TÍTULO

TÉCNICAS AVANZADAS DE SELECCIÓN DE ATRIBUTOS PARA CLASIFICACIÓN

ANÁLISIS Y ESTUDIO EMPÍRICO

AUTOR

Daniel Alejandro Ortiz Tandazo

Esta edición electrónica ha sido realizada en 2025

Tutor	Dr. Antonio Javier Tallón Ballesteros
Instituciones	Universidad Internacional de Andalucía; Universidad de Huelva
Curso	*Máster Universitario en Economía, Finanzas y Computación (2024/25)*
©	Daniel Alejandro Ortiz Tandazo
©	De esta edición: Universidad Internacional de Andalucía
Fecha documento	2025

Técnicas avanzadas de selección de atributos para clasificación: análisis y estudio empírico

By

Daniel Alejandro Ortiz Tandazo

A thesis submitted in conformity with the requirements
for the MSc in Economics, Finance and Computer Science

University of Huelva & International University of Andalusia

Julio 2025

Técnicas avanzadas de selección de atributos para clasificación: análisis y estudio empírico.

Autor

Daniel Alejandro Ortiz Tandazo

danielalejandro.ortiztandazo@estudiante.unia.es

Máster en Economía, Finanzas y Computación

Supervisor

Antonio Javier Tallón Ballesteros antonio.tallon@diesia.uhu.es

Universidad de Huelva y Universidad Internacional de Andalucía

2025

Abstract

This study explores and evaluates the effectiveness of advanced feature selection techniques in the context of machine learning, with a particular focus on classification problems. In the Big Data era, where vast amounts of information are generated, identifying and selecting the most relevant features is crucial to building efficient classification models and avoiding overfitting. This research focuses on five advanced techniques: Probability-based Particle Swarm Optimization (PSO), Golden Fish Search (GFS), Lasso, Ridge, and Elastic Net, evaluating their capacity to enhance model performance through dimensionality reduction. By applying these techniques to high-dimensional datasets, the impacts on accuracy, recall, F1-score, AUC-ROC, and execution time are analyzed. The findings help determine best practices for feature selection in complex environments and provide recommendations for their application in various classification domains, highlighting the ability of advanced techniques to optimize model efficiency and accuracy in scenarios where traditional methods prove ineffective.

Keywords: Feature Selection, Particle Swarm Optimization (PSO), Golden Fish Search (GFS), Lasso, Ridge Regression, Elastic Net, High-Dimensional Data, Classification, Model Performance, Dimensionality Reduction.

Resumen

El presente trabajo explora y evalúa la efectividad de técnicas avanzadas de selección de atributos en el ámbito del aprendizaje automático, con un enfoque particular en problemas de clasificación. En la era del Big Data, donde se generan grandes volúmenes de información, es crucial identificar y seleccionar los atributos más relevantes para construir modelos de clasificación eficientes y evitar el sobreajuste. Este estudio se centra en cinco técnicas avanzadas: Optimización por Enjambre de Partículas (PSO) basada en probabilidades, Golden Fish Search (GFS), Lasso, Ridge y Elastic Net, evaluando su capacidad para mejorar el rendimiento de los modelos de clasificación a través de la reducción de la dimensionalidad. A través de la aplicación de estas técnicas en conjuntos de datos de alta dimensionalidad, se analizan los impactos en accuracy, recall, F1-score, AUC-ROC y tiempo de ejecución. Los resultados permiten determinar las mejores prácticas para la selección de atributos en entornos complejos y ofrecen recomendaciones para su implementación en diferentes dominios de clasificación, destacando la capacidad de las técnicas avanzadas para optimizar la eficiencia y la exactitud de los modelos en escenarios donde los métodos tradicionales resultan ineficaces.

Palabras Clave: Selección de Atributos, Optimización por Enjambre de Partículas (PSO), Golden Fish Search (GFS), Lasso, Regresión Ridge, Elastic Net, Datos de Alta Dimensionalidad, Clasificación, Rendimiento del Modelo, Reducción de Dimensionalidad.

Agradecimientos

A mis padres, por ser mi inspiración y apoyo para poder cursar este máster.

A mis hermanos, por ser siempre motivarme a seguir adelante.

Y a mis gatos, que me acompañaron en cada momento de elaboración de este TFM.

Índice

1. Introducción 8
2. Objetivos 10
2.1. Objetivo General: 10
2.2. Objetivos Específicos: 10
3. Hipótesis 11
3.1. Hipótesis General: 11
3.2. Hipótesis Específicas: 11
4. Marco Teórico 12
4.1. Introducción a la Selección de Atributos 12
4.2. Técnicas Clásicas de Selección de Atributos 12
4.3. Métodos Embebidos 13
4.4. Técnicas Avanzadas de Selección de Atributos 15
4.5. Comparación de Técnicas: 16
4.6. Impacto de la Selección de Atributos en Modelos de Clasificación 17
4.7. Justificación del Uso de Técnicas Avanzadas Frente a Métodos Tradicionales 18
5. Metodología 18
5.1. Selección de Conjuntos de Datos 19
5.2. Preprocesamiento de los Datos 20
5.3. Técnicas de Selección de Atributos 21
5.4. Evaluación de Modelo de Clasificación 22
5.5. Comparación y Justificación del Uso de Técnicas Avanzadas 23
5.5.1. Justificación del Uso de Técnicas Avanzadas 24
6. Resultados 24
6.1 Resultados de la Selección de Atributos 24
6.1.1 Particle Swarm Optimization (PSO) 25
6.1.2 Golden Fish Search (GFS) 28
6.1.3 Lasso 31
6.1.4 Ridge 33
6.1.5 Elastic Net 35
6.1.6. Comparación General de las Técnicas de Selección de Atributos 36
6.2 Resultados de la clasificación 39
6.2.1 Resultados con SVM tras Selección con Lasso 40
6.2.2 Resultados con SVM tras Selección con Ridge 41
6.2.3 Resultados con SVM tras Selección con Elastic Net 42
6.2.4 Resultados con SVM tras Selección con GFS 44

