtener un subconjunto óptimo. Los métodos multivariados pueden manejar características redundantes e irrelevantes; por lo tanto, en muchos casos, la precisión alcanzada por los modelos empleando subconjuntos seleccionados con estos métodos puede ser mayor. Otros métodos filtro multivariados puden ser el *método basado en el análisis de correlación* y *algoritmo ReliefF*, entre otros (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015).

2. **Wrappers o métodos envolventes**: Involucran un algoritmo de aprendizaje como caja negra y consisten en usar su resultado para evaluar la utilidad relativa de subconjuntos de características. Aquí, el algoritmo de selección utiliza el método de aprendizaje como una subrutina, para encontrar los subconjuntos de características más relevantes. Esta estrategia es capaz de reconocer variables relevantes y capturar dependencias entre ellas, pero a un costo computacional mayor que los otros métodos. En ese sentido enfrenta desafíos importante cuando el conjunto de datos es de alta dimensionalidad, ya que evaluar 2^n subconjuntos de características puede resultar en un problema intratable. Por esa razón, se recurre comunmente a heurísticas de búsqueda capaces de encontrar soluciones adecuadas sin explorar todo el espacio del problema (R. Zhang et al. 2019).

3. **Embedded o métodos embebidos**: Realizan la selección de características durante el proceso de entrenamiento de un modelo de aprendizaje, y suelen ser específicos para determinados algoritmos (e.g. árboles de decisión y métodos derivados de ellos, eliminación recursiva de características mediante SVM, entre otros). A diferencia de los métodos de filtro y wrappers, los métodos embebidos no separan la selección de características del proceso de entrenamiento, sino que la realizan de manera simultánea. En este caso, optimizan una función de pérdida regularizada con respecto a dos conjuntos de parámetros: los parámetros del algoritmo de aprendizaje y los parámetros que indican las características seleccionadas.(Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). Este enfoque es capaz de capturar dependencias a un costo computacional menor que los wrappers.

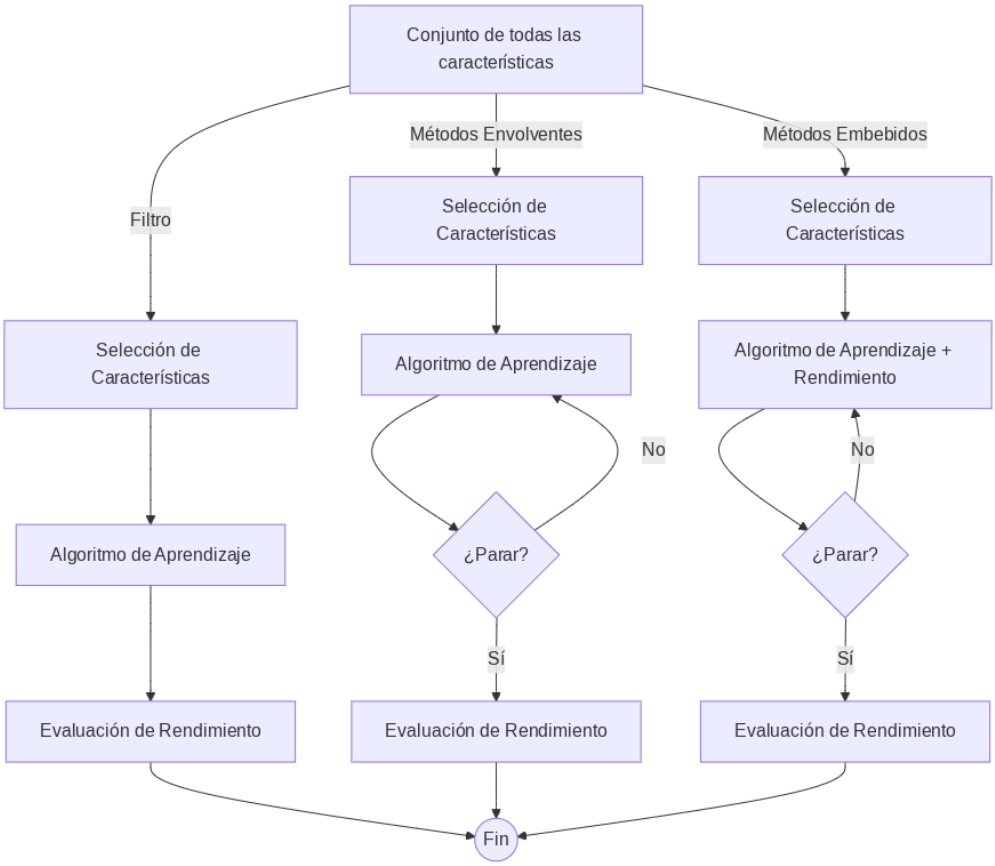

FIGURE 1.1: Diagrama de enfoques para la selección de características

Bolón-Canedo y ot. nos invitan a pensar que no existe un método que sea *superior a los otros* de manera general, sino que cada uno tiene sus ventajas y desventajas, y se ajustan distinto a diferentes tipos de contexto. De los tres enfoques, los métodos de filtro son los únicos que son independientes del algoritmo de aprendizaje, lo que les permite ser rápidos y eficientes computacionalmente. Sin embargo, los métodos de filtro no consideran la correlación entre características, lo que puede llevar a la selección de características irrelevantes o redundantes. Los métodos de wrappers y embedded, por su parte, consideran la correlación entre características y etiquetas de

clase, lo que les permite identificar patrones relevantes correctamente durante la fase de aprendizaje. Sin embargo, los métodos de wrappers y embedded son más complejos computacionalmente y requieren evaluación iterativa del subconjunto seleccionado de características.

1.4. Selección de características y generación sintética de datos

En el ámbito de la selección de características, los algoritmos genéticos (AGs) se han propuesto como un mecanismo de exploración eficiente para la búsqueda de soluciones cercanas al óptimo en espacios de gran dimensionalidad (Vignolo and Gerard 2017). Inspirados en los principios de la evolución natural —que presentaremos en el capítulo 4—, estos algoritmos son capaces de iterar sobre múltiples generaciones de individuos (posibles subconjuntos de características) produciendo soluciones cada vez más relevantes.

Sin embargo, su eficacia depende en gran medida de la disponibilidad de datos suficientes para evaluar el valor de cada subconjunto. En efecto, la falta de datos para estimar de forma confiable el desempeño de características seleccionadas afecta la función objetivo, agregando incertidumbre al proceso de selección. Cabe destacar que esta función objetivo está representada por un modelo de aprendizaje automático con el que se evalúa el desempeño de los subconjuntos (por ejemplo, un clasificador basado en árboles de decisión, máquinas de soporte vectorial, o un perceptrón multicapa). Por eso, respecto de este componente en particular, el algoritmo genético se encuentra en la misma situación que cualquier otro método de optimización donde la escacez de datos afecta la calidad del modelo.

Para abordar esta limitación, se han desarrollado estrategias que mitigan el impacto de la escasez de información. Entre ellas destaca la aumentación de datos mediante Autoencodificadores Variacionales (AVs). Estos modelos generativos, basados en la estimación y reconstrucción de distribuciones latentes, son capaces de reconocer patrones y generar muestras sintéticas que retienen características estadísticamente relevantes del conjunto real de datos. Así, los AVs se presentan como una herramienta potencialmente transformadora para los procesos de selección de características basada en AGs, al permitir la creación de muestras adicionales que permitan la correcta evaluación de los subconjuntos de características. Al beneficiar el entrenamiento de los modelos de aprendizaje, la generación sintética de datos puede mejorar el desempeño de los métodos evolutivos en la búsqueda de características óptimas.

Asimismo, en escenarios de desbalance de clases —especialmente críticos en problemas multiclase con distribuciones asimétricas—, la generación sintética de datos puede ofrecer la posibilidad de reequilibrar las proporciones de cada categoría. En efecto, al proporcionar suficientes muestras representativas de las clases minoritarias, la generación sintética de datos puede mejorar la discriminación entre categorías poco representadas. Con esta estrategia, se reduciría el sesgo introducido por la disparidad de muestras y, por ende, se mejoraría el rendimiento del modelo en su conjunto.

Por último, es importante destacar que la generación sintética de datos mediante AVs también puede contribuir a mitigar el efecto del ruido en los modelos de aprendizaje. Durante el proceso de codificación-decodificación implementado por estos modelos, el proceso de reconstrucción identifica y proyecta las características esenciales de los datos, favoreciendo la reducción de información irrelevante. Este mecanismo, analizado

en mayor detalle en el capítulo 3, abonaría la importancia de los AVs en la creación de muestras sintéticas no solo para reducir el impacto negativo del ruido sobre el proceso de entrenamiento, sino también para mejorar el desempeño de los métodos evolutivos en la búsqueda de características óptimas.

Todas estas razones avalan la importancia de la generación sintética de datos en la selección de características con AGs. En la presente tesis se propone una estrategia integral que vincula la selección de características con la generación sintética de datos, ofreciendo soluciones potenciales a problemas donde la dimensionalidad, el desbalance de clases, la escasez de muestras y el ruido representan barreras habituales en el campo del aprendizaje automático.

Siguiendo esta idea, la tesis se organiza de la siguiente manera: seguidamente, en el capítulo 2, se realiza un análisis preliminar de los datos con los que trabajaremos, y se describen los modelos de aprendizaje automático evaluados para servir de función objetivo en el Algoritmo Genético que desarrollaremos. Además, se presentan las métricas de evaluación aplicadas a lo largo del estudio y se analizan potenciales dificultades. Luego, en el capítulo 3, se aborda la teoría de los Autocodificadores Variacionales (AVs) y la motivación para usarlos en la generación de datos sintéticos. Se describen los experimentos de aumentación de datos y se presentan los resultados obtenidos, enfatizando la forma en que estos modelos capturan la estructura subyacente de la distribución original y potencian la calidad de los datos sintéticos. En el capítulo 4 presentamos nuestro Algoritmo Genético y describimos la integración de los AVs como etapa de preprocesamiento a los fines de enriquecer la selección de características. Se detallan los experimentos realizados tanto con datos originales como con datos aumentados, demostrando el impacto que la generación sintética ejerce en el rendimiento de los métodos de selección de características. Por último, en el capítulo 5 se exponen las conclusiones de la investigación, destacando la relevancia de la generación sintética de datos para la selección de características en escenarios complejos, y se ofrecen perspectivas de trabajo futuro donde las técnicas propuestas podrían ampliarse o refinarse.

Capítulo 2

Marco específico de trabajo

En este capítulo revisaremos el desempeño de algoritmos clásicos en la solución de los conjuntos de datos elegidos para nuestra investigación. Describiremos brevemente la composición de los mismos, los algoritmos seleccionados para su tratamiento y los resultados obtenidos para cada uno. Luego, analizando y comparando dichos resultados, elegiremos un modelo para emplear como función de aptitud en nuestro algoritmo genético. Finalmente, haremos un breve repaso de las aplicaciones de Autocodificadores Variacionales (AVs) al problema de la aumentación de datos, como antesala al capítulo siguiente.

El propósito de esta etapa del trabajo es doble. Por un lado, identificar los modelos más apropiados para servir de *función de aptitud* en la implementación de nuestro algoritmo genético. Esto permitirá construir una función adecuada para evaluar cada solución. Al mismo tiempo, disponer de métricas acerca del desempeño que logran distintas estrategias de clasificación, a partir de las cuales comparar el resultado de nuestras propias soluciones. Por último, revisar ejemplos de aplicación de los AVs en escenarios de aumentación de datos para preparar nuestro trabajo del capítulo siguiente.

2.1. Datos elegidos en nuestro estudio

El conjunto de datos elegidos en este trabajo incluye:

1. *Madelon*: conjunto artificial de datos con 500 características, donde el objetivo es resolver un problema XOR multidimensional con 5 características relevantes y 15 características corresponden a combinaciones lineales de aquellas (i.e. 15 características redundantes). Las otras 480 características fueron generadas aleatoriamente (no tienen poder predictivo). Madelon es un problema de clasificación de dos clases con variables de entrada binarias dispersas. Las dos clases están equilibradas, y los datos se dividen en conjuntos de entrenamiento y prueba. Fue creado para el desafío de Selección de Características NIPS_2003, y está disponible en el Repositorio UCI. Los datos están divididos en un conjunto de entrenamiento y un conjunto de testeo.

2. *Gisette:* es un dataset creado para trabajar el problema de reconocimiento de dígitos escritos a mano (Isabelle Guyon 2004). Este conjunto de datos forma parte de los cinco conjuntos utilizados en el desafío de selección de características NIPS 2003. Tiene 13500 observaciones y 5000 atributos. El desafío radica en diferenciar los dígitos ‘4’ y ‘9’, que suelen ser fácilmente confundibles entre sí. Los dígitos han sido normalizados en tamaño y centrados en una imagen fija

de 28x28 píxeles. Además, se crearon nuevas características como combinación de las existentes para construir un espacio de mayor dimensión. También se añadieron características distractoras denominadas "sondas", que no tienen poder predictivo. El orden de las características y patrones fue aleatorizado. Los datos están divididos en un conjunto de entrenamiento y un conjunto de testeo.

3. *Leukemia*: El análisis de datos de expresión génica obtenidos de microarreglos de ADN se estudia en Golub (1999) para la clasificación de tipos de cáncer. Construyeron un conjunto de datos con 7129 mediciones de expresión génica en las clases ALL (leucemia linfocítica aguda) y AML (leucemia mielogénica aguda). El problema es distinguir entre estas dos variantes de leucemia (ALL y AML). Los datos se dividen originalmente en dos subconjuntos: un conjunto de entrenamiento y un conjunto de testeo.

4. *GCM*: El conjunto de datos GCM fue compilado en Ramaswamy (2001) y contiene los perfiles de expresión de 198 muestras de tumores que representan 14 clases comunes de cáncer humano. Aquí el enfoque estuvo en 190 muestras de tumores después de excluir 8 muestras de metástasis. Finalmente, cada matriz se estandarizó a una media de 0 y una varianza de 1. El conjunto de datos consta de un total de 190 instancias, con 16063 atributos (biomarcadores) cada una, y distribuidos en 14 clases desequilibradas. Los datos están divididos en un conjunto de entrenamiento y un conjunto de testeo.

Para la etapa de experimentos hemos incluido un quinto dataset para validar resultados. Este quinto dataset posee las siguientes características:

5. *All Leukemia*: Es un dataset empleado en el estudio de pacientes pediátricos con leucemia linfoblástica aguda (LLA). Incluye 327 muestras con información de 12600 variables (genes). Está compuesto por 14 clases desequilibradas. Fue compilado por Yeoh et al., y descripto en *Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling*, Cancer Cell, Volume 1, Issue 2, 133 - 143. link

2.2. El desempeño de algoritmos clásicos

Para disponer de métricas de base para la comparación de nuestra solución y, al mismo tiempo, evaluar el grado de dificultad que conlleva la resolución de las tareas asociadas a los datos incluidos en nuestro estudio, hemos seleccionado una serie de modelos ampliamente usados en el campo del aprendizaje automático para tomar su desempeño como indicador. A fin de estandarizar la implementación de estos algoritmos, hemos empleado la librería `scikit-learn` que provee abstracciones convenientes para nuestro entorno de experimentación. Los modelos elegidos son:

Modelos lineales

Los modelos lineales son un conjunto de algoritmos que predicen la salida en función de una combinación lineal de características de entrada (ver EEL). Son particularmente útiles cuando se espera que haya una relación lineal entre variables.

- `LDA`: Análisis Discriminante Lineal, empleado para dimensiones reducidas y asumiendo distribuciones gaussianas.

- `QDA`: Análisis Discriminante Cuadrático, similar a LDA pero con covarianzas distintas por clase.
- `Ridge`: Regresión de Cresta, empleado para tratar con multicolinealidad mediante regularización L2.
- `SGD`: Descenso de Gradiente Estocástico, estrategia central del aprendizaje automático, empleado para optimizar modelos lineales.

Modelos basados en árboles

Los modelos basados en árboles implican la segmentación del espacio de características en regiones simples dentro de las cuales las predicciones son más o menos uniformes (Ver CITA). Son potentes y flexibles, capaces de capturar relaciones complejas en los datos.

- `DTC`: Árbol de Decisión Clásico, modelo intuitivo que divide el espacio de características.
- `AdaBoost`: Estrategia que entrena modelos débiles secuencialmente, enfocándose en que cada nuevo modelo procese las instancias u observaciones previamente difíciles de clasificar.
- `Bagging`: Estrategia que combina predicciones de múltiples modelos para reducir la varianza.
- `Extra Trees Ensemble`: Estrategia que construye múltiples árboles con particiones aleatorias de características y umbrales.
- `Gradient Boosting`: Estrategia que mejora modelos de forma secuencial minimizando el error residual.
- `Random Forest`: Estrategia basada en conjunto de árboles de decisión, cada uno entrenado con subconjuntos aleatorios de datos.
- `ETC`: Árboles Extremadamente Aleatorizados, variante de Random Forest con más aleatoriedad.

Modelos de Naive Bayes

Los modelos de Naive Bayes son clasificadores probabilísticos basados en el teorema de Bayes que presupone independencia entre las características (ver CITA). Son modelos eficientes y de rápida ejecución.

- `BNB`: Naive Bayes Bernoulli, se emplea para características de variables binarias.
- `GNB`: Naive Bayes Gaussiano, es adecuado cuando los datos poseen distribución normal.

Modelos de vecinos más cercanos

El modelo de k vecinos más cercanos (KNN) es un método de clasificación no paramétrico que permite clasificar una muestra basándose en la proporción de clases de las muestras más cercanas en el espacio de características. Es simple y efectivo, particularmente para datos donde las relaciones entre características son complejas o desconocidas.

- `KNN`: K-Vecinos más Cercanos, clasifica asignando la clases mayoritaria en la vecindad considerada.

Modelos de redes neuronales

El Perceptrón Multicapa (MLP, por sus siglas en inglés) es un tipo de red neuronal que consiste en múltiples capas de neuronas con funciones de activación no lineales. Puede modelar relaciones complejas y no lineales entre entradas y salidas, y es altamente adaptable a la estructura de los datos.

Modelos de Máquinas de Soporte Vectorial
Las Máquinas de Soporte Vectorial (SVM) son un conjunto de algoritmos supervisados que buscan la mejor frontera de decisión lineal que puede separar diferentes clases en el espacio de características. Ofrecen alta precisión y son muy efectivos en espacios de alta dimensión y en casos donde el número de dimensiones supera al número de muestras.

- `LSVC`: Máquinas de Soporte vectorial Lineal, se emplea en espacios de alta dimensión.
- `NuSVC`: SVC con parámetro Nu, que controla el número de vectores de soporte.

Finalmente, es preciso destacar que para GCM, que contiene 14 clases en la variable objetivo, hemos excluido modelos no compatibles o ineficientes para problemas de clasificación multiclases.

2.3. Configuración de los Modelos

Para evaluar los modelos clásicos hemos decidido su configuración a partir de la búsqueda de la mejor combinación de parámetros. A tal fin, hemos seleccionado aquellos parámetros más importantes en cada modelo y establecimos una búsqueda en grilla de sus respectivos valores. Hemos seleccionado para parámetros numéricos un mínimo de 3 valores y máximo de 20, y para no numéricos hemos decidido la configuración estándar según cada modelo. El espacio de búsqueda resultante para cada modelo puede verse en el siguiente link.

[Incluir Tabla]

2.4. Resultados Obtenidos

En la siguiente tabla resumimos los resultados obtenidos del entrenamiento de los modelos en los dataset estudiados.

[-Tal vez se podrían trazar líneas horizontales para separar los "tipos" de modelos? Por ejemplo, lineales de árboles, de svm, etc. Además, tal vez dejaría sólo los resultados de test, que son los que uno busca analizar generalmente. Otra opción podría ser poner el resultado de test y, entre paréntesis el de train, pero me parece muy rebuscado. -Sugiero resaltar en negrita el mejor resultado por columna, en lugar de fila, ya que queremos analizar cuán complejo es resolver la tarea para cada dataset. -Qué métrica se está analizando? -Los resultados son para test? Qué esquema de particionado se empleó? El propuesto previamente?]

Models	Leukemia Train	Leukemia Test	Madelon Train	Madelon Test	Gisette Train	Gisette Test	GCM Train	GCM Test
LDA	0.93	0.85	0.82	0.6	1.0	0.96	-	-
QDA	1.0	0.5	1.0	0.66	1.0	0.7	-	-
Ridge	1.0	0.99	0.82	0.6	1.0	0.97	-	-
SGD	1.0	0.98	0.63	0.64	1.0	0.99	1.0	0.71
AdaBoost	1.0	0.91	0.89	0.84	1.0	0.99	-	-
Bagging	1.0	1.0	0.97	0.91	-	-	-	-
DTC	1.0	0.72	0.77	0.64	0.95	0.92	0.95	0.53
ETC	1.0	0.54	0.62	0.57	0.95	0.94	0.98	0.48
Ext.Trees.Ens.	1.0	1.0	1.0	0.71	0.99	0.99	1.0	0.57

Models	Leukemia Train	Leukemia Test	Madelon Train	Madelon Test	Gisette Train	Gisette Test	GCM Train	GCM Test
Gradient Boost.	1.0	0.99	1.0	0.82	1.0	1.0	1.0	0.58
Random Forest	1.0	1.0	1.0	0.78	0.99	0.99	1.0	0.62
BNB	1.0	0.89	0.73	0.63	0.95	0.94	-	-
GNB	1.0	0.91	0.81	0.65	0.91	0.85	-	-
KNN	0.86	0.86	0.74	0.65	0.99	0.99	-	-
LSVC	1.0	0.99	0.78	0.62	1.0	0.99	1.0	0.62
NuSVC	1.0	1.0	1.0	0.61	1.0	0.99	0.99	0.58
SVC	1.0	1.0	1.0	0.61	1.0	0.99	1.0	0.58
MLP	1.0	0.96	1.0	0.58	1.0	0.99	1.0	0.68

Estos valores dan forma a la siguiente representación:

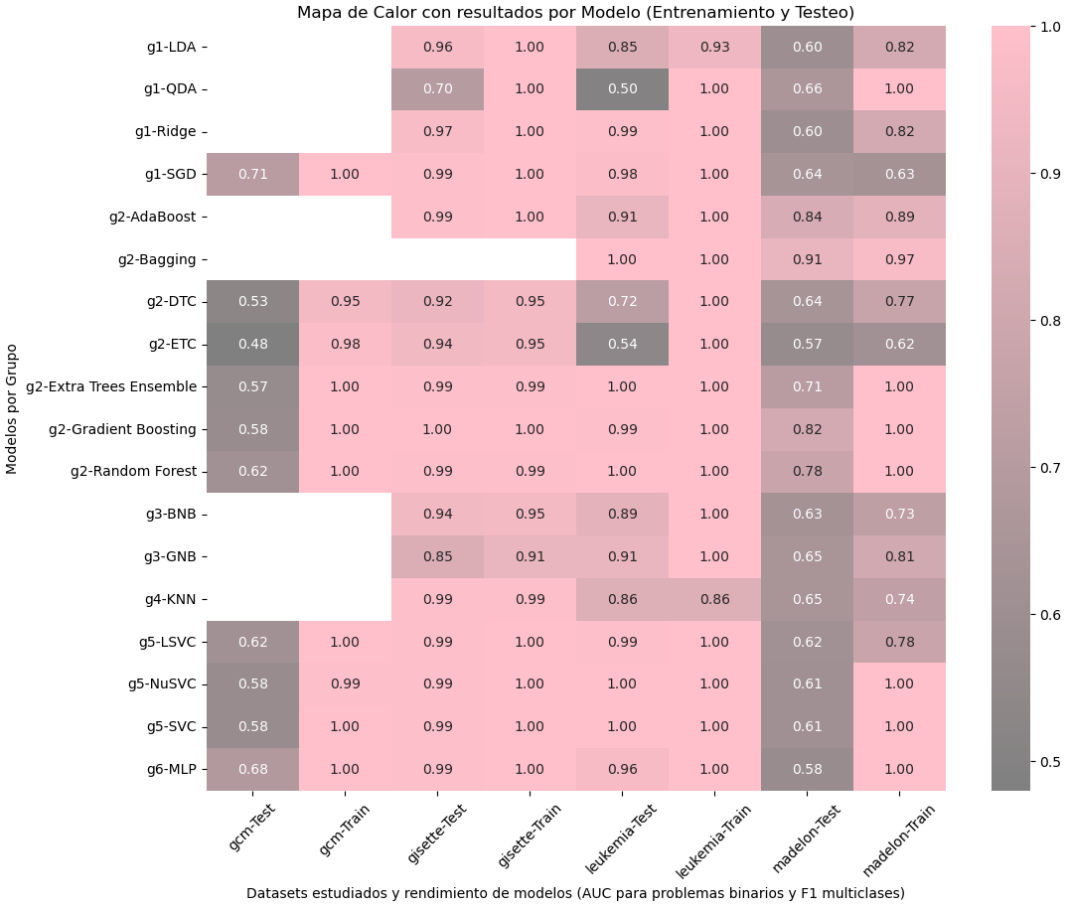

FIGURE 2.1: algoritmosclasicos

[DESARROLLAR]Podría resaltarse en estos resultados que hay problemas en los datasets desbalanceados (que es un poco la hipótesis de trabajo de la tesis: los VAEs pueden ayudar a balancear de forma efectiva los conjuntos de datos para que las tareas posteriores se realizan de forma correcta)

2.5. El uso de Autocodificadores Variacionales como técnica de aumentación

La aplicación de AVs como técnica de aumentación de datos en el contexto de distintos problemas de aprendizaje automático es extendida fuera del campo de la selección de características. Se aplica al tratamiento de imágenes (Fajardo et al. 2021; Ai et al. 2023; Khmaissia and Frigui 2023; Kwarciak and Wodzinski 2023), texto (Y. Zhang et al. 2019) , habla (Blaauw and Bonada 2016; Latif et al. 2020) y música (Roberts et al. 2019), y distintos formatos de datos: tabulares (Leelarathna et al. 2023), longitudinales (Ramchandran et al. 2022) y grafos (Liu et al. 2018). En lo que sigue repasaremos las experiencias más afines a nuestro enfoque sobre el impacto de la aumentación en el aprendizaje y la selección de características.

En Fajardo et al. (2021) se investiga si los AVs y las redes generativas antagónicas (GAN) pueden aumentar datos desbalanceados vía sobremuestreo de las clases minoritarias, y mejorar así el rendimiento de un clasificador. Para ello se crean versiones desbalanceadas de reconocidos datos multiclases tales como MNIST (Lecun 1998) y Fashion MNIST (Xiao, Rasul, and Vollgraf 2017), a los cuales, posteriormente, se los re-balancea agregándoles muestras sintéticas generadas por un AV condicionado por clase (AV Condicional). Para la tarea de clasificación se emplea un Perceptrón Multicapa (MLP), y se evalúa su desempeño promediando métricas de precisión, exhaustividad [controlar-concepto] y F1 score sobre distintos experimentos. La evaluación incluye la comparación de resultados del clasificador con datos aumentados por sobre-muestreo aleatorio, mediante SMOTE (Blagus and Lusa 2013), GAN y AVs. El resultado muestra a los AVs -en su versión condicional- como el mejor modelo generativo para resolver el problema de datos desbalanceados mediante sobre-muestreo de las clases minoritarias.

Ai et al. (2023) vuelve sobre los problemas planteados en Fajardo et al. (2021), proponiendo una nueva metodología que superaría sus resultados. La propuesta en esta oportunidad plantea la aumentación de datos de la clase minoritaria condicionada a las características de la distribución que tienen los datos de la clase mayoritaria. El método se llama AV-Guiado-por-la-Mayoría (*Majority-Guided VAE* o MGVAE) y procura incorporar en la generación no solo información intra-clase sino también inter-clase, con el fin de propagar la diversidad y riqueza de la mayoría en la minoría, y mitigar así riesgos de sobre-ajuste en los modelos. Este modelo se pre-entrena utilizando muestras de la clase mayoritaria, y luego se ajusta con datos de la clase minoritaria para retener el aprendizaje de la etapa previa (Kirkpatrick et al. 2017). Para evaluar la eficacia de MGVAE, se realizaron experimentos en varios conjuntos de datos de imágenes y tabulares, utilizando diversas métricas de evaluación como Precisión Balanceada (B-ACC), Precisión Específica Promedio por Clase (ACSA) y Media Geométrica (GM). Los resultados muestran que MGVAE supera a otros métodos de sobre-muestreo en tareas de clasificación.

Un problema diferente es tratado en Khmaissia and Frigui (2023) donde se emplean AVs para aumentar datos en una tarea de clasificación con enfoque semi-supervisado. Aquí el desafío no pasa por el desbalance entre clases, sino en la búsqueda de mejorar el clasificador en regiones del espacio de características con bajo desempeño (ratios de error altos). Para eso, se mapea el espacio de características entrenando un modelo de WideResNet (Zagoruyko and Komodakis 2017) y luego se seleccionan las muestras mal clasificadas o con bajo nivel de confianza en la clasificación. Estas muestras se utilizan para entrenar un AV y generar datos sintéticos. Finalmente, las imágenes

sintéticas se usan junto con las imágenes originales etiquetadas para entrenar un nuevo modelo de manera semi-supervisada. Se evalúan los resultados sobre STL10 y CIFAR-100 obteniendo mejoras en la clasificación de imágenes en comparación con los enfoques supervisados.

Finalmente, antecedente interesante es el presentado por Martins, Rocha, and Pereira (2022), pues pese a no estar directamente vinculado a la aumentación de datos, incluye la generación sintética de muestras mediante AV y la selección de características por AG. En efecto, el artículo propone la generación de individuos y optimización de características orientados al diseño de proteínas (específicamente variantes de Luciferasa bacteriana *luxA*). Partiendo de muestras de ADN de proteínas obtenidas de un subconjunto de datos de la base InterPro (identificados bajo el código "IPR011251") se generan conjuntos de individuos combinando datos originales, datos muestreados de la capa latente -*encoder*- del AV (configurado como MSA-AV para procesar sequencias alineadas de ADN) y datos optimizados por aplicación del AG. En el caso del algoritmo genético se emplean dos enfoques de optimización: de objetivo único y multiobjetivos, con funciones asociadas a la búsqueda de propiedades deseables en las muestras de ADN (solubilidad, síntesis, estabilidad y agregación de proteínas). El resultado de los experimentos realizados muestra que el diseño de proteínas guiado por la optimización mediante AG resultó en mejores soluciones que las obtenidas mediante muestreo directo, y que por su parte la optimización multiobjetivos permitió la selección de proteínas con el mejor conjunto de propiedades.

Los casos mencionados en el apartado nos ofrecen un conjunto de experiencias significativas a considerar al momento de resolver el problema planteado en este trabajo. Otras experiencias, como por ejemplo la configuración evolutiva de un AV (Wu, Cao, and Qi 2023), el ensamble de AVs (Leelarathna et al. 2023), por mencionar algunas novedosas, escapan al recorte que hemos fijado.

Capítulo 3

Autocodificadores Variacionales

En este capítulo presentamos la arquitectura del Autocodificador Variacional (AV) que empleamos para la generación de datos sintéticos. Exponemos brevemente sus fundamentos teóricos, los pasos que hemos seguidos en su implementación en este trabajo y las variaciones introducidas para su apropiada aplicación a los problemas abordados. En el capítulo siguiente nos enfocaremos en los Algoritmos Genéticos, sus fundamentos y características. Finalmente, el último capítulo expondremos los resultados obtenidos combinando ambas tecnologías para resolver problemas de selección de características.

3.1. Modelos generativos

Los modelos generativos (MG) son un amplio conjunto de algoritmos de aprendizaje automático que buscan modelar la distribución de probabilidad de datos observados $p_\theta(x)$. A diferencia de los modelos discriminantes (MD), cuyo objetivo es aprender un predictor a partir de los datos, en los modelos generativos el objetivo es *resolver un problema más general vinculado con el aprendizaje de la distribución de probabilidad conjunta de todas las variables*. Así, siguiendo a Kingma, podemos decir que *un modelo generativo simula la forma en que los datos son generados en el mundo real* (Kingma and Welling 2019). Dada estas propiedades, estos modelos permiten crear nuevos datos que se asemejan a los originales, y se aplican en tareas de generación de datos sintéticos, imputación de datos faltantes, reducción de dimensionalidad y selección de características, entre otros.

Los modelos generativos pueden tener como *inputs* diferentes tipos de dato, como imágenes, texto, audio, entre otros. Por ejemplo, las imágenes son un tipo de dato para los cuales los MG han demostrado gran efectividad. En este caso, cada dato de entrada x es una imagen que puede estar representada por un vector miles de elementos que corresponden a los valores de píxeles. El objetivo de un modelo generativo es aprender las dependencias (Doersch 2021) entre los píxeles (e.g. pixeles vecinos tienden a tener valores similares) y poder generar nuevas imágenes que se asemejen a las imágenes originales.

Podemos formalizar esta idea asumiendo que tenemos ejemplos de datos x, distribuidos según una distribución de probabilidad conjunta no conocida que queremos modelo $p_\theta(x)$ para que sea capaz de generar datos similares a los originales.

3.2. Autocodificadores

Los autocodificadores son un tipo de MG especializado en la representación de un espacio de características dado en un espacio de menor dimensión (de la Torre 2023). El objetivo de esta transformación es obtener una representación de baja dimensionalidad y la mayor fidelidad posible del espacio original. Para ello el modelo aprende a preservar la mayor cantidad de información relevante en un vector denso de menos dimensiones que las originales, y descarta -al mismo tiempo- lo irrelevante. Luego, a partir de esa información codificada, se busca reconstruir los datos observados según el espacio original.

Los autocodificadores se componen de dos partes: un *codificador* y un *decodificador*. El *codificador* es una función no lineal que opera sobre una observación x_i y la transforma en un vector de menor dimensión z, mientras que el *decodificador* opera a partir del vector z y lo transforma en una observación x_i', buscando que se asemeja a la observación original. Este vector de menor dimensión z es conocido como *espacio latente*.

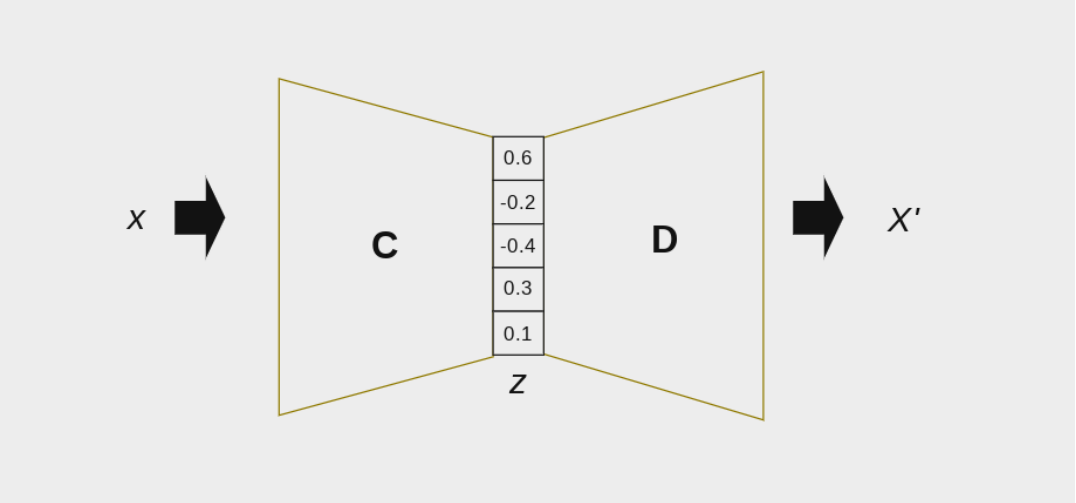

FIGURE 3.1: autocodificadores

En el proceso de aprendizaje de un autocodificador, la red modela la distribución de probabilidad de los datos de entrada x y aprende a mapearlos a un espacio latente z. Para ello, se busca minimizar la diferencia entre la observación original x_i y la reconstrucción x_i', diferencia que se denomina *error de reconstrucción*. Esta optimización se realiza a través de una *función de pérdida* que se define como la diferencia entre x_i y x_i', que permite la optimización simultánea del codificador y decodificador.

Formalmente, podemos establecer estas definiciones (de la Torre 2023):

- Sea x el espacio de características de los datos de entrada y z el espacio latente, ambos espacios son euclidianos, $x = \mathbb{R}^m$ y $z = \mathbb{R}^n$, donde $m > n$.
- Sea las siguientes funciones paramétricas $C_\theta : x \to z$ y $D_\phi : z \to x'$ que representan el codificador y decodificador respectivamente.
- Entonces para cada observación $x_i \in x$, el autocodificador busca minimizar la función de pérdida $L(x_i, D_\phi(E_\theta(x_i)))$. Ambas funciones E_θ y D_ϕ son redes neuronales profundas que se entrenan simultáneamente.

Para optimizar un autocodificador se requiere una función que permita medir la diferencia entre la observación original y la reconstrucción. Esta diferencia usualmente

se basa en la *distancia euclidia* entre x_i y x'_i, es decir, $||x_i - x'_i||^2$. La función de pérdida se define como la suma de todas las distancias a lo largo del conjunto de datos de entrenamiento. Tenemos entonces que:

$$L(\theta, \phi) = argmin_{\theta,\phi} \sum_{i=1}^{N} ||x_i - D_\phi(C_\theta(x_i))||^2$$

Donde $L(\theta, \phi)$ representa la función de pérdida que queremos minimizar: θ son los parámetros del codificador C y ϕ son los parámetros del decodificados D.

3.3. Autocodificadores y el problema de la generación de datos

En el proceso de aprendizaje antes descripto, la optimización no está sujeta a otra restricción mas que minimizar la diferencia entre la observación original y la reconstrucción, dando lugar a espacios latentes generalmente discontinuos. Esto sucede porque la red puede aprender a representar los datos de entrada de manera eficiente sin necesidad de aprender una representación continua. En la arquitectura del autocodificador no hay determinantes para que dos puntos cercanos en el espacio de características se mapeen a puntos cercanos en el espacio latente.

Esta discontinuidad en el espacio latente hace posible que ciertas regiones de este espacio no tengan relación significativa con el espacio de características. Durante el entrenamiento el modelo simplemente no ha tenido que reconstruir datos cuyas distribuciones coincidan con estas regiones. Esto es un problema en la generación de datos, ya que la red podrá generar representaciones alejadas de los datos originales. Regularmente lo que se busca en los MG, no es simplemente una generación de datos completemante igual o totalmente distintos a los orginales, sino cierta situación intermedia donde los nuevos datos introducen variaciones en características específicas.

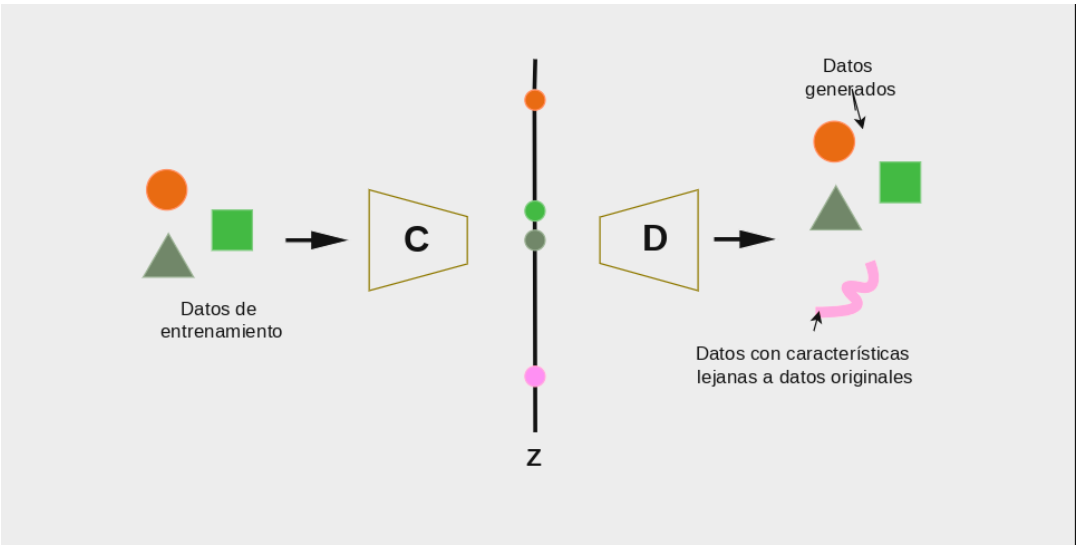

FIGURE 3.2: Discontinuidad del espacio latente

3.4. Autocodificadores Variacionales

Los Autocodificadores Variacionales (AVs) buscan resolver los problemas de discontinuidad y falta de regularidad en el espacio latente de los Autocodificadores. Comparten con éstos la arquitectura *codificador-decodificador*, pero introducen importantes

modificaciones en su diseño para crear un espacio latente continuo.

Estos modelos, a diferencia de los autocodificadores que realizan transformaciones determinísticas de los datos de entrada (codificándolos como vectores n-dimensionales), buscan modelar la distribución de probabilidad de dichos datos aproximando la distribución *a posteriori* de las variables latentes $p_\theta(z|x)$. Para ello, la codificación se produce mediante la generación de dos vectores (μ y σ) que conforman el espacio latente, a partir del cual se toman las muestras para la generación.

La red codificadora, también llamada *red de reconocimiento*, mapea los datos de entrada x a los vectores μ de medias y σ de desvíos estándar, que parametrizan una distribución de probabilidad en el espacio latente. Generalmente, esta distribución es una distribución simple, como la distribución normal multivariada. La red decodificadora, también llamada *red generativa*, toma muestras de esta distribución para generar un vector, y lo transforma según la distribución de probabilidad preexistente del espacio de características. De esta manera, se generan nuevas instancias que reflejan la probabilidad de los datos originales. Estas transformaciones implican que, incluso para el mismo dato observado (donde los parámetros de z son iguales), el dato de salida podrá ser diferente debido al proceso estocástico de reconstrucción.

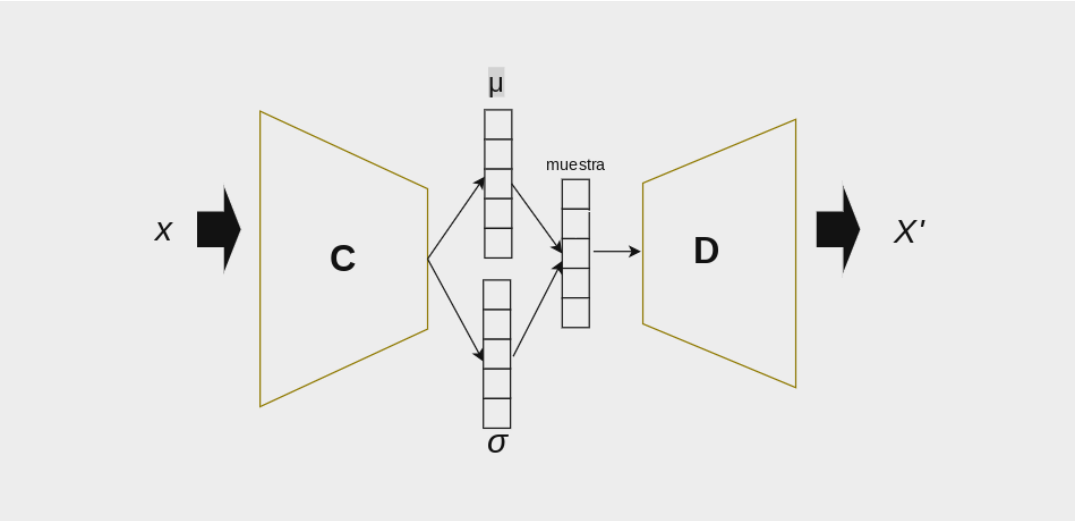

FIGURE 3.3: Autocoficadores Variacionales

Una forma de entender esta arquitectura sería relacionar los vectores que componen z como 'referecias', donde el vector de medias controla el *centro* en torno al cual se distribuirán los valores codificados de los datos de entrada, mientras que el vector de los desvíos traza el *área* que pueden asumir dichos valores en torno al *centro*.