6.2.5 Resultados con SVM tras Selección con PSO 46
6.2.6 Resultados de SVM de las Bases de Datos sin Técnicas de Selección de Atributos 49
6.3. Comparativa de clasificación por SVM entre Bases de datos 50
6.3.1 Divorce Predictors Data Set: 50
6.3.2 Breast Cancer Wisconsin (Diagnostic): 51
6.3.3 Ionosphere: 53
6.3.4 Predict Students' Dropout and Academic Success: 54
6.3.5 Hepatitis C Virus for Egyptian patients (HCV): 55
6.3.6 Molecular Biology (Splice-junction Gene Sequences): 57
6.3.7 Parkinson's Disease Classification: 58
6.3.8 Statlog (German Credit Data): 59
6.3.9 Musk (Version 1): 61
6.3.10 Mice Protein Expression: 62
7. Conclusiones 64
8. Recomendaciones 66
9. Referencias 68
10. Anexos 69
10.1. Código en Python 69

1. Introducción

En la era del Big Data, la cantidad de datos disponibles ha crecido exponencialmente, presentando desafíos y oportunidades para el análisis y la toma de decisiones. Aunque este incremento de datos ofrece una gran cantidad de información potencialmente útil, también incluye datos irrelevantes o redundantes que pueden complicar la construcción de modelos predictivos eficaces (Liu & Motoda, 1998). Aquí es donde la selección de atributos adquiere un papel fundamental, ya que permite identificar y seleccionar las características más relevantes para mejorar el rendimiento de los modelos de clasificación (Guyon & Elisseeff, 2003).

La selección de atributos es un proceso clave en el preprocesamiento de datos, que tiene como objetivo reducir la dimensionalidad de los conjuntos de datos. La eliminación de atributos irrelevantes o redundantes ayuda a simplificar el modelo y mejora la eficiencia computacional, lo que es especialmente importante en entornos donde el volumen de datos es masivo (Chandrashekar & Sahin, 2014). Además, una selección adecuada de atributos puede aumentar la precisión de los modelos de clasificación al reducir el ruido en los datos, evitando el sobreajuste y mejorando la capacidad de generalización (Hastie, Tibshirani, & Friedman, 2009).

Impacto de la Selección de Atributos en el Rendimiento de los Modelos
La selección de atributos no solo mejora la precisión y eficiencia de los modelos, sino que también facilita su interpretabilidad. En campos como la medicina o la bioinformática, donde la interpretabilidad es fundamental para la toma de decisiones, la selección de atributos permite a los expertos identificar cuáles son los factores clave que contribuyen a las predicciones (Guyon & Elisseeff, 2003; Hastie, Tibshirani, & Friedman, 2009). Por ejemplo, en el diagnóstico médico, identificar los biomarcadores relevantes a partir de grandes conjuntos de datos puede mejorar el proceso de diagnóstico y pronóstico (Liu & Motoda, 1998).

El impacto de la selección de atributos en el rendimiento de los modelos es evidente en múltiples dominios. Al reducir la dimensionalidad, se disminuyen los recursos computacionales necesarios para entrenar y ejecutar los modelos, lo que a su vez reduce los tiempos de ejecución. En el contexto del Big Data, donde los tiempos de procesamiento son críticos, esta reducción en el tiempo de entrenamiento es particularmente valiosa (Jović, Brkić, & Bogunović, 2015).

Técnicas Tradicionales y Avanzadas de Selección de Atributos

A lo largo de los años, se han desarrollado diversas técnicas para abordar el problema de la selección de atributos. Los métodos tradicionales, como los métodos de filtrado, evaluaban los atributos de manera independiente utilizando medidas estadísticas como la información mutua o la correlación (Liu & Motoda, 1998). Sin embargo, estos métodos no consideran las interacciones entre atributos, lo que puede resultar en la selección de conjuntos subóptimos de características (Guyon & Elisseeff, 2003).

Por otro lado, los métodos de envoltura y los métodos basados en la incrustación han demostrado ser más efectivos al considerar las interacciones entre los atributos y el modelo de clasificación. Los métodos de envoltura, como la eliminación recursiva de características (RFE), utilizan el modelo predictivo para evaluar la calidad de diferentes subconjuntos de atributos, lo que los hace más precisos, aunque también más costosos en términos computacionales (Chandrashekar & Sahin, 2014). Los métodos basados en la incrustación, como la regresión Lasso, integran la selección de atributos dentro del proceso de modelado, lo que reduce la complejidad computacional y selecciona automáticamente los atributos más relevantes (Zou & Hastie, 2005).

Técnicas Avanzadas de Optimización para la Selección de Atributos

En años recientes, se han introducido técnicas avanzadas de optimización que aplican algoritmos inspirados en la naturaleza, como la Optimización por Enjambre de Partículas (PSO), para resolver el problema de la selección de atributos. PSO, inspirado en el comportamiento social de las bandadas de aves, ha demostrado ser efectivo para explorar grandes espacios de búsqueda y

encontrar subconjuntos óptimos de atributos (Pan, Wu, & Qian, 2019). Una versión basada en probabilidades de PSO ha mostrado mejoras significativas en la precisión y la eficiencia en comparación con los métodos tradicionales (Pan, Wu, & Qian, 2019).