Para indagar en estas intuiciones, veamos la solución que proponen los AV detenidamente, utilizando un enfoque formal. Así, dado un conjunto de datos de entrada $x = \{x_1, x_2, ..., x_N\}$, donde $x_i \in \mathbb{R}^m$, se asume que cada muestra es generada por un mismo proceso o sistema subyacente cuya distribución de probabilidad se desconoce. El modelo buscado procura aprender $p_\theta(x)$, donde θ son los parámetros de la función. Por las ventajas que ofrece el logaritmo[1] para el cálculo de la misma tendremos la siguiente expresión:

[1]El logaritmo convierte la probabilidad conjunta (que se calcula como el producto de las probabilidades condicionales) en una suma de logaritmos, facilitando el cálculo y evitando problemas de precisión numérica: $\log(ab) = \log(a) + \log(b)$.

$\log p_\theta(x) = \sum_{x_i \in x} \log p_\theta(x)$[2]

La forma más común de calcular el parámetro θ es a través del estimador de *máxima verosimilitud*, cuya función de optimización es: $\theta^* = \arg\max_\theta \log p_\theta(x)$, es decir, buscamos los parámetros θ que maximizan la log-verosimilitud asignada a los datos por el modelo.

En el contexto de los AVs, el objetivo es modelar la distribución de probabilidad de los datos observados x a través de una distribución de probabilidad conjunta de variables observadas y latentes: $p_\theta(x, z)$. Aplicando la regla de la cadena de probabilidad podemos factorizar la distribución conjunta de la siguiente manera: $p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$. Aquí $p_\theta(x|z)$ es la probabilidad condicional de los datos observados dados los latentes, y $p_\theta(z)$ es la probabilidad *a priori*[3] de los latentes.

Para determinar la distribución marginal respecto de los datos observados, es preciso integrar sobre todos los elementos de z, dando como resultado la siguiente función:

$p_\theta(x) = \int p_\theta(x, z)dz$ [4]

Esta distribución marginal puede ser extremadamente compleja, y contener un número indeterminable de dependencias (Kingma and Welling 2019), volviendo el calculo de la verosimilitud de los datos observados intratable. Esta intratabilidad de $p_\theta(x)$ está determinada por la intratabilidad de la distribución *a posteriori* $p_\theta(z|x)$, cuya dimensionalidad y multi-modalidad pueden hacer difícil cualquier solución analítica o numérica eficiente. Dicho obstáculo impide la diferenciación y por lo tanto la optimización de los parámetros del modelo.

Para abordar este problema, se acude a la inferencia variacional que introduce una aproximación $q_\phi(z|x)$ a la verdadera distribución *a posteriori* $p_\theta(z|x)$. Generalmente se emplea la distribución normal multivariada para aproximar la distribución *a posteriori*, con media y varianza parametrizadas por la red neuronal[5]. Sin embargo, la elección de la distribución no necesariamente tiene que pasar por una distribución normal, el único requerimiento es que sea una distribución que permita la diferenciación y el cálculo de la divergencia entre ambas distribuciones (por ejemplo si X es binaria la distribución $p_\theta(x|z)$ puede ser una distribución Bernoulli).

Así, en lugar de maximizar directamente el logaritmo de la verosimilitud (*log-verosimilitud*), se maximiza una cota inferior conocida como *límite inferior de evidencia* (*ELBO* por sus siglas en ingles). La derivación procede de la siguiente manera:

1. Log-verosimilitud marginal (intratable):

   $\log p_\theta(x) = \log \left(\int p_\theta(x, z)\, dz \right)$

2. Aplicando inferencia variacional:

[2] Esta función se lee como la log-verosimilitud de los datos observados x bajo el modelo $p_\theta(x)$ y es igual a la suma de la log-verosimilitud de cada dato de entrada x_i.

[3] La expresión *a priori* alude a que no está condicionada por ningun dato observado.

[4] Aquí dz es el diferencial de z, por lo que la expresión indica la integración sobre todas las posibles configuraciones de la variable latente.

[5] En AVs, se suele asumir que z sigue una distribución normal multivariada: $p_\theta(z) = \mathcal{N}(z; 0, I)$, con media cero y matriz de covarianza identidad. La matriz de covarianza identidad es una matriz diagonal con unos en la diagonal y ceros en los demás lugares, y su empleo simplifica la implementación del modelo, permite que las variables latentes sean independientes (covarianza = 0) y varianza unitaria, evitando así cualquier complejidad vinculada a las dependencias entre dimensiones de z.

$$\log p_\theta(x) = \log\left(\int q_\phi(z|x)\frac{p_\theta(x,z)}{q_\phi(z|x)}\,dz\right)$$

3. Aplicando la desigualdad de Jensen[6]:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{p_\theta(x,z)}{q_\phi(z|x)}\right)\right]$$

4. Descomponiendo la fracción dentro del logaritmo:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z) + \log p_\theta(z) - \log q_\phi(z|x)\right]$$

5. El resultando es el límite inferior de evidencia:

$$\log p_\theta(x) \geq \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x)\|p_\theta(z))$$

Donde:

- $\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$ es el valor esperado (*esperanza*[7]) de la log-verosimilitud bajo la aproximación variacional, y determina la precision de la reconstrucción de los datos de entrada (un valor alto de esta esperanza indica que el modelo es capaz de reconstruir los datos de entrada con alta precisión a partir de los parámetros generados por $q_\phi(z|x)$).
- $D_{\mathrm{KL}}(q_\phi(z|x)\|p_\theta(z))$ es la divergencia de Kullback-Leibler entre la distribución $q_\phi(z|x)$ y la distribución *a priori* de las variables latentes $p_\theta(z)$, y determina la regularización del espacio latente.

Maximizando esta cota inferior (*ELBO*), se optimizan simultáneamente los parámetros θ del modelo y los parámetros ϕ de la distribución empleada en la aproximación, permitiendo una inferencia eficiente y escalable en modelos con z de alta dimensionalidad [8].

El objetivo de aprendizaje del AV se da entonces por:

$$\mathcal{L}_{\theta,\phi}(x) = \text{máx}(\phi,\theta)\left(E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x)\|p_\theta(z))\right),$$

Como puede apreciearse en la ecuación anterior la función de pérdida del AV se compone de dos términos: el primero es la esperanza de la log-verosimilitud bajo la aproximación variacional y el segundo es la divergencia de Kullback-Leibler relacionada a la reconstrucción de los datos y la regularización del espacio latente. Existe

[6] Nótese que ese límite es siempre menor o igual y esto se deriva de una de las propiedades de las funciones convexas. Esta propiedad, denominada *desigualdad de Jensen*, establece que el valor esperado de una función convexa es siempre mayor o igual a la función del valor esperado. Es decir, $\mathbb{E}[f(x)] \geq f(\mathbb{E}[x])$. En el caso de funciones cóncavas, la desigualdad se invierte: $\mathbb{E}[f(x)] \leq f(\mathbb{E}[x])$. En este caso, la función logaritmo es cóncava, por lo que la desigualdad se expresa como: $\log(\mathbb{E}[x]) \geq \mathbb{E}[\log(x)]$.

[7] La esperanza es un promedio ponderado de todos los posibles valores que puede tomar una variable aleatoria, donde los pesos son las probabilidades de esos valores. En un AV, donde consideramos una distribución aproximada $q_\phi(z|x)$ para el espacio latente, la expresión citada es la esperanza de la log-verosimilitud bajo esta distribución.Aunque teóricamente esto implica un promedio sobre todas las posibles muestras z de la distribución $q_\phi(z|x)$, en la práctica, esta esperanza se estima utilizando una única muestra durante el entrenamiento por razones de eficiencia computacional. Esta única muestra permite calcular directamente $\log p_\theta(x_i|z_i)$, proporcionando una aproximación a la esperanza teórica y determinando la precisión de la reconstrucción de los datos de entrada.

[8] En la teoría, cuando derivamos el objetivo de un AV, estamos maximizando la evidencia inferior variacional (ELBO), para que la aproximación sea lo más cercana posible a la verdadera distribución de los datos. LLevado el problema a una implementación práctica generalmente se emplean optimizadores (SGD, Adam, etc.) que minimizan una función de pérdida. Para convertir el problema de maximización del ELBO en un problema de minimización, simplemente negamos el ELBO, resultando que los términos de la ecuación se reescriben como suma de cantidades positivas. El error o pérdida de reconstrucción se mide, según los casos, mediante MSE o entropía cruzada.

entre ambos términos una relación de compromiso que permite al AV aprender una representación representativa de los datos de entrada y, al mismo tiempo, un espacio latente continuo y regularizado. Cuanto mayor sea la divergencia de Kullback-Leibler, más regularizado será el espacio latente y más suave será la distribución de probabilidad de los datos generados. Cuanto menor sea la divergencia de Kullback-Leibler, más se parecerá la distribución de probabilidad de los datos generados a la distribución de probabilidad de los datos de entrada, sin embargo, el espacio latente será menos regularizado y la generación de datos mas ruidosa.

3.5. Presentación de nuestro modelo de AV

El desarrollo del modelo de Autoencodificador Variacional (AV) empleado en esta investigación se organizó en dos pasos. El primero giró en torno al diseño y validación de la arquitectura del modelo, mientras que el segundo estuvo enfocado en la optimización de dicha arquitectura para la generación de datos sintéticos en cada uno de los dataset bajo estudio. A continuación describiremos brevemente este proceso y la configuración final de los modelos elegidos para los experimentos de aumentación.

3.5.1. Modelo Inicial

En la primera etapa, se centró el esfuerzo en el diseño de la arquitectura del AV. Este proceso comenzó con la creación de una versión exploratoria del modelo, cuya finalidad era establecer una base sobre la cual iterar en mejoras sucesivas. Se optó por una red con 2 capas ocultas en el encoder y en el decoder, siguiendo la estructura básica descripta precedentemente. Esto permitiría al modelo aprender representaciones latentes complejas.

El encoder incluyó dos capas lineales, cada una seguida de una activación ReLU, un diseño que sigue la lógica de la transformación no lineal en un espacio de menor dimensión para luego reconstruir la observación original a partir del espacio latente. Como se discutió en la primera parte, el encoder genera dos vectores, uno para la media y otro para la varianza logarítmica de la distribución latente, componentes críticos para el proceso de reparametrización que permite al modelo generar nuevas muestras en el espacio latente. El decoder, encargado de reconstruir los datos originales a partir del espacio latente, fue diseñado con una estructura simétrica a la del encoder, utilizando nuevamente activaciones ReLU y finalizando con una función Sigmoidea en la capa de salida. La función Sigmoidea condiciona la salida a un rango entre 0 y 1, lo que es particularmente útil para la normalización de los datos de entrada y salida.

La función de pérdida del modelo combinó la divergencia Kullback-Leibler (KLD) y error cuadrático medio (MSE). La KLD se utilizó para medir la diferencia entre la distribución aprendida por el modelo y una distribución normal estándar, mientras que el error cuadrático medio se empleó para evaluar el error de reconstrucción, es decir, qué tan bien el modelo era capaz de replicar los datos de entrada a partir del espacio latente.

Este modelo se probó en la generación de datos sintéticos en un dataset de clases binarias: Madelon, y un dataset multiclases: GCM. Para evaluar los datos generados se realizaron experimentos de clasificación utilizando un MLP, comparando los resultados obtenidos en el dataset original y en el dataset con muestras sintéticas.

Inicialmente, la arquitectura empleada resultó insuficiente para capturar la complejidad de los datos, lo que llevó a un modelo incapaz de representar con precisión las características latentes, produciendo datos sintéticos de baja calidad.

3.5.2. Segundo Modelo AV para clases binarias

En respuesta a los problemas identificados previamente, se diseñó un nuevo modelo con una arquitectura de tres capas lineales en el encoder y el decoder, cada una con activaciones ReLU seguidas de normalización por lotes. Esta técnica de normalización fue seleccionada debido a su capacidad para estabilizar y acelerar el proceso de entrenamiento, promoviendo la rápida convergencia y mejorando la precisión de la reconstrucción. Al mitigar el problemas de desplazamiento de covariables (*covariate shift*) durante el entrenamiento, la normalización por lotes estabiliza las activaciones intermedias de la red y permite que la información relevante sea conservada a lo largo de las capas. Dado que el modelo fue ajustado para capturar la estructura de los datos a través de capas lineales y normalización por lotes, mantuvimos la elección de ReLU como función de activación dada su eficiencia computacional.

El modelo resultante fue entrenado con un optimizador Adam y una tasa de aprendizaje en el rango de [1e-5, 1e-3]. Se experimentó con diferentes tamaños del espacio latente, evaluando el equilibrio entre la calidad de reconstrucción y la capacidad de generalización del modelo. Se empleó un termino de paciencia para detener el entrenamiento si no se observaba mejora en los datos de test durante 10 épocas consecutivas. Este mecanismo de corte temprano mejoró la eficiencia del entrenamiento y la capacidad de generalización del modelo.

Estos experimentos fueron clave para ajustar el AV a las necesidades específicas de los conjuntos de datos utilizados en la investigación, permitiendo una generación de datos sintéticos que no solo replicara los patrones de los datos originales, sino que también capturara la variabilidad inherente a estos.

Arquitectura del Autocodificador Variacional

La arquitectura del Autocodificador Variacional está compuesta por tres capas lineales (una capa de entrada y dos capas ocultas), cada una seguida de una normalización por lotes (Batch Normalization) y una activación ReLU. El proceso de codificación se realiza de la siguiente manera:

$h_1 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_1}x + b_1))$
$h_2 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_2}h_1 + b_2))$
$h_3 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_3}h_2 + b_3))$

Donde:

- $\mathbf{W_1}$es una matriz de pesos que transforma el vector de entrada x al espacio de características de dimensión H.
- $\mathbf{W_2}$transforma h_1 a un espacio de características de dimensión $H2$.
- $\mathbf{W_3}$ mantiene la dimensión $H2$ mientras transformah_2.
- b_1,b_2, y$b_3$son los sesgos correspondientes a cada capa.

Después de las tres capas, se generan los vectores latentes μ y $\log(\sigma^2)$ mediante dos capas lineales independientes que también aplican normalización por lotes.

Reparametrización

El vector latente z se obtiene mediante la técnica de reparametrización, donde se introduce ruido gaussiano para permitir la retropropagación del gradiente:

$z = \mu + \sigma \times \epsilon$

Donde ϵ es una variable aleatoria con distribución normal estándar, y σ se calcula a partir de $\log(\sigma^2)$.

Decodificador

El decodificador reconstruye el vector de entrada a partir del vector latente$z$utilizando una arquitectura de tres capas lineales, cada una seguida por una normalización por lotes y una activación ReLU:

$h_4 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_4}z + b_4))$
$h_5 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_5}h_4 + b_5))$
$\hat{x} = \text{BatchNorm}(\mathbf{W_6}h_5 + b_6)$

Donde:

- $\mathbf{W_4}$ transforma el vector latente z al espacio de características de dimensión $H2$.
- $\mathbf{W_5}$ transforma h_4 al espacio de características de dimensión H.
- $\mathbf{W_6}$ transforma h_5 de regreso al espacio de la dimensión original de la entrada D_{in}.
- b_4,b_5, y$b_6$son los sesgos correspondientes a cada capa.

Finalmente, la salida $\hat{x}$ es una aproximación reconstruida de la entrada original x.

3.5.3. Modelo AVC para datos multiclase

Para abordar el dataset GCM, que contiene 14 clases con distribuciones desiguales, se creo un Autocodificador Variacional Condicional (AVC) que combina la capacidad de generación de un AV tradicional con el condicionamiento explícito en las etiquetas de clase. El AVC propuesto permitió una modelización más precisa de los datos, al incorporar información de clase en el proceso de codificación y decodificación.

En escenarios donde los datasets están desbalanceados, los modelos generativos pueden tender a favorecer las clases mayoritarias, ignorando las minoritarias. Para abordar el desbalance de clases que presenta GCM, se implementó una estrategia de ponderación de clases dentro de la función de pérdida, penalizando de manera diferenciada los errores de reconstrucción en función de la clase, mejorando así la capacidad del modelo para representar adecuadamente las clases minoritarias.

Arquitectura del Autocodificador Variacional Condicional (AVC)

La arquitectura del Autocodificador Variacional Condicional (AVC) se basa en una modificación del Autocodificador Variacional tradicional para incorporar información adicional en forma de etiquetas. Esta información se concatena tanto en la fase de codificación como en la de decodificación, permitiendo que el modelo aprenda distribuciones condicionales.

Codificador

El codificador del AVC combina la entrada original con las etiquetas antes de ser procesada por una secuencia de capas lineales (una capa de entrada y dos capas ocultas), cada una seguida por una normalización por lotes (Batch Normalization) y una activación ReLU. El proceso de codificación se realiza de la siguiente manera:

$h_1 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_1}[x, y] + b_1))$
$h_2 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_2}h_1 + b_2))$
$h_3 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_3}h_2 + b_3))$

Donde:

- $[x, y]$ es la concatenación del vector de entrada x con las etiquetas y.
- $\mathbf{W_1}$ es una matriz de pesos que transforma el vector combinado $[x, y]$ al espacio de características de dimensión H.
- $\mathbf{W_2}$ transforma h_1 a un espacio de características de dimensión $H2$.
- $\mathbf{W_3}$ mantiene la dimensión $H2$ mientras transforma h_2.

- b_1, b_2, y b_3 son los sesgos correspondientes a cada capa.

Al igual que en el Autocodificador Variacional tradicional, se generan los vectores latentes μ y $\log(\sigma^2)$ mediante dos capas lineales independientes.

Reparametrización

El vector latente z se obtiene mediante la técnica de reparametrización, similar al Autocodificador Variacional tradicional:

$z = \mu + \sigma \times \epsilon$

Donde ϵ es una variable aleatoria con distribución normal estándar, y σ se calcula a partir de $\log(\sigma^2)$.

Decodificador

El decodificador del AVC reconstruye el vector de entrada a partir del vector latente z y las etiquetas y, utilizando una arquitectura de tres capas lineales, cada una seguida por una normalización por lotes y una activación ReLU:

$h_4 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_4}[z, y] + b_4))$
$h_5 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_5}h_4 + b_5))$
$\hat{x} = \text{sigmoid}(\mathbf{W_6}h_5 + b_6)$

Donde:

- $[z, y]$ es la concatenación del vector latente z con las etiquetas y.
- $\mathbf{W_4}$ transforma el vector combinado $[z, y]$ al espacio de características de dimensión $ H2 + labels_length$.
- $\mathbf{W_5}$ transforma h_4 al espacio de características de dimensión H.
- $\mathbf{W_6}$ transforma h_5 de regreso al espacio de la dimensión original de la entrada $ D_{in}$.

- b_4, b_5, y b_6 son los sesgos correspondientes a cada capa.

Finalmente, la salida $\hat{x}$ es una aproximación reconstruida de la entrada original x, condicionada por las etiquetas y.

Elementos distintivos respecto a la arquitectura anterior:

- **Incorporación de etiquetas**: Tanto en el codificador como en el decodificador, se concatenan las etiquetas y con las entradas y el vector latente, respectivamente.
- **Dimensiones ajustadas**: Se han ajustado las dimensiones de las capas para acomodar las etiquetas, reflejadas en las matrices de pesos y las normalizaciones por lotes.
- **Capas adicionales en la fase de decodificación**: Se añaden capas y ajustes para manejar las etiquetas adicionales en el proceso de decodificación.

Configuraciones específicas para datasets desbalanceados

Para abordar el problema del desbalance de clases en el dataset GCM, se implementaron ajustes específicos en la configuración del modelo y en las estrategias de entrenamiento. El primero consistió en la asignación de pesos diferenciados a las clases dentro de la función de pérdida, con el objetivo de aumentar la penalización por errores de clasificación en las clases minoritarias. Estos pesos se calcularon de manera inversamente proporcional a la frecuencia de las clases en el conjunto de datos, lo que significa que las clases con menos instancias recibieron pesos más altos. Al incorporar estos pesos de clase en la función de pérdida, nos aseguramos que los errores en las clases con menor representación tuvieran un impacto mayor durante el proceso de optimización, logrando un aprendizaje equilibrado.

Además de ajustar la función de pérdida, se implementó una estrategia de muestreo ponderado en el proceso de entrenamiento. A través de un muestreador aleatorio ponderado, nos aseguramos que cada mini-lote de datos durante el entrenamiento tuviera una representación equilibrada de cada clase, asignando mayor probabilidad de ser seleccionadas a las instancias de clases minoritarias. Esta técnica también tuvo un papel importante en el funcionamiento del modelo, mitigando el sesgo hacia las clases más frecuentes.

Para garantizar que la combinación de ambas estrategias no tuviera un efecto indeseado en el rendimiento del modelo, se llevaron a cabo experimentos con el fin de validar la aplicación simultánea de ambas técnicas. Un efecto ha evitar era, precísamente, que el modelo se volviera excesivamente sensible a las clases minoritarias, en detrimento de su capacidad para generalizar correctamente en clases mayoritarias. Los resultados de estos experimentos mostraron que, implementadas en conjunto, ambas estrategias lograban los mejores resultados.

Esta arquitectura permite que el AVC capture relaciones condicionales más complejas entre las entradas y sus correspondientes etiquetas.

3.5.4. Optimización de Hiperparámetros

La búsqueda y ajuste de hiperparámetros para los modelos de AV y AVC ha sido un proceso crucial para optimizar la generación de datos sintéticos y, en última instancia, mejorar el rendimiento de nuestro Algoritmo Genético. A lo largo de esta etapa de investigación, se implementaron diversas estrategias para identificar las configuraciones óptimas de los modelos, así como para evitar el sobreajuste y garantizar la robustez de los resultados.

Uno de los primeros pasos fue ampliar la búsqueda de hiperparámetros, ajustando variables clave como las dimensiones latentes, las tasas de aprendizaje, y el número de neuronas en las capas ocultas. Un cambio significativo fue la implementación de un mecanismo de paciencia, configurado para detener el entrenamiento si no se observaba

mejora en los datos de test durante 10 épocas consecutivas. Esta modificación tuvo un impacto directo en la calidad del modelo, ya que en experimentos iniciales, los modelos seguían entrenándose durante todas las épocas establecidas, lo que en muchos casos resultaba en un sobreajuste. La implementación de este corte temprano no solo mejoró la eficiencia del entrenamiento, sino que también contribuyó a reducir el error y mejorar la capacidad de generalización del modelo.

El análisis de los resultados obtenidos mediante estas configuraciones reveló que un MLP entrenado con datos sintéticos generados por un AV puede igualar o incluso superar en ciertos casos el rendimiento de un MLP entrenado con datos reales. Este hallazgo es particularmente relevante, ya que sugiere que, bajo ciertas configuraciones, los datos sintéticos pueden ser tan útiles como los datos reales para el entrenamiento de modelos predictivos. Este fenómeno se observó de manera consistente en varios conjuntos de datos, como Leukemia, Madelon, y GCM, donde la precisión y la exactitud del modelo entrenado con datos sintéticos alcanzaron o superaron las métricas obtenidas con los datos originales.

3.5.4.1. Leukemia

Modelo	Clase	Precision	Recall	F1Scr	Support
MLP con datos reales	0	0.76	1.00	0.87	13
	1	1.00	0.56	0.71	9
Accuracy				0.81	22
Macro Avg		0.88	0.78	0.79	22
Weighted Avg		0.86	0.82	0.80	22
MLP con datos sintéticos	0	0.93	1.00	0.96	13
	1	1.00	0.89	0.94	9
Accuracy				**0.95**	22
Macro Avg		0.96	0.94	0.95	22
Weighted Avg		0.96	0.95	0.95	22

3.5.4.2. Madelon

Modelo	Clase	Precision	Recall	F1Scr	Support
MLP con datos reales	0	0.56	0.55	0.55	396
	1	0.54	0.55	0.55	384
Accuracy				0.55	780
Macro Avg		0.55	0.55	0.55	780
Weighted Avg		0.55	0.55	0.55	780
MLP con datos sintéticos	0	0.56	0.72	0.63	396
	1	0.59	0.42	0.49	384
Accuracy				**0.57**	780
Macro Avg		0.58	0.57	0.56	780
Weighted Avg		0.58	0.57	0.56	780

3.5.4.3. Gisette

Modelo	Clase	Precision	Recall	F1Scr	Support
MLP con datos reales	0	0.98	0.98	0.98	904
	1	0.98	0.98	0.98	896
Accuracy				**0.97**	1800
Macro Avg		0.98	0.98	0.98	1800
Weighted Avg		0.98	0.98	0.98	1800
MLP con datos sintéticos	0	0.95	0.97	0.96	904
	1	0.97	0.95	0.96	896
Accuracy				0.95	1800
Macro Avg		0.96	0.96	0.96	1800
Weighted Avg		0.96	0.96	0.96	1800

3.5.4.4. GCM

Modelo	Clase	Precision	Recall	F1Scr	Support
MLP con datos reales	0	0.14	0.25	0.18	4
	1	0.00	0.00	0.00	1
	2	1.00	1.00	1.00	3
	3	1.00	0.33	0.50	6
	4	0.89	1.00	0.94	8
	5	0.40	0.67	0.50	3
	6	0.80	0.80	0.80	5
	7	0.50	0.75	0.60	4
	8	0.33	0.25	0.29	4
	9	0.25	0.67	0.36	3
	10	1.00	0.25	0.40	4
	11	1.00	0.67	0.80	3
	12	1.00	0.25	0.40	4
	13	1.00	0.20	0.33	5
Accuracy				**0.54**	57
Macro Avg		0.67	0.51	0.51	57
Weighted Avg		0.74	0.54	0.56	57
MLP con datos sintéticos	0	1.00	0.25	0.40	4
	1	0.00	0.00	0.00	1
	2	1.00	0.67	0.80	3
	3	1.00	0.17	0.29	6
	4	1.00	1.00	1.00	8
	5	0.43	1.00	0.60	3
	6	1.00	0.80	0.89	5
	7	0.10	0.25	0.14	4
	8	0.67	0.50	0.57	4
	9	0.50	0.33	0.40	3
	10	0.00	0.00	0.00	4
	11	1.00	0.67	0.80	3
	12	1.00	0.75	0.86	4
	13	0.21	0.60	0.32	5
Accuracy				**0.54**	57

Modelo	Clase	Precision	Recall	F1Scr	Support
Macro Avg		0.64	0.50	0.50	57
Weighted Avg		0.70	0.54	0.55	57

Un hallazgo interesante se refiere a las dimensiones latentes del modelo. A medida que se ampliaba la búsqueda de hiperparámetros, se descubrió que las mejores configuraciones para la variable latente no eran necesariamente las más grandes. De hecho, en muchos casos, valores para la dimensión latente entre 3 y 100 ofrecieron los mejores resultados. Esto, que inicialmente puede ser contraintuitivo, ya que se podría suponer que un mayor espacio latente permitiría capturar más complejidad en los datos; sugiere que un espacio latente excesivamente grande puede introducir ruido y hacer que el modelo pierda la capacidad de generalizar correctamente.

Las pruebas con diferentes arquitecturas también proporcionaron información valiosa. En el caso de leukemia, se exploraron modelos AV de dos, tres y cuatro capas ocultas, así como AVC con múltiples capas, pero no se observaron mejoras significativas al aumentar la complejidad del modelo. En particular, se encontró que las configuraciones más simples (i.e. dos capas ocultas), ofrecían resultados tan buenos o incluso mejores que sus contrapartes más complejas. Esta observación refuerza la idea de que, en algunos casos, la simplicidad puede ser preferible y que el sobredimensionamiento de la arquitectura no necesariamente se traduce en mejores resultados.

Por otro lado, los experimentos realizados en el dataset GCM presentaron un desafío diferente. A pesar de la implementación de un AVC de tres capas ocultas, los resultados no mostraron mejoras sustanciales en comparación con un más simple de dos capas. Además, se observó una disminución en la capacidad del modelo para predecir correctamente clases con menor soporte en el conjunto de datos, lo que sugiere que la complejidad del modelo no fue capaz de capturar adecuadamente la variabilidad de las clases menos representadas. Este resultado subraya la dificultad inherente al trabajo con conjuntos de datos multiclase, especialmente cuando las clases tienen distribuciones subyacentes similares, están desbalanceadas o ambos.

En cuanto a la búsqueda de hiperparámetros, se utilizaron tanto Grid Search como Optimización Bayesiana (BO). Cada una de estas técnicas tiene sus fortalezas, y la elección entre ellas depende en gran medida del objetivo de la búsqueda. Grid Search, por ejemplo, permite un control total sobre el espacio de búsqueda, lo que es útil para responder preguntas específicas, como la configuración óptima de la dimensión latente. Sin embargo, la BO demostró ser particularmente eficiente en la exploración de un espacio de hiperparámetros más amplio y menos definido, logrando un equilibrio entre la exploración y la explotación que resultó especialmente útil en nuestra investigación dado el tamaño del espacio de búsqueda.

A pesar de los avances logrados, también se encontraron limitaciones. Por ejemplo, incrementar el tamaño de los datos sintéticos en leukemia no condujo a una mejora significativa en los resultados, lo que sugiere que, para ciertos datasets, los beneficios de aumentar los datos sintéticos son marginales una vez alcanzado un umbral de rendimiento. En el caso de GCM, los problemas de baja calidad en la reconstrucción de datos sintéticos por parte del AVC sugieren que simplemente aumentar el tamaño del dataset no compensa una arquitectura subóptima.

En efecto, uno de los experimentos más interesantes fue el relacionado con el aumento del tamaño del conjunto de datos sintéticos en el dataset GCM. Inicialmente, se

logró incrementar las observaciones del conjunto de datos de entrenamiento a 3000 muestras balanceadas, con 214 observaciones por clase. Este aumento resultó en una mejora significativa en la performance del modelo, logrando igualar los resultados obtenidos con el clasificador MLP entrenado con datos reales. Sin embargo, al continuar incrementando la cantidad de datos sintéticos a 6000 muestras, se observó una degradación en el rendimiento. Esto sugiere la existencia de un umbral en la cantidad de datos sintéticos que, una vez superado, introduce ruido en el modelo en lugar de aportar valor. Este ruido puede estar relacionado con el solapamiento de las fronteras de decisión en las muestras generadas, lo que aumenta el error y disminuye la precisión del modelo.

Los resultados obtenidos en los experimentos reflejan que los beneficios de la aumentación de datos tienen un límite. Superado este umbral, la generación adicional de datos no solo deja de ser útil, sino que puede ser perjudicial, como se evidenció en nuestros experimentos. Este fenómeno destaca la importancia de una cuidadosa calibración en la cantidad de datos sintéticos generados, especialmente en conjuntos de datos con características complejas y altamente dimensionales como GCM.

Otro aspecto explorado fue la implementación de la pérdida L1 en lugar de MSE. Se realizaron pruebas para evaluar si la L1_loss podría ofrecer mejoras, pero los resultados no mostraron diferencias significativas en comparación con MSE. Este hallazgo sugiere que, al menos en este contexto específico, la L1_loss no proporciona un beneficio claro sobre el MSE para la tarea de generación de datos sintéticos.

Además, se experimentó con el uso de dropout como técnica de regularización. Se probaron tasas de dropout en un rango de 0.05 a 0.5 en distintas configuraciones de AVC, tanto con arquitecturas pequeñas (100-500 neuronas por capa) como grandes (1000-7000 neuronas por capa). Aunque algunos experimentos con una arquitectura más pequeña mostraron resultados interesantes en el clasificador MLP, en conjunto este grupo de experimentos se mantuvieron por debajo de los mejores experimentos previos. Esto sugiere que el dropout, si bien útil en otros contextos, no aporta beneficios en configuraciones ya optimizadas del AVC para este tipo de tareas.

Estos resultados llevaron a una reflexión sobre la falta de impacto positivo de ciertos ajustes, como la introducción de L1_loss y dropout. Es probable que la estabilidad y el buen rendimiento de las configuraciones ya validadas de AV y AVC, alcanzados a través de numerosos experimentos, limiten el potencial de mejora adicional mediante estos métodos. De hecho, en lugar de mejorar el rendimiento, estos cambios podrían estar degradando los resultados debido a la interferencia con una configuración ya optimizada.

En resumen, la búsqueda de hiperparámetros y el ajuste de la arquitectura del AV y AVC revelaron la importancia de un enfoque balanceado que evite tanto la simplicidad excesiva como la complejidad innecesaria. Los resultados obtenidos muestran que, bajo ciertas configuraciones, los datos sintéticos pueden igualar o superar la utilidad de los datos reales en la formación de modelos predictivos, aunque la eficiencia y la calidad de estos resultados dependen en gran medida de la cuidadosa calibración de los hiperparámetros y de la adecuada elección de la arquitectura del modelo.

Capítulo 4

Algoritmos Genéticos

En este capítulo presentamos una descripción general de los AGs, exponemos brevemente sus características y describimos la implementación realizada en nuestro trabajo. Específicamente, comentaremos su componentes y operadores básicos, y presentaremos un script genérico de un AG implementado con la librería `DEAP` de Python, que puede ser adaptado para la selección de características en problemas de alta dimensionalidad. En el capítulo siguiente nos enfocaremos finalmente en los resultados obtenidos combinando ambas tecnologías para resolver problemas de selección de características.

4.1. Elementos básicos de los Algoritmos Genéticos

Los Algoritmos Genéticos (AGs) son una clase de algoritmos de optimización inspirados en la evolución biológica y en la teoría de la selección natural. Los AGs se basan en el concepto de evolución de una población de soluciones potenciales a lo largo de múltiples generaciones, utilizando operadores genéticos como la selección, el cruce y la mutación para generar nuevas soluciones y mejorar la calidad de las mismas.

En el siguiente fragmento de código presentamos las operaciones elementales de un AG, que como dice Goldberg son *extraordinariamente sencillas*. La secuencia de operaciones se inicializa con un conjunto de soluciones, denominado población. El ciclo iterativo principal del Algoritmo Genético genera nuevas soluciones candidatas descendientes mediante cruce y mutación hasta que la población esté completa. En cada iteración, los individuos son evaluados mediante una función de aptitud que mide su calidad en relación al problema a resolver (aquí, la función de aptitud es simplemente la suma de los valores de los genes, en contextos reales, esta función se ajusta a las necesidades del problema pudiendo ser una función de costo, una métrica de desempeño, entre otras). Los individuos más aptos son seleccionados para reproducirse, lo que implica la aplicación de operadores genéticos para generar nuevos individuos. Este proceso se repite a lo largo de múltiples generaciones, permitiendo que la población evolucione y se adapte a las condiciones del problema.

```
import random

# Parámetros del Algoritmo Genético
num_individuals = 5
chromosome_length = 10
num_generations = 10
mutation_rate = 0.1
```

```
# Inicializar la población con individuos aleatorios
individuals = [random.randint(0, 1) for _ in range(chromosome_length)]
population = [individuals for _ in range(num_individuals)]

# Ejecutar el Algoritmo Genético
for generation in range(num_generations):
    # Calcular aptitud
    fitness_values = [sum(ind) for ind in population]

    # Crear nueva población
    new_population = []
    while len(new_population) < num_individuals:
        # Selección de dos padres
        parent1 = random.choices(population, weights=fitness_values)[0]
        parent2 = random.choices(population, weights=fitness_values)[0]

        # Cruce de un punto
        point = random.randint(1, chromosome_length - 1)
        child1 = parent1[:point] + parent2[point:]
        child2 = parent2[:point] + parent1[point:]

        # Mutación
        for i in range(chromosome_length):
            if random.random() < mutation_rate:
                child1[i] = 1 - child1[i]
            if random.random() < mutation_rate:
                child2[i] = 1 - child2[i]

        new_population.append(child1)
        if len(new_population) < num_individuals:
            new_population.append(child2)

    # Reemplazar la población antigua con la nueva
    population = new_population

    # Mostrar la población actual y sus aptitudes
    print(f"Generación {generation + 1}:")
    for ind in population:
        print(f"Individuo: {ind}, Aptitud: {sum(ind)}")
    print()
```

A pesar de su extraordinaria simpleza -o quizas gracias a ella-, los AG constituyen algoritmos robustos, capaces de encontrar soluciones efectivas en una amplia variedad de problemas de optimización. Dicha robustez está determinada, como bien sostiene Goldberg (1989), por una serie de características distintivas, que fortalecen su configuración de búsqueda, a saber: a) operan sobre un espacio *codificado* del problema y no sobre el espacio en su representación original; b) realizan la exploración evaluando una *población de soluciones* y no soluciones individuales; c) tienen como guía una *función objetivo* (también llamada *función de aptitud*) que no requiere derivación u

otras funciones de cálculo; y d) suponen *métodos probabilísticos de transición* (operadores estocásticos) y no reglas determinísticas. Estas características permiten a los AG superar restricciones que tienen otros métodos de optimización, condicionados -por ejemplo- a espacios de búsqueda continuos, diferenciables o unimodales. Por ello, su aplicación se ha difundido notablemente, trascendiendo los problemas clásicos de optimización, aplicándose en distintas tareas (Vie, Kleinnijenhuis, and Farmer 2021) y a lo largo de diversas industrias (Jiao et al. 2023).

En lo que sigue expondremos brevemente cada una de estas características, destacando su relevancia en el contexto de los AGs y su aplicación en nuestra investigación.

4.2. a) Codificación del espacio de búsqueda

Como señalamos, los AGs se distinguen de otros algoritmos por su capacidad para operar en un espacio codificado del problema, en lugar de operar directamente el espacio en su representación original. Esto sucede gracias a la transformación de las soluciones potenciales en cadenas de datos, comúnmente conocidas como **cromosomas**, que luego son objeto de transformación mediante operadores genéticos como la mutación y el cruce. La capacidad de los AGs para operar con estas representaciones codificadas determina su adaptabilidad y eficacia en una amplia gama de problemas de optimización.

La codificación adecuada del problema es un paso inicial clave para el correcto desempeño del algoritmo. La elección de la codificación depende de la naturaleza del problema y de las características de las soluciones que se buscan optimizar. Por ejemplo, en problemas de optimización combinatoria, dado que las soluciones pueden representarse como permutaciones, una opción intuitiva en términos de codificación es la secuencias de números enteros. Por otro lado, en problemas de optimización continua, como la optimización de funciones matemáticas, las soluciones pueden representarse como vectores de números reales, lo que sugiere una codificación real-valuada. Así, la elección de la codificación adecuada en principio no tiene una respuesta única, antes bien admite múltiples alternativas confiriendo flexibilidad al diseño del AG.

Dada la importancia que tiene la codificación, es fácil advertir que así como una elección adecuada de la estrategia de codificación puede facilitar la convergencia del AG hacia soluciones óptimas, una elección inadecuada puede tener consecuencias negativas en su desempeño. En efecto, una codificación inapropiada puede llevar a una exploración ineficaz del espacio de soluciones o incluso a la generación de soluciones inviables. Así, una codificación que no preserve la viabilidad de las soluciones durante la evolución, puede resultar en la convergencia prematura del AG hacia soluciones subóptimas.

La traducción entre la representación interna codificada (genotipo) y la solución en el contexto del problema (fenotipo) es un componente importante del diseño de los AGs [1]. Este mapeo no solo permite interpretar las soluciones generadas por el algoritmo,

[1] El **genotipo** se refiere a la representación interna de una solución en el AG. Es la "cromosoma" o la estructura de datos que codifica la información genética de un individuo. En la mayoría de los casos, el genotipo se representa como una cadena de bits (0 y 1), pero también puede ser una cadena de números, caracteres, o cualquier otra estructura adecuada dependiendo del problema. Por otro lado, el **fenotipo** es la manifestación externa o la interpretación del genotipo en el contexto del problema. Es la forma en que se evalúa la solución codificada por el genotipo. El fenotipo corresponde a la solución real en el espacio de búsqueda y es lo que se evalúa mediante la función de aptitud para determinar la calidad de un individuo. Por ejemplo, en un problema de selección de características

sino que también influye en la eficacia de los operadores genéticos. Ello así, por cuanto los operadores genéticos actúan directamente sobre la representación codificada, lo que puede afectar la exploración del espacio de soluciones y la convergencia del AG.

Una de las principales ventajas de operar en un espacio codificado del problema radica en la posibilidad de aplicar operadores genéticos de manera eficiente, lo que permite una exploración exhaustiva del espacio de soluciones. En efecto, los operadores genéticos -que veremos en breve- son diseñados específicamente para actuar directamente sobre la representación codificada, generando nuevas soluciones de manera efectiva.

Un proceso típico de codificación y decodificación en un AG incluye los siguientes pasos:

1. **Espacio Original**: Representación directa del problema, por ejemplo, valores continuos o categóricos.
2. **Codificación**: Traducción del espacio original a una forma binaria o simbólica.
3. **Operadores Genéticos**: Aplicación de mutación, cruce y selección en la representación codificada.
4. **Decodificación**: Traducción inversa de la solución codificada al espacio original para evaluación.

En el caso de nuestra investigación, dada la alta dimensionalidad de los datos y la complejidad de los modelos, la codificación adecuada de las soluciones fue un proceso fundamental para garantizar que los AGs puedan encontrar soluciones óptimas o cercanas al óptimo en tiempo razonable.

4.3. b) Búsqueda por población de soluciones

Una de las características distintivas de los AGs es su enfoque en la evaluación de una **población** de soluciones en cada iteración, en lugar de centrarse en una única solución. Esta población de soluciones, también conocida como población de **individuos**, permite a los AGs explorar simultáneamente múltiples regiones del espacio de búsqueda, aumentando así la probabilidad de encontrar soluciones óptimas o cercanas al óptimo.

Como vimos en el ejemplo de más arriba, la población inicial regularmente se genera de manera aleatoria, y cada individuo dentro de esta población representa una solución potencial al problema. A lo largo de las generaciones, los AGs aplican operadores genéticos como selección, cruce y mutación para producir nuevas generaciones de individuos, mejorando iterativamente la calidad de las soluciones.

---

donde cada característica puede ser incluida o excluida. El genotipo podría ser una cadena binaria donde cada bit representa si una característica está seleccionada (1) o no (0). Genotipo: `1100101`. Aquí, el genotipo representa la selección de ciertas características en un conjunto de datos. Este genotipo incluye las características 1, 2, 4, y 7, mientras que excluye las características 3, 5, y 6. El **fenotipo** es la manifestación externa o la interpretación del genotipo en el contexto del problema. Es la forma en que se evalúa la solución codificada por el genotipo. El fenotipo corresponde a la solución real en el espacio de búsqueda y es lo que se evalúa mediante la función de aptitud para determinar la calidad de un individuo. Siguiendo el ejemplo del genotipo `1100101`, el fenotipo sería la selección efectiva de características en un conjunto de datos. Si tenemos `7` características disponibles, el fenotipo se traduciría en el subconjunto de características seleccionadas por el genotipo. Por ejemplo: Características Disponibles: `[X1, X2, X3, X4, X5, X6, X7]`, Fenotipo: `[X1, X2, X4, X7]` En este caso, el fenotipo es el subconjunto de datos que incluye solo las características seleccionadas (X1, X2, X4, X7). Este subconjunto se utilizará para entrenar un modelo, y su desempeño será evaluado para determinar la aptitud del individuo.

Un esquema general del proceso de búsqueda por población en un AG se presenta a continuación:

Esquema del proceso de búsqueda por población

1. **Población Inicial**: Generación aleatoria de un conjunto de individuos que representan soluciones potenciales.
2. **Evaluación de Población**: Cada individuo es evaluado según una función de aptitud para determinar su calidad.
3. **Operadores Genéticos**:
   - **Selección**: Elegir individuos más aptos para reproducirse.
   - **Cruce (Crossover)**: Combinar partes de dos individuos para crear uno nuevo.
   - **Mutación**: Alterar aleatoriamente un individuo para introducir variabilidad.
4. **Nueva Generación**: Creación de una nueva población basada en los individuos más aptos.
5. **Iteración**: Repetición del proceso a través de múltiples generaciones hasta alcanzar un criterio de terminación.

La diversidad genética dentro de la población es fundamental para la eficacia de los AGs, ya que permite a los algoritmos explorar de manera más exhaustiva el espacio de características y evitar la convergencia prematura hacia soluciones subóptimas. En efecto, consideremos una población homogénea donde todos los individuos son idénticos. En este caso, la capacidad del AG para explorar nuevas regiones del espacio de búsqueda se ve severamente limitada, lo que puede resultar en una convergencia temprana hacia soluciones subóptimas. Por el contrario, una población diversa, donde cada individuo representa una solución única, permite al AG explorar una variedad de soluciones y adaptarse a las condiciones cambiantes del problema.