Otra técnica emergente es el algoritmo de búsqueda del pez dorado (Golden Fish Search, GFS), una metaheurística que ha mostrado potencial para evitar los mínimos locales y explorar eficazmente grandes espacios de soluciones en el contexto de la selección de atributos (Dorigo & Birattari, 2010). Estas técnicas avanzadas, al igual que Lasso y Elastic Net, son especialmente útiles cuando se trabaja con conjuntos de datos de alta dimensionalidad, como aquellos que se encuentran comúnmente en aplicaciones de Big Data (Zou & Hastie, 2005).

2. Objetivos

2.1. Objetivo General:

Evaluar la efectividad de diferentes técnicas avanzadas de selección de atributos en la mejora del rendimiento de modelos de clasificación, en términos de precisión, eficiencia y reducción de dimensionalidad, aplicando dichas técnicas en conjuntos de datos diversos para determinar cuál ofrece mejores resultados en diferentes contextos.

2.2. Objetivos Específicos:

1. Comparar diferentes técnicas de selección de atributos (por ejemplo, PSO basado en probabilidades, Golden Fish Search, Lasso, Ridge y Elastic Net) en cuanto a su capacidad para mejorar el rendimiento de los modelos de clasificación.

2. Implementar y analizar la eficiencia de las técnicas de selección de atributos en términos de reducción de dimensionalidad y tiempo de ejecución en conjuntos de datos de alta dimensionalidad.

3. Evaluar el impacto de la selección de atributos en la precisión de los modelos de clasificación utilizando métricas estándar como la precisión, el recall, el F1-score y el AUC-ROC.

4. Determinar los criterios para elegir una técnica de selección de atributos específica, dependiendo de las características del conjunto de datos, como la cantidad de atributos, la correlación entre ellos, y la naturaleza de los datos (estructurados o no estructurados).

5. Desarrollar recomendaciones prácticas sobre cuándo utilizar cada técnica de selección de atributos en función de los resultados obtenidos.

3. Hipótesis

3.1. Hipótesis General:

La implementación de técnicas avanzadas de selección de atributos mejora significativamente el rendimiento de los modelos de clasificación al reducir la dimensionalidad, mejorar la precisión, y aumentar la eficiencia en la ejecución de los algoritmos, en comparación con la utilización de todos los atributos sin ningún tipo de filtrado.

3.2. Hipótesis Específicas:

1. Las técnicas de selección de atributos basadas en optimización metaheurística (como PSO o Golden Fish Search) son más eficientes en la exploración del espacio de soluciones, proporcionando un mejor rendimiento en modelos de clasificación con datos de alta dimensionalidad que las técnicas clásicas como Lasso y Ridge.

2. El uso de Elastic Net, que combina las penalizaciones L1 y L2, es más efectivo que Lasso y Ridge por separado en conjuntos de datos con alta correlación entre atributos, mejorando la precisión y reduciendo el sobreajuste.

3. Los modelos de clasificación que implementan técnicas de selección de atributos muestran mejoras significativas en la precisión y la interpretabilidad en comparación con aquellos que utilizan todos los atributos disponibles sin filtrado.

4. La reducción de la dimensionalidad mediante técnicas de selección de atributos reduce el tiempo de entrenamiento de los modelos, sin afectar negativamente su precisión.

4. Marco Teórico

4.1. Introducción a la Selección de Atributos

La selección de atributos, también conocida como selección de características, es un proceso fundamental en el preprocesamiento de datos para tareas de clasificación en el aprendizaje automático. Este proceso implica identificar y seleccionar un subconjunto de los atributos más relevantes con el fin de construir un modelo predictivo eficiente, mejorando la precisión, reduciendo el tiempo de entrenamiento y evitando el sobreajuste (Guyon & Elisseeff, 2003). En entornos de Big Data, la presencia de atributos irrelevantes o redundantes puede deteriorar el rendimiento de los modelos, aumentando la complejidad computacional (Liu & Motoda, 1998).

4.2. Técnicas Clásicas de Selección de Atributos

Aunque en este estudio no se utilizarán métodos clásicos de selección de atributos, su discusión es relevante para contextualizar el desarrollo de las técnicas avanzadas.

A. Métodos de Filtrado (Filter Methods)

Los métodos de filtrado evalúan cada atributo de manera independiente y se basan en medidas estadísticas como la información mutua, la correlación, o la prueba chi-cuadrado (Liu & Motoda, 1998). Estos métodos son rápidos y fáciles de implementar, pero no consideran las interacciones entre los atributos, lo que puede limitar su efectividad en problemas complejos (Guyon & Elisseeff, 2003).
Ecuación de Información Mutua:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

Esta ecuación mide la dependencia entre el atributo X y la variable objetivo Y, lo que es útil para seleccionar atributos con una relación directa con la clase objetivo.

Ecuación de chi-cuadrado:

$$x^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

El test chi-cuadrado es especialmente útil para atributos categóricos, ya que mide la dependencia entre los atributos y la variable objetivo (Guyon & Elisseeff, 2003).

B. Métodos de Envoltura (Wrapper Methods)

A diferencia de los métodos de filtrado, los métodos de envoltura evalúan subconjuntos de atributos utilizando un modelo de clasificación para determinar su calidad. Este enfoque suele ofrecer mejores resultados, pero es más costoso en términos computacionales, ya que se deben entrenar modelos repetidamente para cada subconjunto evaluado (Jović, Brkić, & Bogunović, 2015). Un ejemplo popular es la Eliminación Recursiva de Características (RFE), que selecciona los atributos eliminando los menos importantes en cada iteración (Chandrashekar & Sahin, 2014).