A modo de ejemplo consideremos estas dos población de 5 individuos codificados como sigue:

Población A, con 5 individuos de longitud 5, de alta diversidad:

- Individuo 1: `11001`
- Individuo 2: `10110`
- Individuo 3: `01101`
- Individuo 4: `11100`
- Individuo 5: `00011`

Población B, con 5 individuos de longitud 5, de baja diversidad:

- Individuo 1: `11111`
- Individuo 2: `11111`
- Individuo 3: `11011`
- Individuo 4: `11010`
- Individuo 5: `11010`

Como podemos advertir en este ejemplo, cada individuo representa una solución potencial al problema, donde cada bit en la cadena codificada corresponde a una característica que puede ser seleccionada o excluida. En estas poblaciónes los individuos de A son distintos entre sí, lo que permite al AG explorar una variedad de soluciones y adaptarse a las condiciones cambiantes del problema, mientras que los individuos

de B son idénticos, lo que limita la capacidad del AG para explorar nuevas regiones del espacio de búsqueda.

4.4. c) Función de aptitud y evaluación de soluciones

La función de aptitud es el núcleo que dirige el proceso evolutivo en los AGs, determinando qué soluciones sobreviven y se propagan a la siguiente generación. Su diseño y correcta implementación son esenciales para asegurar que el AG no solo converja hacia soluciones de alta calidad, sino que también lo haga de manera eficiente y efectiva, especialmente en problemas donde las evaluaciones de aptitud son costosas o complejas.

En el proceso evolutivo de los AGs, La función de aptitud se aplica al **fenotipo** de cada solución, es decir, a su manifestación en el contexto del problema a resolver, después de que el **genotipo** (la representación codificada de la solución) ha sido transformado. Esta evaluación cuantifica qué tan bien una solución potencial cumple con los objetivos del problema, asignándole un valor numérico que refleja su desempeño relativo en comparación con otras soluciones dentro de la población.

El diseño de la función de aptitud es un aspecto crítico del proceso de modelado en los AGs, ya que guía la dirección de la búsqueda evolutiva. Específicamente, la función de aptitud debe estar alineada con los objetivos del problema, reflejando correctamente las restricciones necesarias a satisfacer. En situaciones de optimización multiobjetivo, donde varios criterios deben ser optimizados simultáneamente, es común que funciones de aptitud individuales se combinen en una única métrica a través de técnicas como la suma ponderada de los valores de aptitud individuales. En el contexto de nuestra investigación, orientada a la selección de características, la función de aptitud combina la aptitud de un individuo en términos de precisión y el tamaño del conjunto de características seleccionadas (veremos un ejemplo en breve).

En línea con lo anterior, la evaluación precisa de las soluciones mediante la función de aptitud puede constituir un proceso sujeto a multiples restricciones. Aunque la asignación de valores de aptitud más bajos a soluciones subóptimas y más altos a soluciones superiores pueda parecer un criterio ineludible, en la práctica, este proceso requiere comunmente consideraciones adicionales. Por ejemplo, en problemas con restricciones, una solución que se acerque significativamente al óptimo global pero que infrinja requerimientos esenciales del problema debería recibir una calificación de aptitud inferior a una solución factible aunque menos cercana al óptimo con el fin de orientar la búsqueda hacia soluciones viables. Con esa lógica, en la optimización multiobjetivo es necesario establecer criterios para ponderar la proximidad al óptimo, especialmente cuando los distintos objetivos compiten entre sí.

Otro aspecto importante en la función de aptitud es la minimización del número de evaluaciones necesarias para alcanzar el óptimo o una solución lo suficientemente cercana a este. En muchos casos, cada evaluación de aptitud puede ser costosa en términos de tiempo y recursos computacionales, especialmente cuando la evaluación implica la simulación de modelos complejos o el entrenamiento de algoritmos de aprendizaje automático. Por ello, reducir el número de evaluaciones de aptitud es fundamental para mejorar la eficiencia del AG, sin sacrificar la calidad de las soluciones generadas. Este aspecto fue particularmente relevante en nuestra investigación, donde la evaluación de la función de aptitud implicaba el entrenamiento y validación de modelos de clasificación en conjuntos de datos de alta dimensionalidad. La

técnica de paralelización y la evaluación diferencial de las soluciones fueron estrategias clave para reducir el tiempo de ejecución, aunque demandó una infraestructura computacional adecuada (más sobre esto en el próximo capítulo).

4.5. d) Operadores estocásticos y *esquemas* genéticos

Como hemos señalado los AGs emplean **métodos probabilísticos de transición** conformados por **operadores estocásticos**, que introducen aleatoriedad en el proceso evolutivo. Esto determina que las transformaciones dentro de un AG no siguen un camino determinista hacia la solución óptima; en su lugar, cada generación de soluciones es producto de un proceso estocástico controlado.

Los **operadores genéticos** fundamentales en este proceso son la **selección**, el **cruce (crossover)** y la **mutación**. Los mismos son responsables de la generación de nuevas soluciones, e inciden directamente en la evolución de los patrones genéticos que los AG tienden a preservar y reproducir. Patrones que se conocen como **esquemas** (1989).

Según explica Goldberg, los **esquemas** son estructuras genéticas que se repiten en la población y que influyen en la evolución de los individuos. Estos esquemas pueden ser de **orden bajo** (pocos genes) o de **orden alto** (más genes), y de **longitud de definición baja** (pocos bits) o de **longitud de definición alta** (más bits). En su operatoria, los AGs tienden a favorecer los esquemas de orden bajo y longitud de definición baja que muestran un rendimiento mejor que la media. Este fenómeno, conocido como **Teorema del Esquema**, proporciona una base para entender cómo la selección y los operadores estocásticos actúan en conjunto para guiar la evolución hacia soluciones óptimas. Veamos esto en detalle.

El operador de **selección** opera identificando y preservando los esquemas con aptitudes superiores a la media de la población. En términos probabilísticos, los esquemas con mejor aptitud tienen una mayor probabilidad de ser seleccionados y reproducidos en la siguiente generación. Esta selección basada en aptitud es clave para mantener y amplificar características beneficiosas dentro de la población. Dicho esto, la selección por sí sola no es suficiente para garantizar la exploración global del espacio de búsqueda, de ahí la importancia del cruce y la mutación.

El operador de **cruce** permite la recombinación de material genético entre dos o más soluciones. En un AG, la función principal del cruce es preservar y mejorar las características exitosas encontradas en los padres, mientras introduce suficiente variación para explorar nuevas áreas del espacio de búsqueda. Por ejemplo, en la representación binaria, un cruce de un punto dividirá dos soluciones en una posición elegida aleatoriamente y combinará segmentos de ambas para crear nuevos individuos. Este proceso asegura la transmisión de esquemas de orden bajo y longitud de definición baja, mientras introduce nuevas combinaciones genéticas que pueden llevar a soluciones más adaptativas.

Un ejemplo de cruce de un punto entre dos soluciones binarias sería:

- Padre 1: `110010`
- Padre 2: `101101`
- Punto de Cruce: `3`
- Hijo 1: `110101`
- Hijo 2: `101010`

En este caso, el cruce de un punto en la posición 3 divide los padres en dos segmentos y combina los segmentos para generar dos nuevos individuos. Este proceso de cruce permite la recombinación de material genético entre los padres, preservando y mejorando las características exitosas encontradas en ellos.

El operador de **mutación**, por su parte, introduce cambios aleatorios en las soluciones existentes, actuando como un mecanismo de perturbación que permite al AG escapar de óptimos locales y explorar más exhaustivamente el espacio de soluciones. La mutación puede variar desde simples alteraciones de bits en cadenas binarias hasta ajustes en representaciones continuas mediante la adición de ruido gaussiano. En términos del Teorema del Esquema, la mutación afecta la probabilidad de preservación de esquemas, especialmente aquellos de mayor orden, pero es crucial para asegurar que el AG mantenga la capacidad de descubrir nuevas áreas del espacio de búsqueda.

Un ejemplo de mutación en una solución binaria sería:

- Solución Original: `110010`
- Posición de Mutación: `4`
- Solución Mutada: `110110`

A esta altura ha de ser evidente que la preservación de ciertos patrones genéticos de aptitud superior es fundamental para la evolución de la población en un AG. La teoría de los esquemas, que se basa en el concepto de esquemas genéticos, proporciona un marco formal para entender cómo los operadores genéticos actúan en conjunto para guiar la evolución hacia soluciones óptimas.

Goldberg nos presenta, en relación a este punto, lo que se conoce como la **Ecuación del Esquema**, que es una herramienta teórica que permite predecir la evolución de los esquemas en una población a lo largo de múltiples generaciones. Esta ecuación tiene en cuenta factores como la aptitud de los esquemas, la probabilidad de cruce y mutación, la longitud de definición y el orden de los esquemas, y proporciona una guía para entender cómo los esquemas se propagan y se mantienen en la población.

La Ecuación del Esquema predice el número esperado de copias de un esquema H en la próxima generación $t+1$, dado su número de copias en la generación actual t. Se expresa de la siguiente manera:

$$m(H, t+1) \geq m(H, t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \frac{\delta(H)}{l-1}\right] \cdot (1 - p_m)^{o(H)}$$

Donde: - $m(H,t)$ es el número de copias del esquema H en la generación t. - $f(H)$ es la aptitud promedio de los individuos que pertenecen al esquema H. - $\bar{f}$ es la aptitud promedio de la población total. - p_c es la probabilidad de cruce. - $\delta(H)$ es la longitud de definición del esquema H, que es la distancia entre el primer y el último gen fijo en el esquema. - l es la longitud del cromosoma. - p_m es la tasa de mutación. - $o(H)$ es el orden del esquema, es decir, el número de posiciones fijas en el esquema.

Consideremos un ejemplo con los siguientes parámetros:

- Longitud del cromosoma: $l = 6$
- Esquema $H = 1 * 0 * 1 *$ (donde $*$ puede ser 0 o 1)
- Población actual tiene 100 individuos.
- $m(H,t) = 20$ (es decir, 20 individuos coinciden con el esquema H).
- Aptitud promedio de la población $\bar{f} = 15$.
- Aptitud promedio de los individuos que coinciden con el esquema H, $f(H) = 18$.
- Probabilidad de cruce $p_c = 0.7$.

- Tasa de mutación $p_m = 0.01$.
- Longitud de definición del esquema $\delta(H) = 4$ (dado que las posiciones fijas son 1, 3 y 5, la distancia entre las posiciones es 4).
- Orden del esquema $o(H) = 3$ (el número de posiciones fijas es 3).

Aplicando estos valores a la Ecuación del Esquema:

1. **Factor de Selección**:

   $\frac{f(H)}{\bar{f}} = \frac{18}{15} = 1.2$

   Esto indica que los individuos que coinciden con el esquema H tienen una aptitud superior a la media y, por lo tanto, es más probable que sean seleccionados.

2. **Probabilidad de Conservación ante el Cruce**:

   $1 - p_c \frac{\delta(H)}{l-1} = 1 - 0.7 \cdot \frac{4}{6-1} = 1 - 0.7 \cdot 0.8 = 1 - 0.56 = 0.44$

   Hay un 44 % de probabilidad de que el esquema H se conserve tras el cruce.

3. **Probabilidad de Conservación ante la Mutación**:

   $(1 - p_m)^{o(H)} = (1 - 0.01)^3 = 0.99^3 \approx 0.9703$

   El esquema H tiene aproximadamente un 97 % de probabilidad de no ser destruido por la mutación.

4. **Cálculo Final**:

   $m(H, t+1) \geq 20 \cdot 1.2 \cdot 0.44 \cdot 0.9703 \approx 20 \cdot 0.5127 = 10.254$

   Por lo tanto, en la próxima generación, se espera que haya al menos 10 copias del esquema H en la población.

Este cálculo muestra cómo el esquema H, que tiene una aptitud superior a la media y ciertas características de proximidad posicional (es decir, una longitud de definición baja), es favorecido en la reproducción y es probable que se mantenga en la población.

Con los elementos vistos hasta aquí podemos pasar, en la parte final del presente capítulo, a la implementación de un Algoritmo Genético.

4.6. Implementación de un Algoritmo Genético para la selección de características

En esta sección se describe la implementación que hemos realizado de AG, usando la librería `DEAP` de Python, para la selección de características en problemas de alta dimensionalidad. La implementación se centra en la optimización simultánea de la precisión de un modelo de clasificación y la reducción del número de características seleccionadas, utilizando operadores genéticos clásicos como el cruce y la mutación, junto con técnicas avanzadas de evaluación y selección.

La configuración inicial del AG a lo largo de los distintos experimentos que formaron parte de esta investigación (y que revisaremos en detalle en el próximo capítulo) incluyó:

- **Población inicial**: individuos generados aleatoriamente, cada uno representado como una lista de bits de longitud igual al número de características.

- **Función de aptitud**: Maximización de la precisión del modelo de clasificación y minimización del número de características activas.
- **Operadores genéticos**: Selección por torneo, cruce de dos puntos y mutación por inversión de bits.
- **Parámetros del AG**: Probabilidad de mutación (e.g.`PROB_MUT = 0.1`), probabilidad de cruce (e.g. `PX = 0.75`), número máximo de generaciones (e.g.`GMAX = 15`).
- **Evaluación de características**: Análisis de la frecuencia de activación de las características a lo largo de las generaciones.
- **Criterio de terminación**: Convergencia o número máximo de generaciones alcanzado.
- **Análisis de resultados**: Selección de las características más relevantes basadas en su frecuencia de activación.

En cada experimento, la implementación del AG comienza con la definición de los componentes básicos del algoritmo. Se define una función de aptitud (`fitness`) orientada a maximizar, que evalúa cada individuo en función de dos criterios: la precisión (`accuracy`) del modelo de clasificación entrenado con las características seleccionadas, y la fracción de características activas. Esta función de aptitud está diseñada para balancear la necesidad de un modelo predictivo preciso con la simplicidad y la eficiencia del modelo, evitando el sobreajuste y facilitando la interpretación del modelo final.

Los individuos, representados como listas de bits, se construyen utilizando una función de construcción de genes que genera un bit aleatorio basado en una probabilidad definida (`p_indpb`). Estos individuos se agrupan en una población inicial, que luego se somete a un proceso evolutivo. Durante la evolución, los individuos se seleccionan mediante la técnica de torneo, donde aquellos con mejor aptitud tienen una mayor probabilidad de ser elegidos para reproducción. Los individuos seleccionados se cruzan utilizando un operador de cruce de dos puntos (`cxTwoPoint`), que intercambia segmentos de los cromosomas de los padres para generar descendientes con combinaciones genéticas novedosas. Posteriormente, se aplica un operador de mutación que invierte los bits en el cromosoma según la probabilidad de mutación definida, asegurando que el AG mantenga la capacidad de explorar nuevas regiones del espacio de búsqueda.

A lo largo de las generaciones, el AG monitoriza y registra diversas estadísticas de la población, como la aptitud promedio, la precisión y el número de genes activos. Estas métricas permiten evaluar el progreso del algoritmo y la convergencia hacia soluciones óptimas. Al final del proceso evolutivo, se realiza un análisis de las características seleccionadas, calculando la frecuencia de activación de cada característica a lo largo de las generaciones y seleccionando las más recurrentes como las más relevantes. Este enfoque permite identificar un subconjunto óptimo de características que no solo maximiza la precisión del modelo, sino que también minimiza su complejidad.

Con fines ilustrativos, en el **Apéndice A** se presenta un script genérico de un Algoritmo Genético implementado con la librería `DEAP` de Python, que puede ser adaptado para la selección de características en problemas de alta dimensionalidad. Este script incluye la definición de los componentes básicos del AG que vimos antes, como la función de aptitud, los operadores genéticos y los parámetros del algoritmo, así como la configuración de la población inicial y la ejecución del proceso evolutivo a lo largo de 15 generaciones.

Capítulo 5

Experimentos realizados y sus resultados

En este capítulo se presentan los experimentos realizados en el marco de la investigación, cuyo objetivo principal fue evaluar la efectividad de la técnica de aumentación de datos en la selección de características mediante Algoritmos Genéticos (AGs) en contextos de escasez muestral y alta dimensionalidad. Para ello, se diseñaron y ejecutaron experimentos utilizando cuatro conjuntos de datos distintos: *Leukemia*, *Gisette*, *Madelon* y *GCM*, cada uno representando diferentes desafíos en términos de tamaño y características de los datos.

El enfoque experimental adoptado fue comparativo, contrastando el desempeño de los AGs en la selección de características utilizando datos originales frente a la misma tarea utilizando datos aumentados con muestras sintéticas generados por AVs. Los parámetros de los AGs fueron mantenidos constantes entre los grupos de experimentos con y sin aumento, permitiendo una evaluación directa del impacto de la aumentación.

La metodología seguida para cada experimento se describe a continuación. Se presentan los resultados obtenidos, incluyendo métricas como precisión en la clasificación, número de características seleccionadas y estabilidad de la selección de características a lo largo de las generaciones. Se discuten las implicaciones de los resultados y se proponen posibles ajustes y mejoras para futuros experimentos.

5.1. Leukemia

5.1.1. Metodología

Según lo visto en el Capítulo 1, el conjunto de datos *Leukemia* de expresión génica obtenidos de micro-datos de ADN es reconocido por su alta dimensionalidad (7129 mediciones) relativa al número de muestras (38 para entrenamiento y 34 para testeo). Esto lo convierte en un candidato apropiado para evaluar la capacidad de los AVs para generar datos sintéticos, a partir de los cuales aumentar el número de muestras disponibles y mejorar el desempeño de los AGs en la selección de características. Por ello, nuestros experimentos estuvieron orientadod a comparar el rendimiento de los AGs sobre los datos originales y sobre un conjunto de datos aumentado con muestras sintéticas. Comparación que se realizó en términos de precisión en la clasificación, número de características seleccionadas y estabilidad de la selección de características.

Se diseñaron entonces experimentos utilizando el modelo de AG presentado en el Capítulo 3, con una representación binaria de las características. Se utilizó una función

de aptitud basada en un clasificador MLP, que evaluó la precisión en la clasificación, penalizando el número de características seleccionadas. El objetivo del AG era encontrar un subconjunto óptimo de características que maximizara la precisión y minimizara la dimensión del espacio de características.

Como hemos señalado, la innovación en este enforque estuvo en el uso de un AV para enriquecer el proceso de entrenamiento del AG. El AV, según vimos en el Capítulo 2, fue optimizado para generar muestras sintéticas que preservaran la estructura subyacente de los datos originales, y permitiera al AG explorar un espacio de características más amplio y diverso. La idea era que, al aumentar el número de muestras disponibles, el AG pudiera identificar un subconjunto más efectivo de características, lo que se traduciría en una mejora en la precisión y la eficiencia de la selección de características.

La configuración experimental del AG que empleamos en Leukemia fue la siguiente:

- **Mutación:** PROB_MUT = 1/IND_SIZE (~0.0001)
- **Probabilidad de cruce:** PX = 0.75
- **Cromosoma activo:** p = 0.1
- **Número máximo de generaciones:** GMAX = 20

Se realizaron dos conjuntos de experimentos:

1. **Datos originales:** Se trabajó directamente con las muestras disponibles en el conjunto *leukemia*, siguiendo la partición original en un conjunto de entrenamiento y un conjunto de prueba.
2. **Datos aumentados:** Se generaron experimientos con 100, 200 y 1000 muestras sintéticas adicionales mediante un AV entrenado específicamente para este conjunto de datos.

La primera serie de experimentos con datos originales y datos aumentados tuvo por objetivo contar con una primera aproximación al problema y establecer una línea base a partir de la cual iterar con ajustes y mejoras en la configuración del AG y el AV. Luego, se realizaron experimentos adicionales para explorar diferentes configuraciones del AG y el AV, con el objetivo de identificar las condiciones óptimas para la selección de características en este conjunto de datos. Se investigó el impacto de variar la probabilidad de mutación, la probabilidad de cruce, el tamaño del cromosoma activo y el número de muestras generadas por el AV. Particularmente se exploró el impacto de la reducción del tamaño del cromosoma activo en la eficiencia de la selección de características, pasando de p = 0.01 a p = 0.005.

En el primer conjunto de experimentos, se estableció una configuración base utilizando una probabilidad de mutación de 0.01 %, una probabilidad de cruce del 75 %, y un cromosoma activo con un tamaño del 10 % del total de características, limitado a un máximo de 20 generaciones. Esta configuración fue seleccionada para equilibrar la exploración y explotación del espacio de búsqueda, asegurando que el Algoritmo Genético (AG) pudiera explorar adecuadamente las posibles combinaciones de características sin prologar el procesamiento en exceso. Para los experimentos con aumentación de datos, se generaron 100 muestras sintéticas adicionales mediante un AV. En estas primeras pruebas los resultados mostraron que tanto la precisión como el número de genes seleccionados fueron similares entre los grupos con y sin aumentación de datos. Sin embargo, se destacó una menor dispersión en los resultados del grupo con aumentación, lo que sugirió que la generación de datos sintéticos contribuyó a una mayor estabilidad del modelo.

En un segundo grupo de experimentos exploratorios, se investigaron variaciones en la configuración del cromosoma activo y el tamaño del conjunto de datos para examinar en profundidad los efectos de la aumentación. Específicamente, se realizaron pruebas con conjuntos de datos aumentados que incluían 200 y 1000 muestras sintéticas adicionales, y se redujo agresivamente el tamaño del cromosoma activo, en algunos casos hasta un 0.5 % de las características totales. Estas configuraciones más extremas fueron seleccionadas para evaluar la robustez del AG frente a diferentes tamaños del espacio de búsqueda, especialmente en escenarios donde se esperaba que la reducción dimensional comprometiera la capacidad del AG para encontrar soluciones óptimas.

5.1.2. Resultados

En los experimentos realizados sobre el conjunto de datos leukemia, los resultados mostraron leves diferencia en favor de la aumentación, tanto en lo que respecta a precisión, como al número de características seleccionadas y la estabilidad de los resultados. Aunque la diferencia en términos de precisión no es estadísticamente significativa -como puede verse en el gráfico de caja-, si se observa una mayor estabilidad.