4.3. Métodos Embebidos

Las técnicas embebidas seleccionan los atributos durante el proceso de entrenamiento del modelo mediante la aplicación de regularización, que controla el tamaño de los coeficientes de los atributos. Las tres técnicas más utilizadas en este enfoque son Lasso, Ridge y Elastic Net.

1. Lasso (Least Absolute Shrinkage and Selection Operator)

Lasso utiliza una penalización L_1, que fuerza algunos coeficientes a ser exactamente cero, lo que efectivamente elimina los atributos irrelevantes del modelo. Es especialmente útil cuando solo se espera que un pequeño subconjunto de los atributos sea relevante (Tibshirani, 1996).

La capacidad de Lasso para seleccionar automáticamente un subconjunto de atributos relevantes lo ha convertido en una técnica popular en problemas de alta dimensionalidad (Zou & Hastie, 2005).

Ecuación de Lasso:

$$\min_{\beta}\left(\frac{1}{2n}\sum_{i=1}^{n}(y_i-\hat{y_i})^2+\lambda\sum_{j=1}^{p}|\beta_j|\right)$$

El término de penalización L_1 $(\lambda\sum_{j=1}^{p}|\beta_j|)$ reduce algunos coeficientes β_j a cero, eliminando atributos.

2. Ridge Regression

Ridge es otra técnica de regularización que penaliza los coeficientes, pero en lugar de forzar algunos a cero, reduce todos los coeficientes de manera continua. Ridge utiliza una penalización $\boldsymbol{L_2}$, lo que evita que los coeficientes crezcan demasiado, pero sin eliminar por completo los atributos (Hoerl & Kennard, 1970). Esta técnica es útil cuando se espera que todos los atributos tengan alguna relevancia, aunque su capacidad para reducir los coeficientes sin eliminarlos puede no ser suficiente para problemas de alta dimensionalidad, donde muchos atributos son irrelevantes (Hastie, Tibshirani, & Friedman, 2009).

Ecuación de Ridge:

$$\boldsymbol{\min_{\beta}\left(\frac{1}{2n}\sum_{i=1}^{n}(y_i-\hat{y_i})^2+\lambda\sum_{j=1}^{p}\beta_j^2\right)}$$

El término de penalización $\boldsymbol{L_2}$ $\boldsymbol{(\lambda\sum_{j=1}^{p}\beta_j^2)}$ reduce los coeficientes sin eliminarlos por completo.

3. Elastic Net

Elastic Net combina las penalizaciones $\boldsymbol{L_1}$ y $\boldsymbol{L_2}$, lo que le permite manejar datos donde los atributos están altamente correlacionados. Este modelo equilibra las ventajas de Lasso y Ridge, aplicando ambas penalizaciones de manera

simultánea. Es particularmente útil en situaciones en las que hay múltiples atributos correlacionados que deben ser seleccionados o descartados conjuntamente (Zou & Hastie, 2005).

Esta técnica ha sido ampliamente adoptada en áreas como la genética y el análisis de datos de alto rendimiento, debido a su capacidad para manejar datos con una estructura correlacional compleja (Zou & Hastie, 2005).
Ecuación de Elastic Net:

$$\min_{\beta}\left(\frac{1}{2n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2+\lambda_1\sum_{j=1}^{p}|\beta_j|+\lambda_2\sum_{j=1}^{p}\beta_j^2\right)$$

Elastic Net es útil cuando se requiere combinar ambas penalizaciones para seleccionar de manera efectiva los atributos más relevantes, especialmente en problemas donde los atributos están correlacionados (Zou & Hastie, 2005).

4.4. Técnicas Avanzadas de Selección de Atributos

A. PSO Basado en Probabilidades

La Optimización por Enjambre de Partículas (PSO) es una técnica de optimización inspirada en el comportamiento de grupos sociales como bandadas de aves o bancos de peces. En su aplicación a la selección de atributos, PSO se ha demostrado efectivo en la exploración de grandes espacios de búsqueda y en la identificación de subconjuntos óptimos de atributos (Pan, Wu, & Qian, 2019). La versión basada en probabilidades de PSO, propuesta por Xiaoying Pan, introduce un enfoque probabilístico que ha mostrado mejoras en la precisión de la clasificación comparado con métodos tradicionales (Pan, Wu, & Qian, 2019).

Ecuaciones de PSO:

$$\begin{aligned}\boldsymbol{v_i(t+1)} &= \boldsymbol{wv_i(t)} + \boldsymbol{c_1}r_1(\boldsymbol{p_i}-\boldsymbol{x_i(t)}) + \boldsymbol{c_2 r_2}\big(\boldsymbol{g}-\boldsymbol{x_i(t)}\big)\boldsymbol{x_i(t+1)}\\ &= \boldsymbol{x_i(t)} + \boldsymbol{v_i(t+1)}\end{aligned}$$

Donde:

- $\boldsymbol{v_i(t)}$ es la velocidad de la partícula $\boldsymbol{i}$ en el tiempo $\boldsymbol{t}$,

- $\boldsymbol{x_i(t)}$ es la posición de la partícula,
- $\boldsymbol{p_i}$ es la mejor posición personal de la partícula,
- $\boldsymbol{g}$ es la mejor posición global,
- $\boldsymbol{w}$ es el factor de inercia que controla la exploración,
- $\boldsymbol{c_1}$ y $\boldsymbol{c_2}$ son coeficientes de aceleración que ponderan el efecto de la mejor posición personal y global (Pan, Wu, & Qian, 2019).

B. Golden Fish Search (GFS)

El Golden Fish Search (GFS) es una metaheurística emergente que se basa en el comportamiento de los peces dorados. Aunque es menos conocida que otras técnicas, ha mostrado promesas en la selección de atributos debido a su capacidad para evitar estancarse en mínimos locales y explorar eficientemente el espacio de soluciones (Dorigo & Birattari, 2010).