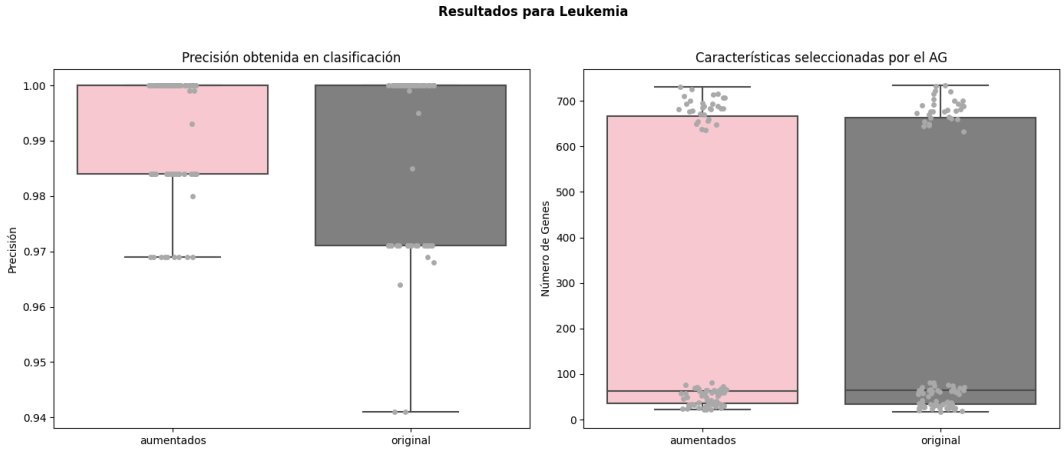

Figure 5.1: Precisión en Leukemia

Sin perjuicio de lo señalado, entendemos posible que, en conjuntos de datos donde los modelos alcanzan una precisión alta (como es el caso que nos ocupa), la aumentación no produce diferencias sustanciales en la performance del AG. La estabilidad, por otro lado, podría ser un factor importante a considerar, ya que sugiere que la aumentación puede contribuir a una selección de características más robusta y consistente a lo largo de las generaciones. Aspecto, este último, que podría ser especialmente útil en conjuntos de datos complejos.

Un punto a resaltar sobre este dataset es la alta correlación entre las características, lo que se refleja en la cantidad significativa de características seleccionadas al menos una vez durante todas las pruebas realizadas. Tanto en los experimentos con datos aumentados como en los originales, el AG seleccionó al menos una vez un número muy similar de características: 6779 y 6772. Este comportamiento sugiere que la mayoría de las características están fuertemente vinculadas, lo que permite que el AG identifique soluciones efectivas utilizando diferentes subconjuntos de características. Asimismo, se observó que un pequeño porcentaje de características (aproximadamente el 10 %) es suficiente para resolver el problema de clasificación, independientemente de cuáles sean esas características, debido a la redundancia en la información aportada por las correlaciones. El análisis cuantitativo de las correlaciones muestra que, aunque solo el

0.32 % de los pares de características posibles tienen una correlación absoluta mayor a 0.7, estos representan un número considerable (80742) de correlaciones significativas. Es decir: aunque el número de características altamente correlacionadas es una fracción pequeña del total, su impacto en la selección de características es notable, dado que el AG tiende a seleccionar conjuntos de características que son efectivamente intercambiables debido a su alta redundancia.

5.2. Gisette

5.2.1. Metodología

El conjunto de datos *Gisette* es un problema de clasificación binaria con alta dimensionalidad y un número equilibrado de muestras en ambas clases. Es un set de datos creado para trabajar el problema de reconocimiento de dígitos escritos a mano (Isabelle Guyon 2004), y tiene 13500 observaciones y 5000 atributos. Este conjunto de datos fue seleccionado para evaluar cómo la aumentación de datos afecta la selección de características en un contexto donde el espacio de características es grande, pero la relación señal-ruido es moderada.

La metodología seguida en los experimentos con *gisette* fue similar a la utilizada en *leukemia*, con la diferencia de que se exploraron configuraciones más agresivas en términos de reducción del espacio de búsqueda y generación de datos sintéticos.

La configuracion del AG fue la siguiente:

- **Mutación:** PROB_MUT = 1/IND_SIZE (~0.0002)
- **Probabilidad de cruce:** PX = 0.75
- **Cromosoma activo:** p = 0.1
- **Número máximo de generaciones:** GMAX = 30

Nuevamente aquí se realizaron dos conjuntos de experimentos: uno con datos originales y otro con datos aumentados. Se prestó especial atención a la reducción del espacio de búsqueda mediante un tamaño del cromosoma activo equivalente a 500 características. Se investigó también el impacto de generar un número mucho mayor de muestras sintéticas (6000).

5.2.2. Resultados

Los resultados en Gisette fueron similares a los observados en leukemia. La precisión media fue ligeramente superior en los experimentos con datos aumentados (0.960) en comparación con los datos originales (0.959), pero sin diferencia estadística significativa.

Este nuevo hallazgo refuerza la hipótesis de que la aumentación de datos tiene un impacto limitado en conjuntos de datos donde los modelos ya alcanzan una alta precisión. Sin embargo, al igual que en leukemia, la estabilidad de la selección de características mejoró con la aumentación, como se refleja en la menor desviación estándar y el rango intercuartílico (IQR):

	Datos aumentados	Datos originales
Características seleccionadas (media)	424.231	436.667
Desviación Estándar	17.796	14.355

	Datos aumentados	Datos originales
Rango Intercuartil (IQR)	8.000	21.500

En efecto, en Gisette se observa una diferencia más importante en la eficiencia de la selección de características. En los experimentos con datos aumentados, el número promedio de características seleccionadas fue menor (424) en comparación con los datos originales (436), señalando que el AG fue más efectivo en el reconocimiento de un subconjunto relevantes.

5.3. Madelon

5.3.1. Metodología

El conjunto de datos *Madelon* es un caso especial donde solo cinco características son relevantes, mientras que otras quince son combinaciones lineales de estas, y el resto son datos aleatorios. Así, este conjunto representa un desafío interesante para evaluar la capacidad de los AGs para identificar características útiles en un entorno donde la señal está oculta entre una gran cantidad de ruido.

La metodología seguida en los experimentos con *Madelon* fue similar a la utilizada en *Leukemia* y *Gisette*. La configuración del AG fue la siguiente:

- **Mutación:** PROB_MUT = 1/IND_SIZE (~0.002)
- **Probabilidad de cruce:** PX = 0.75
- **Cromosoma activo:** p = 0.1
- **Número máximo de generaciones:** GMAX = 30

Se generaron 2000 muestras sintéticas para incrementar el número de observaciones disponibles y evaluar si esto mejoraba la capacidad del AG para encontrar las características relevantes. Como en los experimentos precedentes, se exploraron diferentes configuraciones del cromosoma activo y se investigó cómo la aumentación de datos, en el contexto de un espacio de búsqueda complejo y ruidoso, afectaba la eficiencia de la selección de características.

5.3.2. Resultados

Los experimentos en Madelon mostraron resultados significativamente diferentes a Leukemia y Gisette. La precisión media en los experimentos con datos aumentados fue de 0.83, lo que representa un aumento del 10.4 % en comparación con la precisión de los datos originales de 0.75. Esta diferencia es estadísticamente significativa, lo que indica que la aumentación de datos tuvo un impacto positivo en el desempeño del AG.

Estos resultados sugieren que la aumentación de datos puede ser especialmente efectiva en conjuntos de datos con características complejas y un alto nivel de ruido, como es el caso de Madelon. La generación de muestras sintéticas permitió al AG identificar mejor las características relevantes, lo que se tradujo en una mejora significativa en la precisión del modelo.

Por otro lado, el análisis de selección de características reveló que, en promedio, el número de características seleccionadas fue menor en los experimentos con datos

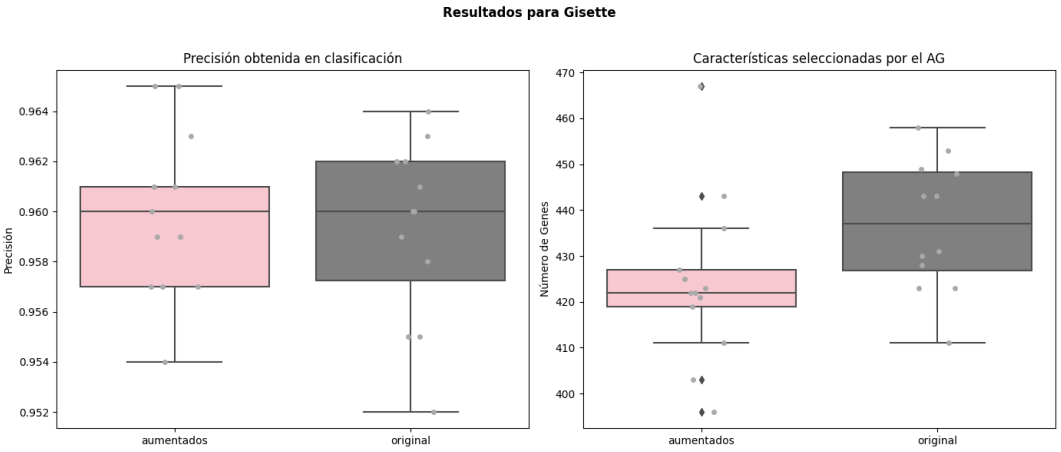

FIGURE 5.2: Precisión en Gisette

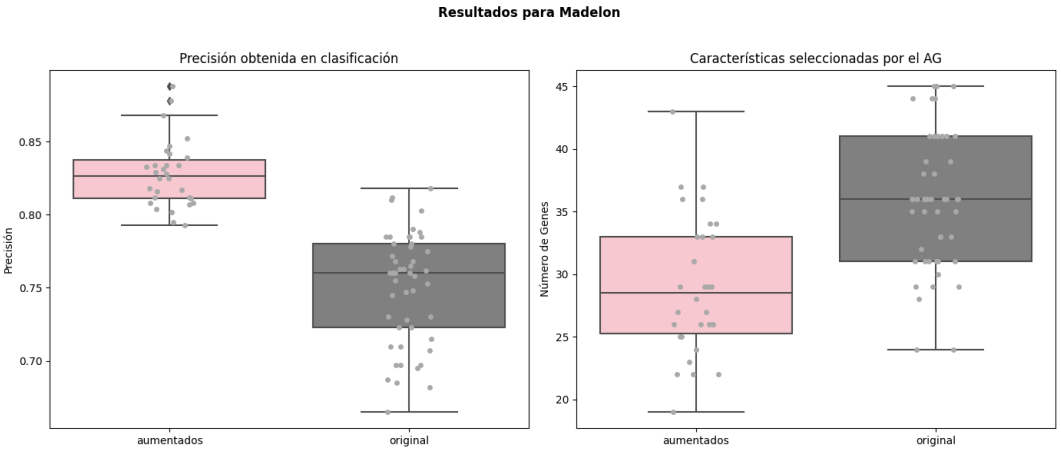

FIGURE 5.3: Precisión en Madelon

aumentados (29) en comparación con los datos originales (35). Este hallazgo, también resalta la eficiencia de la aumentación de datos en la selección de características.

Finalmente, podemos destacar también que de los algoritmos clásicos evaluados (cuyo reporte hicimos en el Capítulo 1) solo 2 de los 18 logra superan los valores de precisión obtenidos con el AG en los datos aumentados. Ambos modelos, AdaBoost y Baggind, lograron esos resultados luego de un proceso de optimización de hiperparámetros. El resto de los modelos clásicos oscila entre 0.5 y 0.7 de precisión, lo que refuerza la idea de que la aumentación de datos puede ser una estrategia efectiva para mejorar el desempeño de los AGs en conjuntos de datos complejos.

5.4. GCM

5.4.1. Metodología parte 1

El conjunto de datos *GCM* representa un desafío aún mayor que los anteriores no solo debido a la alta dimensionalidad y bajo número de muestras, sino también por las múltiples clases y el desbalance entre ellas. El dataset consiste en 16063 atributos (biomarcadores), sobre 190 muestras de tumores que refieren a 14 clases de cáncer humano.

El proceso de experimentación en GCM estuvo dividido en dos partes. En la primera parte, se realizaron experimentos exploratorios con diferentes configuraciones del AVC y el AG. El objetivo de esta etapa fue establecer una línea base para evaluar la efectividad de la aumentación de datos. En la segunda parte, se realizaron ajustes en la metodología, el modelo de AVC y el diseño de la integración AVC-AG. Esta vez, el objetivo era mejorar la calidad de los datos sintéticos y la eficiencia de la selección de características.

Aunque se realizaron una gran cantidad de experimentos (algunos exploratorios y otros más específicos), nos centraremos en los resultados de la primera y segunda parte, que nos permitieron identificar los desafíos y oportunidades en la utilización de AVs en conjuntos de datos complejos como GCM.

La cantidad de muestras sintéticas generadas en los experimentos fue ajustada a lo largo de las pruebas, conforme se exploraban opciones de mejora. Ello dio lugar a la siguiente tabla de experimentos por cantidad de muestras analizadas, donde la primera fila representa la cantidad experimentos con las muestras originales (sin aumentación):

n_muestras	cantidad_experimentos
133	20
3129	15
1322	4
1323	4
1304	2
254	2
6106	1

Al enfrentarnos a GCM asumíamos que la generación sintética de muestras y ulterior proceso de selección de características plantearía un desafío significativo al AG debido

a la complejidad del dataset. La calidad de reconstrucción lograda en los datos sintéticos generados por el AVC presentado en el Capítulo 2, aunque capaz de preservar la estructura subyacente del conjuntos de datos de GCM, no permitía suponer una mejora significativa en la precisión de la clasificación con un AG. Por otro lado, el desbalance que tienen las clases en GCM, determinaba que la calidad de reconstrucción de las clases minoritarias fuera menor, lo que podría afectar la capacidad del AG para identificar un subconjunto relevante de características. Por todo ello, esperabamos que los experimentos con datos aumentados presentaran resultados mixtos, con posibles mejoras en la estabilidad de la selección de características, pero sin mejoras significativa en la precisión.

Respecto de los recursos de hardware utilizados, dado que la aumentación de datos requeriría un poder de cómputo proporcional a al tamaño del dataset por la cantidad de muestras sintéticas generadas, se optó por utilizar un entorno en la nube para ejecutar los experimentos de manera eficiente. Para este propósito, se trabajó con una máquina virtual en Google Cloud Platform con 8 vCPUs y 30 GB de RAM.

La configuración inicial del AG fue la siguiente:

- **Mutación:** PROB_MUT = 1,16,160/IND_SIZE ~(0.0006, 0.01, 0.1)
- **Probabilidad de cruce:** PX = 0.75
- **Cromosoma activo:** p = 0.1
- **Número máximo de generaciones:** GMAX = 20

En la primera etapa se experimentó con un cromosoma activo que representaba el 10 % de las características totales, al rededor de 1600 genes por cromosoma, tanto para el grupo de experimentos con datos originales como al grupo de experimentos con datos aumentados. Esta medida se adoptó luego de realizados dos experimentos exploratorios con cromosomas activos en valores del 20 % y 30 % de las características totales, donde se observó una acentuada degradación en la precisión (en el rango de 0.2 y 0.35).

Asimismo se varió la probabilidad de mutación en tres valores: 0.0006, 0.01 y 0.1, para instar una exploración más amplia del espacio de búsqueda. La probabilidad de cruce se mantuvo constante en 0.75, y el número máximo de generaciones fue de 20.

Respecto del grupo de experimentos con datos aumentados, el dataset mixto de entrenamiento incluía 1400 muestras sintéticas (100 instancias por clase) y la partición original de entrenamiento (con 133 observaciones). La evaluación se realizó sobre los datos originales de testeo (57 observaciones).

5.4.2. Resultados parte 1

Los resultados, como anticipabamos, nos fueron favorables. La precisión media fue ligeramente superior en los experimentos con datos originales (0.538) en comparación con los datos aumentados (0.512). La estabilidad de la selección de características fue similar en ambos grupos, con una dispersión similar en la cantidad de características seleccionadas, como se puede observar en el siguiente gráfico.

Como resultado de esta primera etapa de experimentos se identificó una dificultad importante en la metodología utilizada. La calidad de los datos sintéticos generados por el AVC no contribuyó a mejorar la precisión de la clasificación, lo que sugiere que el AG no fue capaz de identificar un subconjunto relevante de características en los datos aumentados. Ante esta circunstancia se procedió a realizar un test de diagnóstico para evaluar la calidad de los datos sintéticos. Se generaron 3000 muestras,

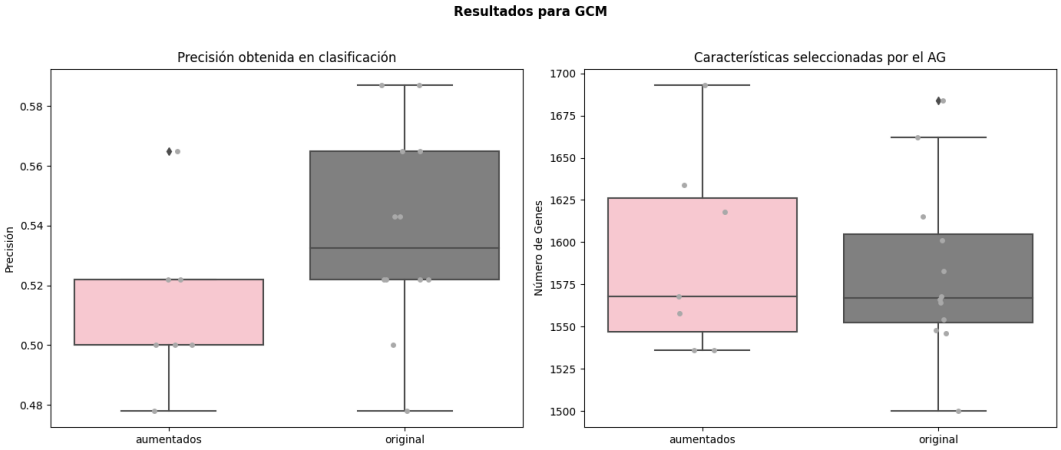

FIGURE 5.4: GCM Resultados exploratorios

se entrenó y evaluó un MLP con estos datos. Los resultados en la partición sintética de testeo fueron los siguientes:

Clase	Precisión	Recall	F1-Score	Support
0	1.00	1.00	1.00	23
1	1.00	1.00	1.00	28
2	1.00	1.00	1.00	31
3	1.00	1.00	1.00	31
4	0.97	1.00	0.98	32
5	1.00	0.97	0.99	34
6	0.96	1.00	0.98	26
7	1.00	0.97	0.98	31
8	1.00	1.00	1.00	30
9	1.00	1.00	1.00	38
10	1.00	1.00	1.00	30
11	1.00	1.00	1.00	31
12	1.00	1.00	1.00	26
13	1.00	1.00	1.00	29
Exactitud			1.00	420
Macro avg	1.00	1.00	1.00	420
Weighted avg	1.00	1.00	1.00	420

Estos resultados sugieren que el MLP fue capaz de aprender correctamente la estructura de los datos sintéticos, lo que se tradujo en una precisión del 100 % en la clasificación, sin embargo, el mismo modelo evaluado en los datos de test originales arrojó una precisión de 0.5. Esta discrepancia sugiere que la distribución de probabilidad de los datos sintéticos no es representativa de la que caracteriza a los datos reales, circuanstancia que estaría limitando la capacidad del AG para identificar un subconjunto relevante de características. Esto explicaría las motivos por los cuales los experimentos con datos aumentados no lograron mejorar la precisión de la clasificación.

En este punto, asumimos que un problema importante que tenía nuestro modelo de AVC se explicaba por la función de pérdida. En efecto,como vimos en el capítulo 2,

dicha función resulta de la combinación de dos términos: la pérdida de reconstrucción y la divergencia KL. La divergencia KL es una medida de la diferencia entre dos distribuciones de probabilidad, y se utiliza para regularizar el espacio latente, mantenerlo distribuido y compacto; mientras que la pérdida de reconstrucción mide la diferencia entre los datos originales y los datos reconstruidos. En el caso de GCM, la divergencia KL podría estar dominando la función de pérdida, lo que resultaría en una distribución latente regularizada, pero no necesariamente representativa de los datos reales. Esto significa que el modelo está produciendo una distribución convenientemente discriminada, a expensas de la precisión en la reconstrucción de los datos originales. Situación particularmente problemática cuando los datos presentan relaciones complejas como en GCM. Ante esta problemática, entendimos que se abrían distintos caminos: podíamos ajustar los pesos de la divergencia KL y la reconstrucción para que el AVC sea capaz de generar datos que sean representativos de los datos reales, o bien, explorar la aplicación del AVC a un subespacio de características seleccionadas por un AG, lo que podría reducir la complejidad del problema y mejorar la calidad de los datos sintéticos. En la próxima parte de los experimentos, exploramos esta segunda opción.

5.4.3. Metodología Parte 2

Para abordar las limitaciones observadas, se realizaron ajustes en distintas configuraciones. En primer lugar, se adaptó la función de pérdida para incluir un factor de peso por clase que permitiera rebalancear el desequilibrio entre las clases. Ello con el fin de penalizar con mayor severidad los errores en las clases minoritarias, y así procurar una representación más fiel de la distribución real de los datos. En segundo lugar, se integró un muestreador aleatorio ponderado en la etapa de entrenamiento del AVC, que permitiría ajustar los lotes de entrenamiento con igual objetivo de mejorar la representación de las clases. Finalmente, se ajustó la etapa de inicio del proceso generativo, aplicando el AVC ya no al conjunto de datos completo, sino a un subconjunto de características seleccionadas por un AG. La idea era reducir la complejidad del problema y mejorar la calidad de los datos sintéticos, permitiendo al AVC enfocarse en un espacio de características reducido y relevante. Veamos esto último en detalle.

Como vimos, una de las limitaciones observadas en los experimentos iniciales fue la dificultad del AVC para generar datos sintéticos representativos de la distribución de los datos reales. Asumimos entonces, que aplicar el AVC a un subconjunto de características seleccionadas tendría como resultado una simplificación de la distribución de probabilidad original, y por lo tanto una mejora en la calidad de los datos sintéticos. Para ello, se diseñó un flujo de procesamiento que partía de un subespacio de características relavantes, y luego procedía a la generación de datos sintéticos con un AVC y la selección de características mediante un AG, tal cual el flujo de trabajo presentado en el Capítulo 3.

La definición del subespacio de entrada partió de las características más frecuentes encontradas en los experimentos iniciales, donde se trabajó con un cromosoma activo que representaba el 10 % de las características totales, al rededor de 1600 genes por cromosoma. Este subconjunto de características, que ya habían sido preseleccionadas por un AG, se utilizó como entrada para el AVC. A partir de este subconjunto, se aplicó el modelo generativo a la creación datos sintéticos, y luego se procedió a la selección de características mediante un AG. Este flujo de trabajo, consistente en el encademiento de tres procesos: *selección-generación-selección*, permitió concentrar el aprendizaje de los modelos en un subespacio de características.

Respecto de la configuración de los experimentos se implementaron las siguientes modificaciones: se ajustó el total de muestras sintéticas generadas por el AVC a 3000.Se redujo el tamaño del cromosoma activo a 0.05 y 0.025 respecto del espacio original de características (~750 y ~450 atributos respectivamente). Se mantuvo constante la probabilidad de cruce en 0.75, como también la probabilidad de mutación en 0.1 (~0.0006). El número máximo de generaciones se redujo a 15, considerando la reducción en la complejidad del problema.

Como resultado de todas estas modificaciones, se esperaba que la calidad de los datos sintéticos mejorara, lo que se traduciría en una mejora en la precisión de la clasificación, y en la eficiencia de la selección de características.

5.4.4. Resultados Parte 2

Efectivamente, los experimentos realizados en la segunda parte de la investigación mostraron una mejora significativa en la precisión de la clasificación.

La precisión media en los experimentos con datos aumentados fue de 0.541, lo que representa un aumento de 8.85 % en comparación con la precisión de los datos originales de 0.497. Esta diferencia es estadísticamente significativa, lo que indica que la aumentación de datos tuvo un impacto positivo en el desempeño del AG.

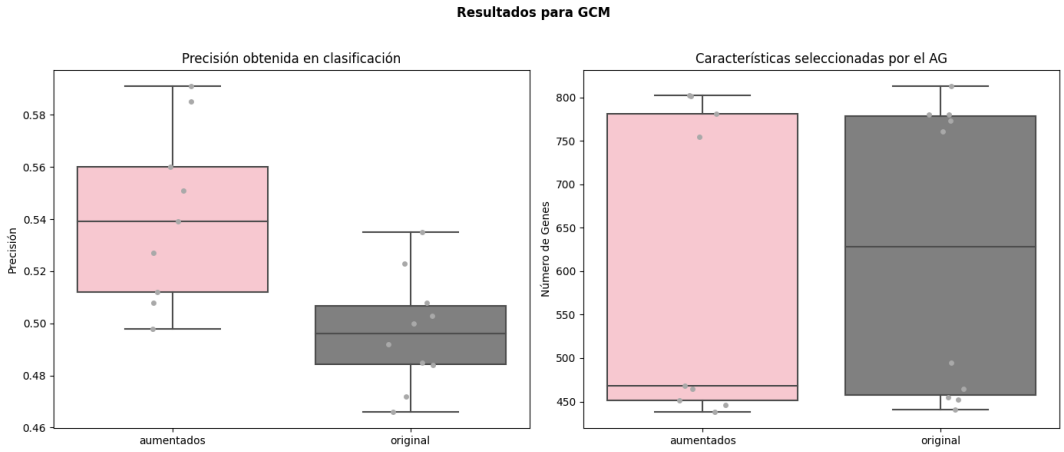

Figure 5.5: GCM Resultados parte 2

Respecto de la selección de características, no se observa gran diferencia en el número de características seleccionadas entre los experimentos con datos originales y los datos aumentados. En efecto, el número de características promedio seleccionadas en los experimentos con aumentación y cromosomas de tamaño 0.05 y 0.025 fue de 453 y 784 respectivamente, en comparación con 461 y 781 en los experimentos con datos originales.

Para validar los resultados obtenidos, se realizó un grupo de experimentos adicionales donde se disminuyó el tamaño del cromosoma activo a 0.015 y 0.003, lo que representó una selección de atributos en torno a ~200 y ~45. Los resultados mostraron que, pese a la importante reducción en el espacio de búsqueda, la precisión se mantuvo en un nivel aceptable en ambos grupos. La mejor performance del grupo con datos aumentados tiene lugar en los experimentos con reducción más agregisiva del cromosomas, con un tamaño 0.003 (~45).

A continuación presentamos la serie completa de experimentos realizados en GCM, donde se puede observar la evolución de la precisión en función del tamaño del cromosoma activo.

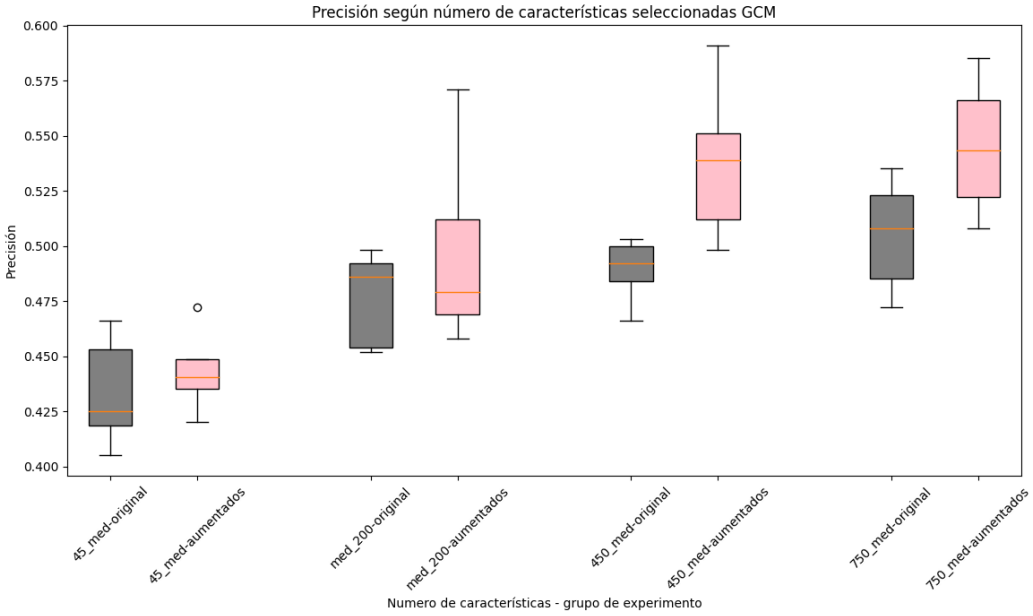

FIGURE 5.6: GCM Resultados completos

Finalmente, se realizó un experimento con 6000 muestras sintéticas, donse se observó una degradación de los resultados. Esto sugiere que la generación de un número excesivo de muestras sintéticas puede comprometer la calidad de los datos y afectar negativamente la precisión de la clasificación.

Todo lo anterior sugiere que, si bien la aumentación de datos mediante AVs puede enfrentar desafíos significativos en conjuntos de datos complejos como gcm, es posible mejorar la calidad de los datos sintéticos mediante ajustes en la arquitectura del AV y en la metodología de selección de características. La combinación de AVs y AGs en un flujo de trabajo encadenado demostró ser una estrategia prometedora para mejorar la precisión y la eficiencia en la selección de características.

5.5. All-Leukemia

Con el fin de validar el tratamiento dado a GCM, particularmente la estrategia de encadenamiento de procesos de *selección-generación-selección*, se decidió aplicar la misma metodología a un nuevo dataset *All-Leukemia*. Este dataset, que contiene 12600 variables (i.e. genes) y 327 muestras, es un caso de estudio clásico en la literatura de clasificación de tumores.

Se repitió el procedimiento de busqueda de la mejor combinación de hiperparámetros para el AVC, con el fin de lograr la mejor reconstrucción de los datos originales. Se mantuvo la misma configuración del modelo AVC utilizado en GCM, descripto en el Capítulo 2.

Al igual que en GCM, se realizaron experimentos con datos originales y aumentados. Los experimentos con aumentación consistieron en la generación de 1000 muestras sintéticas creadas por un AVC entrenado en un subespacio de características relevantes, seleccionadas por un AG. Tanto para la generación como para la selección siempre se trabajó con las particiones de datos originales de entrenamiento y teteo.

Se probaron 3 escenarios de reducción de características, con un ratio de genes activos de 0.003, 0.015 y 0.025, dando como resultado un tamaño de cromosoma a torno a las 35, 180 y 300 variables. El algoritmo genético utilizado en esta oportunidad realizaba 15 generaciones.

Los resultados generales se pueden observar en el siguiente gráfico.

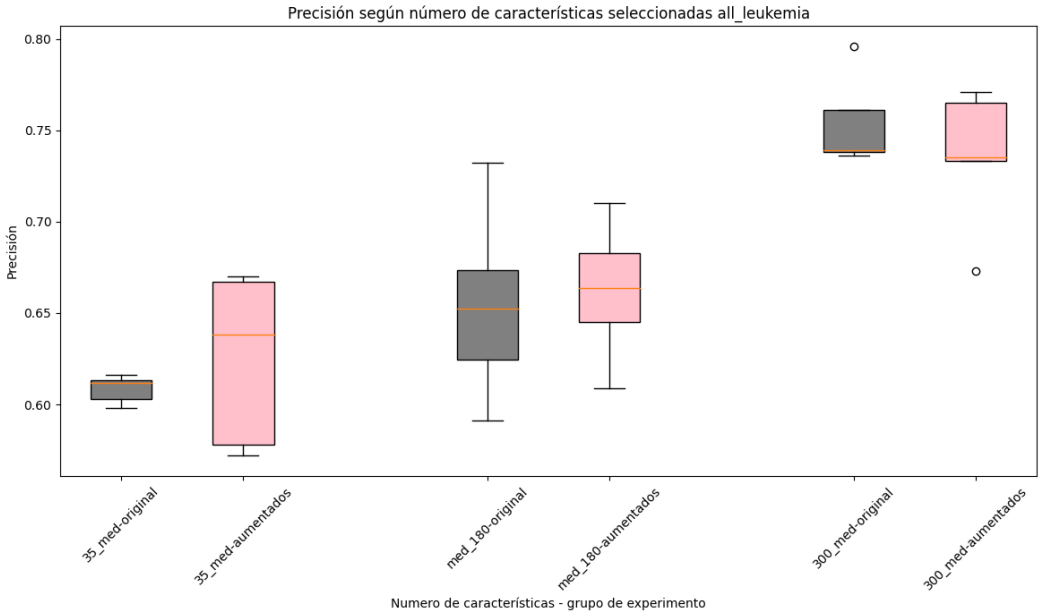

FIGURE 5.7: All-Leukemia Resultados

Como puede advertirse, los experimentos muestran que la estrategia de encadenamiento de procesos de *selección-generación-selección* presenta una ligera mejora en la precisión de clasificación en los casos de drástica reducción de características en torno a los 35 y 200 genes activos. No se evidencia diferencia significativa entre los experimentos con 300 genes activos.

Respecto de la eficacia en la selección de características, no se observa una diferencia significativa entre los experimentos con datos originales y los datos aumentados como puede apreciarse en los siguientes gráficos.

En el caso del experimento con 35 genes activos (promedio) se advierte, pese a la mejora en la precisión, una leve degradación en la estabilidad de los resultados. Entendemos que esto se explica por la reducción en la cantidad de genes activos y la sensibilidad del AG a parámetros como la probabilidad de mutación. En este caso, el experimento tuvo una probabilidad de mutación de 0.028, operando una importante variabilidad en la estructura de los cromosomas y afectando la estabilidad de la selección.

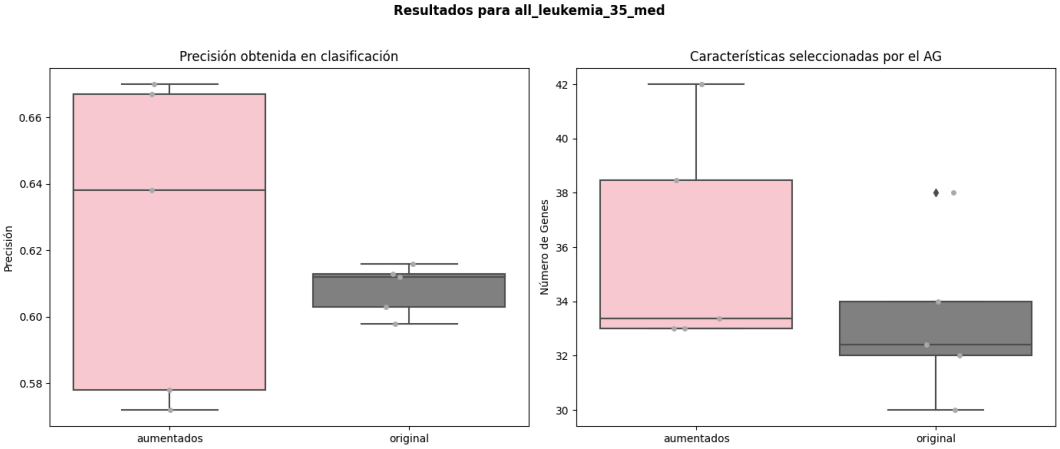

FIGURE 5.8: All-Leukemia Resultados

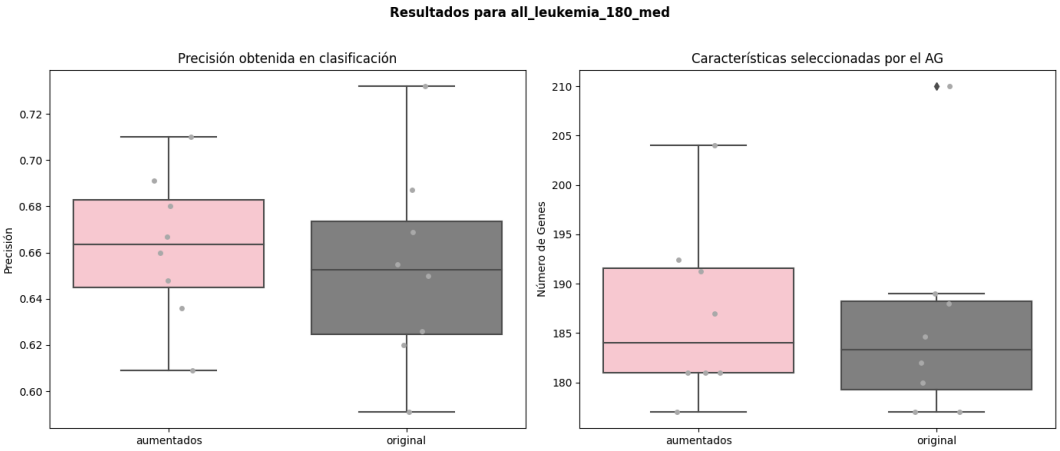

FIGURE 5.9: All-Leukemia Resultados

5.6. Resumen de los resultados

Los experimentos realizados en los cuatro conjuntos de datos (*Leukemia*, *Gisette*, *Madelon* y *GCM*) evidencian una tendencia general sobre el impacto de la aumentación de datos mediante Autocodificacadores Variacionales en la selección de características utilizando Algoritmos Genéticos.

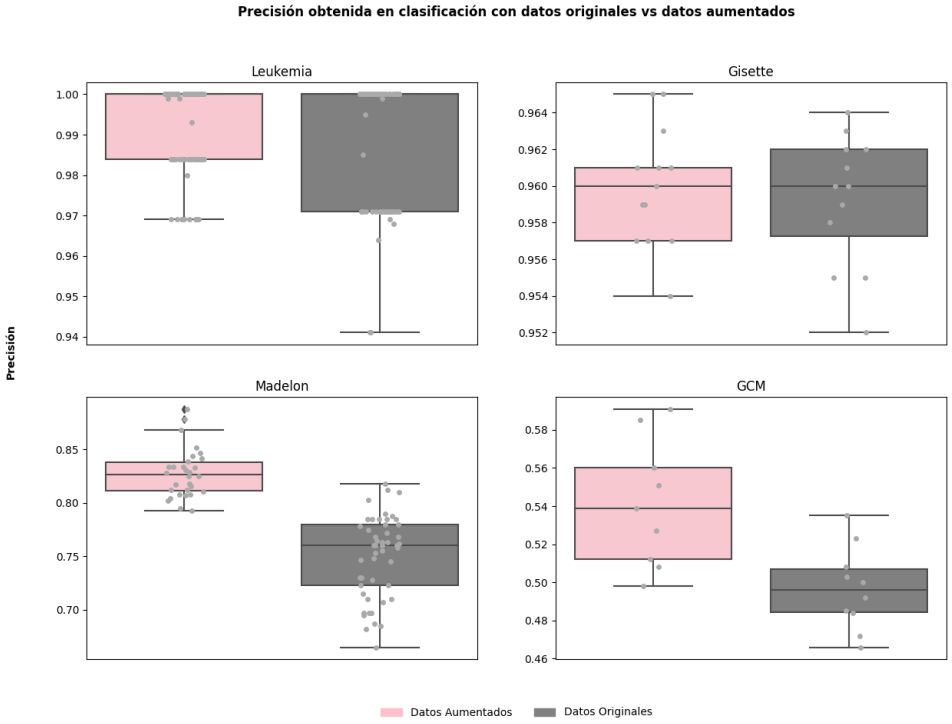

FIGURE 5.10: Precisión en los 4 datasets

Dicha tendencia tiene confirmación en los resultados obtenidos en el dataset *All-Leukemia*:

El uso de datos aumentados no siempre produjo mejoras estadísticamente significativas en la precisión de clasificación, pero sí marcó una diferencia relevante ante los casos más difíciles. En efecto, en problemas con métricas saturadas su aporte agrega un valor marginal, asociado a la estabilidad de los resultados, como se vió en *Leukemia* y *Gisette*. Ambos datasets representan desafíos donde los modelos ya alcanzan una alta performance procesando los datos en su estado original. Sin embargo, en dataset más complejos, donde el margen de majora en las predicciones es mayor, como los casos de *Madelon*, *GCM* y *All-Leukemia*, la técnica de aumentación generó resultados positivos, particularmente en los dos primeros donde se observó un salto del 9 % y 10 % en el resultado final. Este hallazgo sugiere que la aumentación de datos puede contribuir a generar modelos más consistentes a lo largo de múltiples generaciones, lo cual es particularmente relevante en escenarios de alta dimensionalidad y escasez muestral.

En el caso del conjunto de datos *Madelon*, donde el problema de selección de características presenta una clara distinción entre señales relevantes y ruido, la aumentación de datos mostró una mejora significativa en la precisión, incrementando la capacidad

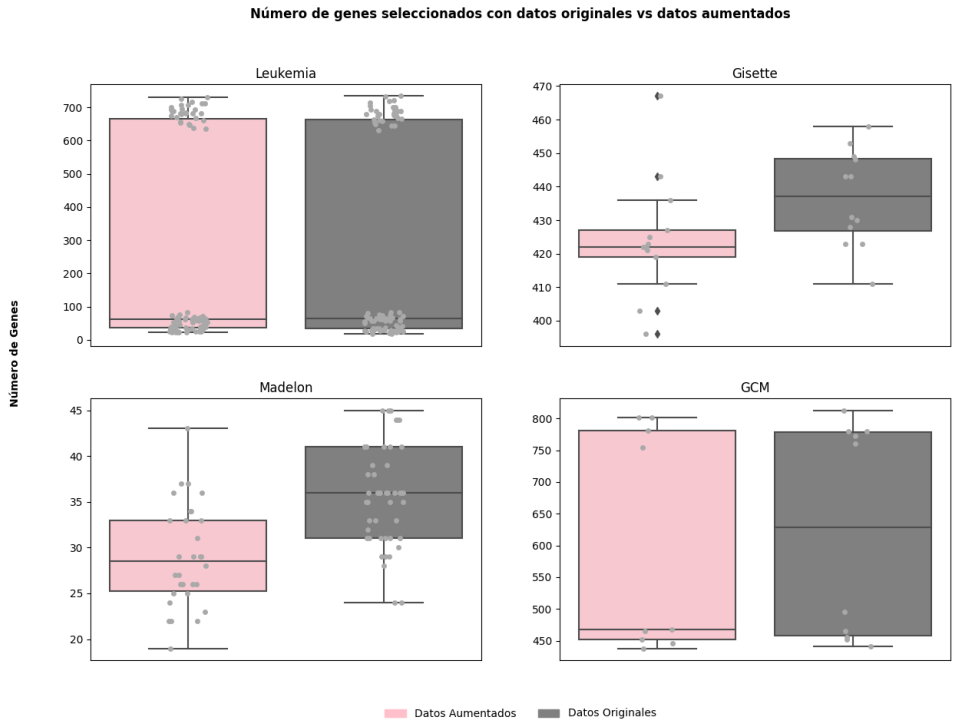

FIGURE 5.11: Número de características seleccionadas en los 4 datasets

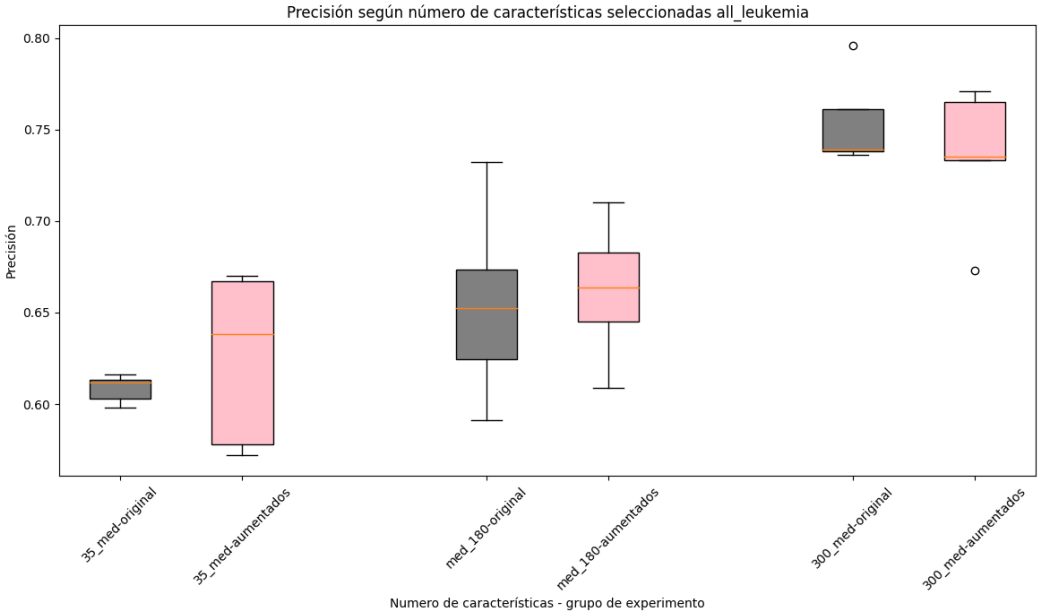

FIGURE 5.12: Precisión en el dataset de validación

del AG para identificar las características correctas en un espacio de búsqueda extremadamente ruidoso. Este resultado resalta la utilidad de la aumentación en problemas donde la presencia de señales distractoras dificulta la tarea de selección.