Condiciones de Optimización en GFS:

En GFS, las soluciones son representadas como posiciones de "peces dorados" en un espacio de búsqueda. La exploración del espacio se optimiza mediante la actualización iterativa de las posiciones de los peces, utilizando mecanismos basados en la distancia entre las soluciones y su grado de aptitud. A diferencia de otros algoritmos metaheurísticos, GFS utiliza un enfoque dinámico para la diversificación y la intensificación, lo que le permite escapar de mínimos locales.

4.5. Comparación de Técnicas:

A. PSO vs. GFS

La técnica de PSO es reconocida por su flexibilidad y su capacidad para abordar problemas de alta dimensionalidad con menos conocimiento previo sobre el espacio de búsqueda. PSO es ideal cuando se requiere una exploración rápida y efectiva del espacio de soluciones. Sin embargo, en escenarios donde es probable que las soluciones caigan en mínimos locales, GFS puede ser una mejor opción debido a su capacidad para mantener la diversidad en las soluciones y evitar el estancamiento (Pan, Wu, & Qian, 2019; Dorigo & Birattari, 2010).

B. Lasso, Ridge, y Elastic Net

La elección entre Lasso, Ridge, y Elastic Net depende del tipo de datos y la correlación entre los atributos:

- Lasso es preferible cuando solo se espera que unos pocos atributos sean relevantes, ya que la penalización $\boldsymbol{L_1}$ fuerza algunos coeficientes a ser exactamente cero, eliminando atributos no relevantes (Tibshirani, 1996).
- Ridge es más adecuado cuando se espera que todos los atributos tengan algún grado de relevancia, ya que la penalización $\boldsymbol{L_2}$ reduce los coeficientes sin eliminarlos por completo (Hoerl & Kennard, 1970).
- Elastic Net es ideal cuando los atributos están altamente correlacionados, ya que combina las ventajas de Lasso y Ridge, penalizando tanto $\boldsymbol{L_1}$ como$\boldsymbol{L_2}$, lo que permite seleccionar grupos de atributos correlacionados (Zou & Hastie, 2005).

4.6. Impacto de la Selección de Atributos en Modelos de Clasificación

A. Mejora del Rendimiento del Modelo

La selección de atributos adecuada puede tener un impacto significativo en el rendimiento de los modelos de clasificación. Al eliminar atributos redundantes o irrelevantes, se reduce la dimensionalidad del conjunto de datos, lo que ayuda a evitar el sobreajuste y mejora la capacidad de generalización del modelo. Estudios han demostrado que una reducción adecuada en la cantidad de atributos seleccionados no solo mejora la precisión del modelo, sino que también incrementa su robustez en escenarios de validación cruzada (Guyon & Elisseeff, 2003; Chandrashekar & Sahin, 2014).

B. Reducción del Tiempo de Ejecución

Otro impacto positivo de la selección de atributos es la reducción en el tiempo de ejecución de los modelos. Un conjunto de datos con menor dimensionalidad requiere menos recursos computacionales, lo que reduce el tiempo de entrenamiento y predicción. Esto es especialmente importante en aplicaciones de Big Data, donde el procesamiento eficiente de grandes volúmenes de datos es crítico (Liu & Motoda, 1998; Jović, Brkić, & Bogunović, 2015).

C. Mejora en la Interpretabilidad del Modelo

En ciertos campos, como la medicina y la bioinformática, la interpretabilidad del modelo es tan importante como su precisión. La selección de atributos permite a los expertos humanos identificar qué características específicas están contribuyendo a las predicciones del modelo, lo que facilita su comprensión y aceptación en aplicaciones sensibles, como el diagnóstico clínico (Hastie, Tibshirani, & Friedman, 2009).

D. Impacto en la Robustez del Modelo

La eliminación de atributos irrelevantes no solo reduce el sobreajuste, sino que también incrementa la robustez del modelo ante variaciones en los datos de entrenamiento. Los modelos con un menor número de atributos seleccionados tienden a ser más estables y presentan una menor varianza en su rendimiento cuando se utilizan diferentes particiones de los datos para la validación (Guyon & Elisseeff, 2003).

4.7. Justificación del Uso de Técnicas Avanzadas Frente a Métodos Tradicionales

Aunque los métodos clásicos de filtrado y envoltura se han utilizado ampliamente en la selección de atributos, sus limitaciones hacen que las técnicas avanzadas, como PSO, GFS, Lasso, Ridge y Elastic Net, sean más apropiadas para escenarios de alta dimensionalidad y conjuntos de datos con atributos correlacionados. Los métodos tradicionales de filtrado no consideran las interacciones entre los atributos, lo que puede llevar a la selección de subconjuntos subóptimos. Por otro lado, los métodos de envoltura, aunque más precisos, son computacionalmente costosos y no escalables para grandes conjuntos de datos (Guyon & Elisseeff, 2003; Chandrashekar & Sahin, 2014).

5. Metodología

En esta sección se describe el enfoque metodológico que se empleará para evaluar la efectividad de las técnicas avanzadas de selección de atributos. El objetivo principal es comparar el rendimiento de cinco técnicas de selección de

atributos: PSO basado en probabilidades, Golden Fish Search (GFS), Lasso, Ridge y Elastic Net, dentro de modelos de clasificación en un conjunto de datos de alta dimensionalidad.

5.1. Selección de Conjuntos de Datos

Los conjuntos de datos empleados en este estudio provendrán del *UCI Machine Learning Repository*, una fuente reconocida por su diversidad y accesibilidad. Se seleccionarán bases de datos con un número suficiente de atributos (alta dimensionalidad) y una cantidad representativa de instancias, lo que permitirá aplicar y evaluar técnicas de selección de atributos y, posteriormente, algoritmos de clasificación.

En total, se emplearán diez conjuntos de datos. Estos han sido elegidos por su diversidad en cuanto a número de atributos, tamaño de muestra y tipo de problema (clasificación binaria, multiclase o clustering), Esta variedad permitirá validar la robustez y versatilidad de las técnicas analizadas en distintos contextos. La siguiente tabla resume sus características principales:

Base de datos	**Instancias**	**Atributos**	**Tipo de problema**
Breast Cancer Wisconsin Diagnostic (WDBC)	569	30	Clasificación binaria
Divorce Predictors data set	170	54	Clasificación binaria
Hepatitis C Virus for Egyptian patients (HCV)	615	28	Clasificación multiclase
Ionosphere	351	34	Clasificación binaria
Mice Protein Expression	1080	80	Clasificación multiclase
Musk (v1)	476	168	Clasificación binaria
Parkinson's	756	753	Clasificación binaria
Splice	3190	60	Clasificación multiclase
Statlog (German)	1000	20	Clasificación binaria
Student Dropout	4424	36	Clasificación multiclase

Estos conjuntos serán utilizados tanto para la evaluación de distintas técnicas de selección de atributos como para el análisis del rendimiento en tareas de clasificación supervisada.

5.2. Preprocesamiento de los Datos

El preprocesamiento de los datos constituye un paso fundamental antes de aplicar las técnicas de selección de atributos. Este proceso permite asegurar que los conjuntos de datos estén en un formato adecuado para el análisis, reduciendo posibles ruidos, inconsistencias y sesgos que podrían afectar el rendimiento de los modelos.

A. Limpieza de Datos

Los datos incompletos, inconsistentes o duplicados serán eliminados o imputados. Se emplearán técnicas como la imputación de valores faltantes utilizando la media, mediana o métodos más avanzados como KNN para datos numéricos (Wen et al., 2018). En el caso de atributos categóricos, se utilizarán técnicas de codificación como one-hot encoding para convertir los valores categóricos en variables binarias (Chandrashekar & Sahin, 2014).

B. Normalización y Estandarización

Dado que algunas de las técnicas seleccionadas, como Lasso, Ridge y Elastic Net, son sensibles a la escala de los atributos, se aplicarán técnicas de normalización o estandarización a los datos numéricos. Se utilizará la normalización min-max para ajustar los valores entre 0 y 1, o la estandarización (Z-score) para transformar los datos a una distribución con media 0 y desviación estándar 1 (Hastie, Tibshirani, & Friedman, 2009).

C. Detección de Atributos Correlacionados

Antes de aplicar las técnicas avanzadas de selección, se realizará un análisis de correlación entre los atributos mediante la matriz de correlación de Pearson. Si se encuentran atributos altamente correlacionados ($\boldsymbol{r > 0.9}$), se considerará la

eliminación de algunos de ellos, ya que podrían introducir redundancia en el modelo (Guyon & Elisseeff, 2003).

5.3. Técnicas de Selección de Atributos

Se utilizarán las siguientes técnicas avanzadas de selección de atributos:

- PSO Basado en Probabilidades (Propuesta de Xiaoying Pan): Esta técnica se basa en la optimización por enjambre de partículas, donde las partículas representan subconjuntos de atributos. A través de un enfoque probabilístico, se ajustarán los parámetros de las partículas para explorar de manera eficiente el espacio de búsqueda (Pan, Wu, & Qian, 2019).

- Golden Fish Search (GFS): Esta técnica metaheurística explora el espacio de búsqueda basándose en el comportamiento de los peces dorados. Su capacidad para evitar caer en mínimos locales la hace adecuada para conjuntos de datos de alta dimensionalidad (Dorigo & Birattari, 2010).

- Lasso: Utilizando la penalización $\boldsymbol{L_1}$, Lasso se aplicará para seleccionar un subconjunto de atributos relevantes. Se implementará en Python utilizando sklearn y se ajustará el parámetro de regularización mediante validación cruzada (Tibshirani, 1996).

- Ridge: Ridge utilizará la penalización $\boldsymbol{L_2}$ y se ajustará para minimizar el error cuadrático medio (MSE) utilizando el paquete sklearn (Hoerl & Kennard, 1970).

- Elastic Net: Combinando $\boldsymbol{L_1}$ y $\boldsymbol{L_2}$, Elastic Net será implementado utilizando sklearn, y se ajustará mediante validación cruzada para equilibrar las penalizaciones (Zou & Hastie, 2005).

5.4. Evaluación de Modelo de Clasificación

Para evaluar el rendimiento del modelo de clasificación, se empleará el clasificador máquina de soporte vectorial (SVM) sobre los subconjuntos de atributos seleccionados por cada técnica de reducción.

Con el fin de garantizar la robustez de los resultados y evitar el sobreajuste, se utilizará un esquema de validación cruzada estratificada de 10 pliegues. Esto significará que cada conjunto de datos se dividirá en 10 partes, entrenando el modelo con 9 partes y validando con la restante, repitiendo el proceso 10 veces y promediando los resultados.

Este enfoque permitirá una evaluación más fiable que una simple división en entrenamiento y prueba, especialmente en conjuntos de datos con tamaño moderado o desigual distribución de clases (Hastie, Tibshirani, & Friedman, 2009).

Las métricas que **se utilizarán** para evaluar la efectividad de las técnicas de selección de atributos mediante el rendimiento del modelo SVM serán:

1) Accuracy (precisión):

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{\boldsymbol{TP} + \boldsymbol{TN} + \boldsymbol{FP} + \boldsymbol{FN}}$$

   Es el porcentaje de predicciones correctas en comparación con todas las predicciones realizadas.