Por otro lado, los experimentos con *GCM* ilustran las limitaciones de la aumentación en conjuntos de datos con una distribución de clases desbalanceada y de alta complejidad. Sin embargo los ajustes en la arquitectura del AVC y en la configuración experimental permitieron mejoras importantes, alcanzando una generación de datos sintéticos de mayor calidad y una selección de características más eficiente. Resultados que se validaron mediante los experimentos en *All-Leukemia*. En este sentido, la integración de estrategias más complejas, como la combinación de selección de características previa a la aumentación, demostró ser una vía prometedora para mejorar los resultados.

En conjunto, los hallazgos de este capítulo sugieren que la aumentación de datos, cuando se utiliza en combinación con AGs, puede ser una herramienta efectiva en la selección de características, especialmente en contextos donde la dimensionalidad y el ruido dificultan la tarea. Asimismo, se destaca que el éxito de esta técnica depende críticamente de la calidad de los datos sintéticos generados y de la adecuada configuración de los modelos subyacentes, lo que subraya la importancia de ajustar las metodologías de manera específica para cada tipo de conjunto de datos.

Capítulo 6

Próximos Pasos: complejización del modelo y exploración de nuevas arquitecturas

A partir de los resultados obtenidos en los experimentos, se torna evidente que la aumentación de datos mediante Autocodificadores Variacionales y su combinación con Algoritmos Genéticos es una estrategia prometedora en la selección de características. Para maximizar el potencial de este enfoque, es posible avanzar en varias direcciones, tanto en términos de complejización de los modelos AV y AVC, como en la integración y encadenamiento de modelos más complejos. En este capítulo, abordaremos los próximos pasos necesarios para continuar mejorando la eficiencia y precisión de los modelos aplicados en escenarios de alta dimensionalidad y ruido, explorando nuevas arquitecturas y estrategias de integración.

6.1. 1. Complejización del modelo AV

Entendemos que uno de los primeros pasos para mejorar la calidad de los datos sintéticos generados es la complejización de los modelos AV y AVC. Aunque las versiones creadas en el marco de la presente investigación han mostrado ser efectivas en los contextos de experimenatión planteados, como en *Madelon* y en *GCM*, es necesario optimizar su capacidad para capturar mejor las estructuras subyacentes de los datos reales. Para lograr esto, se proponen dos líneas de trabajo:

6.1.1. Mejora en la función de pérdida

La función de pérdida del AV/AVC desempeña un papel fundamental en la calidad de las muestras generadas. Como vimos en los experimentos de GCM, la divergencia KL puede dominar el proceso de regularización del espacio latente, lo que lleva a una reconstrucción insuficiente de los datos originales. Para mitigar este problema, propondríamos -tal cual lo adelantado en el Capítulo 4- una modificación de la función de pérdida, donde se ajusten los pesos entre la pérdida de reconstrucción y la divergencia KL, dando mayor importancia a la precisión en la reconstrucción. Adicionalmente, se pueden explorar técnicas como la Inferencia Variacional Recocida (*Annealed Variational Inference*, Huang et al. (2018)), que ajusta gradualmente el peso de la divergencia KL durante el entrenamiento, permitiendo una transición más suave y efectiva en la regularización del espacio latente.

6.1.2. Uso de AVs jerárquicos

Otra línea de trabajo posible es la utilización de Autocodificadores Variacionales Jerárquicos (Vahdat and Kautz (2020)). Estos modelos permiten la representación de características en múltiples niveles de abstracción, lo que podría ser particularmente útil para capturar relaciones complejas en conjuntos de datos como *GCM*. Al incorporar un nivel adicional de complejidad, los HVAEs podrían generar datos sintéticos que no solo preserven mejor la estructura de los datos originales, sino que también mejoren la capacidad del AG para identificar características relevantes en escenarios de alta dimensionalidad.

6.2. 2. Integración AV-AG optimizada

Aunque los resultados iniciales con la integración AV-AG son alentadores, es posible explorar maneras más eficientes de combinar ambos modelos. El flujo de trabajo encadenado que involucra *selección-generación-selección* mostró ser efectivo en los experimentos con *GCM*, pero podría beneficiarse sustancialmente con ciertos ajustes adicionales en su implementación.

6.2.1. Selección dinámica de características

Así, una opción es, en lugar de aplicar el AG sobre la totalidad de las características de manera uniforme, podríamos implementar un proceso dinámico de selección de características en múltiples etapas. En este enfoque, el AG se aplicaría inicialmente sobre subconjuntos reducidos de características, optimizando en función de la estabilidad y relevancia de los atributos seleccionados. Finalmente, se combinarían los subconjuntos con mejor desempeño, para formar un dataset de baja dimensiones y alto contenido informativo. Posteriormente, los AVs generarían datos sintéticos solo tomando como imputs este último dataset, reduciendo áun más el ruido y permitiendo una exploración más eficiente del espacio de búsqueda. Todo esto, alineado con la técnica implementada en este trabajo, pero radicalizada en su intención y objetivo.

6.2.2. Optimización conjunta de AV y AG

En los experimentos actuales, el AV y el AG se entrenan de manera secuencial y separada, lo que puede limitar la sinergia entre ambos modelos. Un enfoque alternativo sería la optimización conjunta, donde los parámetros de ambos modelos se ajusten simultáneamente durante el entrenamiento. De esta forma, el AV/AVC podría generar muestras sintéticas que maximicen directamente la eficacia del AG en la selección de características, permitiendo una retroalimentación continua entre ambos procesos y una mejora en la calidad del conjunto de datos aumentado.

Reconocemos que, según la experiencia ganada en esta investigación, ajustar los parámetros del AVC y AG, aún de forma independiente, no es un proceso trivial. Particulamente la cantidad de muestras sintéticas en la etapa generativa y el tamaño del cromosoma activo en la etapa de selección resultan parámetros de un impacto crítico en los resultados de la arquitectura.

Pero advertimos también que, en el diseño donde el AV genera muestras sintéticas que luego son utilizadas por el AG para la selección, los modelos no comparten información durante sus respectivas fases de entrenamiento. La **optimización conjunta** permitiría entrenar ambos modelos de forma simultánea, posibilitando una retroalimentación directa y continua entre los procesos de generación de datos sintéticos (por

el AV) y la selección de características (por el AG). Es decir, se ajustaría el proceso de generación de datos en función de la capacidad del AG para seleccionar características relevantes, logrando que el AV genere datos específicamente diseñados para maximizar el rendimiento del AG.

Aunque esta propuesta entaña desafíos inocultables (diseño de una función de pérdida, coordinación entre los procesos de optimización, capacidad computacional, por mencionar algunas), también ofrecería beneficios de gran valor. En particular, podríamos disponer de una mayor sinergia entre generación y selección. Esto se produciría a través de una retroalimentación directa entre los modelos, haciendo que el AV se adapte mejor a las necesidades del AG, generando datos que aborden mejor los desafíos específicos de selección de características en cada problema. Al entrenar el AV para generar datos que optimicen el desempeño del AG, el proceso de búsqueda de características relevantes podría volverse más eficiente. El AG podría explorar de manera más efectiva el espacio de características, identificando subconjuntos más precisos y reduciendo la dimensionalidad sin perder información relevante.

6.3. 3. Exploración de encadenamientos y stacks de modelos

Los resultados obtenidos sugieren que una única iteración del proceso AV-AG puede no ser suficiente para capturar completamente las relaciones entre características en problemas altamente complejos. En este contexto, la construcción de arquitecturas más complejas mediante el encadenamiento de varios modelos (stacks) o la integración de ensambles, similar a los Bosques Aleatorios, se presenta como una vía interesante de investigación.

Una posibilidad es el diseño de un *stack* de AVs y AGs, donde varias instancias de ambos modelos se encadenen de manera secuencial o paralela. En este esquema, por ejemplo, una primer secuencia AG-AV generaría un conjunto de datos sintéticos, que luego alimentaría a una segunda secuencia de AG-AG con configuraciones más específicas. Este proceso de encadenamiento podría permitir una mayor concentración de la generación y selección, especialmente en conjuntos de datos donde las relaciones entre las características son extremadamente complejas.

Al igual que los Bosques Aleatorios combinan múltiples árboles de decisión para mejorar la precisión y la robustez del modelo, se puede explorar la creación de un ensamble de AVs y AGs. Este enfoque involucraría el entrenamiento de múltiples AVs y AGs con diferentes configuraciones y subconjuntos de datos, cuyas salidas se combinarían para producir una solución más robusta. Los ensambles suelen ser efectivos para reducir la varianza de los modelos individuales, lo que podría resultar en una selección de características más estable y en un rendimiento más consistente.

6.4. 4. Exploración de arquitecturas híbridas y meta-aprendizaje

Por último, se abre la posibilidad de explorar arquitecturas híbridas que combinen AVs y AGs con otros enfoques de aprendizaje automático, como los algoritmos de meta-aprendizaje. Estos modelos podrían ser entrenados para aprender a seleccionar automáticamente los mejores hiperparámetros y configuraciones para cada conjunto de datos, adaptándose dinámicamente a las características específicas del problema.

En lugar de fijar los parámetros del AG a priori, el meta-aprendizaje permitiría que el AG aprenda automáticamente cuáles son los mejores parámetros en función de la estructura de los datos. Este enfoque podría incluir la selección adaptativa del tamaño del cromosoma activo, las tasas de mutación y cruce, y el número de generaciones, optimizando el rendimiento del AG en cada iteración.

6.5. Conclusión

La investigación hasta el momento ha demostrado que la combinación de Autocodificadores Variacionales y Algoritmos Genéticos es una estrategia prometedora para la selección de características en escenarios de alta dimensionalidad y ruido. Sin embargo, los resultados también sugieren que existe un espacio amplio para la creatividad y mejora mediante la complejización de los modelos AV, la optimización de la integración AV-AG, y la exploración de arquitecturas más avanzadas basadas en encadenamientos y ensambles de modelos. Considerando la complejidad de los dataset reales, quizas los próximos pasos deberían priorizar la construcción de soluciones más robustas y adaptativas, capaces de abordar la complejidad de los problemas con la menor cantidad de presupuestos posibles.

References

Ai, Qingzhong, Pengyun Wang, Lirong He, Liangjian Wen, Lujia Pan, and Zenglin Xu. 2023. "Generative Oversampling for Imbalanced Data via Majority-Guided VAE." February 14, 2023. http://arxiv.org/abs/2302.10910.

Almugren, Nada, and Hala Alshamlan. 2019. "A Survey on Hybrid Feature Selection Methods in Microarray Gene Expression Data for Cancer Classification." *IEEE Access* 7: 78533–48. https://doi.org/10.1109/ACCESS.2019.2922987.

Blaauw, Merlijn, and Jordi Bonada. 2016. "Modeling and Transforming Speech Using Variational Autoencoders." In *Interspeech 2016*, 1770–74. ISCA. https://doi.org/10.21437/Interspeech.2016-1183.

Blagus, Rok, and Lara Lusa. 2013. "SMOTE for High-Dimensional Class-Imbalanced Data." *BMC Bioinformatics* 14 (1): 106. https://doi.org/10.1186/1471-2105-14-106.

Bolón-Canedo, Verónica, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. 2015. *Feature Selection for High-Dimensional Data*. Artificial Intelligence: Foundations, Theory, and Algorithms. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-21858-8.

Doersch, Carl. 2021. "Tutorial on Variational Autoencoders." January 3, 2021. http://arxiv.org/abs/1606.05908.

El-Hasnony, Ibrahim M., Sherif I. Barakat, Mohamed Elhoseny, and Reham R. Mostafa. 2020. "Improved Feature Selection Model for Big Data Analytics." *IEEE Access* 8: 66989–7004. https://doi.org/10.1109/ACCESS.2020.2986232.

Fajardo, Val Andrei, David Findlay, Charu Jaiswal, Xinshang Yin, Roshanak Houmanfar, Honglei Xie, Jiaxi Liang, Xichen She, and D. B. Emerson. 2021. "On Oversampling Imbalanced Data with Deep Conditional Generative Models." *Expert Systems with Applications* 169 (May): 114463. https://doi.org/10.1016/j.eswa.2020.114463.

Fisher, Ronald. 1935. *The Design of Experiments*. London: Oliver and Boyd.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York, NY, USA: Addison-Wesley.

Golub, T. R., D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, et al. 1999. "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring." *Science* 286 (5439): 531–37. https://doi.org/10.1126/science.286.5439.531.

Huang, Chin-Wei, Shawn Tan, Alexandre Lacoste, and Aaron Courville. 2018. "Improving Explorability in Variational Inference with Annealed Variational Objectives." arXiv.org. September 6, 2018. https://arxiv.org/abs/1809.01818v3.

Isabelle Guyon, Steve Gunn. 2004. "Gisette." UCI Machine Learning Repository. https://doi.org/10.24432/C5HP5B.

Jiao, Ruwang, Bach Hoai Nguyen, Bing Xue, and Mengjie Zhang. 2023. "A Survey on Evolutionary Multiobjective Feature Selection in Classification: Approaches, Applications, and Challenges." *IEEE Transactions on Evolutionary Computation*, 1–1. https://doi.org/10.1109/TEVC.2023.3292527.

Khmaissia, Fadoua, and Hichem Frigui. 2023. "Confidence-Guided Data Augmentation for Improved Semi-Supervised Training." February 21, 2023. http://arxiv.org/abs/2209.08174.

Kingma, Diederik P., and Max Welling. 2019. "An Introduction to Variational Autoencoders." *Foundations and Trends® in Machine Learning* 12 (4): 307–92. https://doi.org/10.1561/2200000056.

Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, et al. 2017. "Overcoming Catastrophic Forgetting in Neural Networks." *Proceedings of the National Academy of Sciences* 114 (13): 3521–26. https://doi.org/10.1073/pnas.1611835114.

Kwarciak, Kamil, and Marek Wodzinski. 2023. "Deep Generative Networks for Heterogeneous Augmentation of Cranial Defects." August 9, 2023. http://arxiv.org/abs/2308.04883.

Latif, Siddique, Rajib Rana, Junaid Qadir, and Julien Epps. 2020. "Variational Autoencoders for Learning Latent Representations of Speech Emotion: A Preliminary Study." July 27, 2020. https://doi.org/10.48550/arXiv.1712.08708.

Lecun, Yann. 1998. "Gradient-Based Learning Applied to Document Recognition." *PROCEEDINGS OF THE IEEE* 86 (11).

Leelarathna, Navindu, Andrei Margeloiu, Mateja Jamnik, and Nikola Simidjievski. 2023. "Enhancing Representation Learning on High-Dimensional, Small-Size Tabular Data: A Divide and Conquer Method with Ensembled VAEs." June 27, 2023. http://arxiv.org/abs/2306.15661.

Li, Chenghao, and Chaoning Zhang. 2023. "When ChatGPT for Computer Vision Will Come? From 2d to 3d." 2023. https://doi.org/10.48550/ARXIV.2305.06133.

Liu, Qi, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. 2018. "Constrained Graph Variational Autoencoders for Molecule Design." In *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/hash/b8a03c5c15fcfa8dae0b03351eb1742f-Abstract.html.

Martins, Miguel, Miguel Rocha, and Vítor Pereira. 2022. "Variational Autoencoders and Evolutionary Algorithms for Targeted Novel Enzyme Design." In *2022 IEEE Congress on Evolutionary Computation (CEC)*, 1–8. Padua, Italy: IEEE Press. https://doi.org/10.1109/CEC55065.2022.9870421.

Ramaswamy, Sridhar, Pablo Tamayo, Ryan Rifkin, Sayan Mukherjee, Chen-Hsiang Yeang, Michael Angelo, Christine Ladd, et al. 2001. "Multiclass Cancer Diagnosis Using Tumor Gene Expression Signatures." *Proceedings of the National Academy of Sciences* 98 (26): 15149–54. https://doi.org/10.1073/pnas.211566398.

Ramchandran, Siddharth, Gleb Tikhonov, Otto Lönnroth, Pekka Tiikkainen, and Harri Lähdesmäki. 2022. "Learning Conditional Variational Autoencoders with Missing Covariates." March 2, 2022. http://arxiv.org/abs/2203.01218.

Roberts, Adam, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. 2019. "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music." November 11, 2019. http://arxiv.org/abs/1803.05428.

Solorio-Fernández, Saúl, J. Ariel Carrasco-Ochoa, and José Fco. Martínez-Trinidad. 2020. "A Review of Unsupervised Feature Selection Methods." *Artificial Intelligence Review* 53 (2): 907–48. https://doi.org/10.1007/s10462-019-09682-y.

Torre, Jordi de la. 2023. "Autocodificadores Variacionales (VAE) Fundamentos Te\'oricos y Aplicaciones." February 18, 2023. https://doi.org/10.48550/arXiv.2302.09363.

Vahdat, Arash, and Jan Kautz. 2020. "NVAE: A Deep Hierarchical Variational Autoencoder." In *Advances in Neural Information Processing Systems*, 33:19667–79.

Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/hash/e3b21256183cf7c2c7a66be163579d37-Abstract.html.

Vie, Aymeric, Alissa M. Kleinnijenhuis, and Doyne J. Farmer. 2021. "Qualities, Challenges and Future of Genetic Algorithms: A Literature Review." September 13, 2021. http://arxiv.org/abs/2011.05277.

Vignolo, Leandro D., and Matias F. Gerard. 2017. "Evolutionary Local Improvement on Genetic Algorithms for Feature Selection." In *2017 XLIII Latin American Computer Conference (CLEI)*, 1–8. Cordoba: IEEE. https://doi.org/10.1109/CLEI.2017.8226467.

Wu, Zhangkai, Longbing Cao, and Lei Qi. 2023. "eVAE: Evolutionary Variational Autoencoder." January 1, 2023. https://doi.org/10.48550/arXiv.2301.00011.

Xiao, Han, Kashif Rasul, and Roland Vollgraf. 2017. "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms." September 15, 2017. https://doi.org/10.48550/arXiv.1708.07747.

Zagoruyko, Sergey, and Nikos Komodakis. 2017. "Wide Residual Networks." June 14, 2017. https://doi.org/10.48550/arXiv.1605.07146.

Zhang, Rui, Feiping Nie, Xuelong Li, and Xian Wei. 2019. "Feature Selection with Multi-View Data: A Survey." *Information Fusion* 50 (October): 158–67. https://doi.org/10.1016/j.inffus.2018.11.019.

Zhang, Yuchi, Yongliang Wang, Liping Zhang, Zhiqiang Zhang, and Kun Gai. 2019. "Improve Diverse Text Generation by Self Labeling Conditional Variational Auto Encoder." March 26, 2019. https://doi.org/10.48550/arXiv.1903.10842.

Apéndice A

Algoritmo Genético con la librería DEAP

```
import random
import numpy as np
from deap import base, creator, tools
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Parámetros del Algoritmo Genético
PROB_MUT = 0.1        # Probabilidad de mutación
PX = 0.75             # Probabilidad de cruce
GMAX = 15             # Número máximo de generaciones
POP_SIZE = 20         # Tamaño de la población

# Carga del conjunto de datos de ejemplo
data = load_breast_cancer()
X = data.data
y = data.target

# División del conjunto de datos en entrenamiento y prueba
Xtrain, Xtest, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.3,
    random_state=42
)

# Tamaños derivados
DAT_SIZE = Xtrain.shape[0]
IND_SIZE = Xtrain.shape[1]
PM = 1 / IND_SIZE    # Probabilidad de mutación por gen

# Definición de la función de fitness
def fitness(individual, Xtrain, Xtest, y_train, y_test):
    """Función de aptitud para evaluar la calidad de un individuo."""
```

```
    if not any(individual):
        return 0,  # Evita seleccionar individuos sin genes activos

    X_train = Xtrain[:, individual]
    X_test = Xtest[:, individual]

    model = MLPClassifier(hidden_layer_sizes=(5, 3),
        max_iter=1000,
        random_state=42
    )
    model.fit(X_train, y_train)

    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)

    # Minimización del número de características seleccionadas
    n_genes = np.sum(individual)
    alpha = 0.5  # Ponderación entre precisión y número de genes

    return alpha * accuracy + (1 - alpha) * (1 - n_genes / IND_SIZE),

# Configuración del entorno evolutivo
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

toolbox = base.Toolbox()
toolbox.register("attr_bool", lambda: random.random() < PM)
toolbox.register("individual", tools.initRepeat,
                    creator.Individual, toolbox.attr_bool, n=IND_SIZE)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=PROB_MUT)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", fitness,
                    Xtrain=Xtrain,
                    Xtest=Xtest,
                    y_train=y_train,
                    y_test=y_test)

# Función principal del Algoritmo Genético
def main():
    # Inicialización de la población
    population = toolbox.population(n=POP_SIZE)

    # Evaluación inicial
    fitnesses = list(map(toolbox.evaluate, population))
    for ind, fit in zip(population, fitnesses):
        ind.fitness.values = fit
```

```
    # Bucle evolutivo
    for gen in range(GMAX):
        # Selección y reproducción
        offspring = toolbox.select(population, len(population))
        offspring = list(map(toolbox.clone, offspring))

        # Aplicación del cruce y mutación
        for child1, child2 in zip(offspring[::2], offspring[1::2]):
            if random.random() < PX:
                toolbox.mate(child1, child2)
                del child1.fitness.values
                del child2.fitness.values

        for mutant in offspring:
            if random.random() < PROB_MUT:
                toolbox.mutate(mutant)
                del mutant.fitness.values

        # Evaluación de los nuevos individuos
        invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
        fitnesses = map(toolbox.evaluate, invalid_ind)
        for ind, fit in zip(invalid_ind, fitnesses):
            ind.fitness.values = fit

        # Reemplazo de la población
        population[:] = offspring

        # Recopilación de estadísticas
        fits = [ind.fitness.values[0] for ind in population]
        print(f"Gen:{gen + 1}-Mejor_fit:{max(fits):.4f}-Promedio:{np.mean(fits):.4f}")

    # Mejor individuo al finalizar
    best_ind = tools.selBest(population, 1)[0]
    print("\nMejor individuo encontrado: ", best_ind)
    print(f"Fitness: {best_ind.fitness.values[0]:.4f}")
    print(f"Número de características seleccionadas: {np.sum(best_ind)}")

if __name__ == "__main__":
    main()
```