2) Recall (Sensibilidad o Tasa de Verdaderos Positivos):

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}}$$

   Es la capacidad del modelo para identificar correctamente las instancias positivas.

3) F1-Score:

$$\boldsymbol{F1} = \boldsymbol{2} * \frac{\boldsymbol{Accuracy} * \boldsymbol{Recall}}{\boldsymbol{Accuracy} + \boldsymbol{Recall}}$$

   Es el promedio ponderado entre Accuracy y Recall

4) AUC-ROC (Área Bajo la Curva ROC):

Área bajo la curva ROC, que mide la capacidad del modelo para discriminar entre clases.

5) Tiempo de Ejecución:
   Tiempo total de procesamiento desde la selección de atributos hasta la clasificación.

   Definición de TP, TN, FP y FN:

   - TP (True Positives): Instancias correctamente clasificadas como positivas.
   - TN (True Negatives): Instancias correctamente clasificadas como negativas.
   - FP (False Positives): Instancias incorrectamente clasificadas como positivas (falsos positivos).
   - FN (False Negatives): Instancias incorrectamente clasificadas como negativas (falsos negativos) (Guyon & Elisseeff, 2003).

5.5. Comparación y Justificación del Uso de Técnicas Avanzadas

Finalmente, se compararán los resultados obtenidos con cada técnica, destacando:

- Precisión: Se espera que PSO y GFS, al ser técnicas metaheurísticas, exploren de manera más efectiva el espacio de búsqueda, mejorando la precisión del modelo comparado con las técnicas de regularización.
- Reducción de la Dimensionalidad: Lasso y Elastic Net son más efectivos para reducir el número de atributos debido a su capacidad de eliminar coeficientes irrelevantes.
- Escalabilidad: Las técnicas de regularización son más rápidas en conjuntos de datos medianos, pero PSO y GFS son más adecuados para problemas de mayor dimensionalidad debido a su capacidad para evitar el sobreajuste (Chandrashekar & Sahin, 2014).

5.5.1. Justificación del Uso de Técnicas Avanzadas

El uso de técnicas avanzadas como PSO, GFS, Lasso, Ridge y Elastic Net está justificado debido a la alta dimensionalidad y correlación entre atributos en los conjuntos de datos seleccionados. Los métodos clásicos de filtrado y envoltura, aunque útiles en escenarios simples, no son efectivos para conjuntos de datos complejos debido a que:

- Los métodos de filtrado no consideran las interacciones entre los atributos, lo que puede llevar a una selección subóptima de características.
- Los métodos de envoltura, aunque más precisos, son costosos computacionalmente y no escalables para grandes volúmenes de datos (Guyon & Elisseeff, 2003; Chandrashekar & Sahin, 2014).

6. Resultados

En esta sección se presentan los resultados obtenidos tras aplicar las cinco técnicas avanzadas de selección de atributos expuestas anteriormente.

Posteriormente, se entrenaron modelos de clasificación utilizando únicamente las características seleccionadas por cada técnica, y se evaluaron mediante métricas de rendimiento como la precisión, recall, F1-score, AUC-ROC y tiempo de ejecución.

6.1 Resultados de la Selección de Atributos

En este apartado se presentan los resultados obtenidos tras aplicar cinco técnicas avanzadas de selección de atributos: Particle Swarm Optimization (PSO), Golden Fish Search (GFS), Lasso, Ridge y Elastic Net. Estas técnicas se seleccionaron por su capacidad para identificar subconjuntos relevantes de atributos en problemas de clasificación, especialmente en contextos de alta dimensionalidad.

Cada técnica fue aplicada a las mismas diez bases de datos previamente descritas, con el objetivo de comparar su desempeño en términos de número de atributos seleccionados, tiempo de ejecución, y otras métricas específicas según el método, como el índice de silueta en el caso de PSO y GFS.

Los resultados obtenidos permiten observar patrones de comportamiento distintos entre técnicas, facilitando un análisis comparativo de su eficacia y eficiencia. En los siguientes subapartados se detalla el rendimiento de cada una de estas técnicas en las bases de datos utilizadas.

6.1.1 Particle Swarm Optimization (PSO)

La técnica de Particle Swarm Optimization (PSO) fue utilizada con el objetivo de seleccionar subconjuntos óptimos de atributos mediante la optimización del índice de silueta, aplicando un algoritmo de clustering (K-Means) con tres clústeres. Se empleó un enfoque iterativo que selecciona características en distintas rondas, hasta que no se produce mejora significativa en la métrica objetivo.

Esta técnica fue aplicada a cada una de las diez bases de datos consideradas. Para cada ejecución se registró el número total de atributos seleccionados, el valor máximo del índice de silueta alcanzado y el tiempo de ejecución en segundos.

A continuación, se muestra una tabla resumen con los resultados obtenidos:

Base de Datos	N° atributos seleccionados	N° atributos originales	% Reducción	Índice de silueta	Tiempo (s)
Divorce Predictors Data Set	5	29	82,76%	0,6431	53,1255
Breast Cancer Wisconsin (Diagnostic)	9	20	55,00%	0,6229	174,4323
Ionosphere	9	34	73,53%	0,4411	99,9098
Predict Students' Dropout and Academic Success	14	32	56,25%	0,6073	1339,9404
Hepatitis C Virus (HCV) for Egyptian Patients	13	28	53,57%	0,2656	506,2993
Molecular Biology (Splice-Junction Gene Sequences)	29	60	51,67%	0,0669	1262,2398
Parkinson's Disease Classification	202	389	48,07%	0,0977	284,1788
Statlog (German Credit Data)	8	20	60,00%	0,959	225,4538
Musk (Version 1)	38	98	61,22%	0,2909	120,3501
Mice Protein Expression	36	73	50,68%	0,192	348,5102

Análisis de los resultados:

La técnica de selección de atributos basada en PSO (Particle Swarm Optimization) ha mostrado un comportamiento consistente con su naturaleza estocástica y exploratoria, logrando reducciones significativas de atributos en todos los conjuntos de datos. A diferencia de las técnicas deterministas como Ridge o Elastic Net, PSO aplica una estrategia metaheurística que busca optimizar el índice de silueta en contextos no supervisados, lo que lo hace útil incluso sin recurrir directamente a la variable objetivo.

En bases de datos como Divorce, Ionosphere y Musk, PSO logró reducir entre un 61 % y un 83 % de los atributos, indicando una alta capacidad de filtrado de variables redundantes o poco informativas. En particular, el caso de Divorce, con una selección de solo 5 atributos de los 29 disponibles, muestra cómo el algoritmo pudo identificar un conjunto muy reducido de variables relevantes manteniendo una estructura de clustering sólida (índice de silueta = 0,6431).

Breast Cancer, StudentDrop, HCV y Mice también presentaron reducciones superiores al 50 %, lo cual es destacable considerando que estas bases varían ampliamente en complejidad, número de clases y escala. Por ejemplo, en StudentDrop, la selección de 14 atributos (de 32) se dio junto a un índice de silueta aceptable (0,6073), aunque con un coste computacional elevado (más de 1300 segundos), reflejando la naturaleza intensiva del método.

Los resultados de Splice y Parkinson's son más conservadores en términos de reducción (alrededor del 48 %–52 %), pero confirman que incluso en bases de datos con alta dimensionalidad (como Parkinson's, con 389 atributos), PSO mantiene su capacidad de filtrado sin caer en reducciones excesivas o ineficaces. No obstante, en ambos casos, el índice de silueta fue bajo (0,0669 en Splice y 0,0977 en Parkinson's), lo que sugiere que la estructura de los datos no es fácilmente agrupable mediante clustering, independientemente de la selección de atributos.

Un aspecto importante que destacar es el tiempo de ejecución: PSO fue considerablemente más lento que los métodos deterministas. En bases como StudentDrop y Splice, los tiempos superaron los 1200 segundos, evidenciando el alto coste computacional que implica una exploración metaheurística completa del espacio de búsqueda, especialmente cuando se optimiza un criterio como el índice de silueta, que requiere cálculos repetidos de clustering.
Finalmente, bases como Statlog y Mice presentan un buen equilibrio: reducciones superiores al 50 % con tiempos más razonables dentro de la escala de PSO (225 y 348 segundos respectivamente), y con un índice de silueta particularmente alto en Statlog (0,9590), lo que evidencia la eficacia del método cuando la estructura interna de los datos es bien capturada por el agrupamiento.

6.1.2 Golden Fish Search (GFS)

Base de Datos	N° atributos seleccionados	N° atributos originales	% Reducción	Índice de silueta	Tiempo (s)
Divorce Predictors Data Set	12	29	58,62%	0,5448	2,2163
Breast Cancer Wisconsin (Diagnostic)	5	20	75,00%	0,3267	7,0400
Ionosphere	16	34	52,94%	0,3445	3,8975
Predict Students' Dropout and Academic Success	13	32	59,38%	0,4111	55,2146
Hepatitis C Virus (HCV) for Egyptian Patients	8	28	71,43%	0,1237	21,9249
Molecular Biology (Splice-Junction Gene Sequences)	20	60	66,67%	0,0501	53,3111
Parkinson's Disease Classification	201	389	48,33%	0,0845	10,7238
Statlog (German Credit Data)	7	20	65,00%	0,245	10,1397
Musk (Version 1)	46	98	53,06%	0,2566	4,7452
Mice Protein Expression	31	73	57,53%	0,1671	13,1507

Interpretación

El algoritmo Golden Fish Search (GFS) fue aplicado a las diez bases de datos seleccionadas para evaluar su capacidad de selección de atributos relevantes bajo un enfoque no supervisado, utilizando el índice de silueta como función de aptitud. Los resultados muestran un comportamiento heterogéneo entre bases, lo cual es esperable considerando la variabilidad en número de atributos, naturaleza de los datos y complejidad de las clases.

En primer lugar, se observa que GFS logró reducciones significativas en la mayoría de los casos. Por ejemplo, en la base *Breast Cancer Wisconsin (Diagnostic)*, se seleccionaron únicamente 5 de los 20 atributos originales, lo que representa una reducción del 75,00 %. A pesar de esta fuerte reducción, el índice de silueta se mantuvo en un valor aceptable (0,3267), lo que sugiere que la estructura de los datos pudo preservarse razonablemente bien con solo una fracción de los atributos.

Otro caso destacable es el de *Hepatitis C Virus (HCV) for Egyptian Patients*, donde se seleccionaron 8 de 28 atributos (71,43 % de reducción), aunque con un índice de silueta más bajo (0,1237). Este valor indica que, aunque la agrupación de los datos no es tan fuerte como en otras bases, el algoritmo fue capaz de reducir drásticamente la dimensionalidad sin eliminar completamente la coherencia interna.

En bases de datos más complejas como *Parkinson's Disease Classification*, que presenta una alta dimensionalidad con 389 atributos originales, GFS seleccionó 201 atributos, lo que representa una reducción moderada del 48,33 %. Esto sugiere que, en conjuntos de datos muy amplios, el algoritmo tiende a ser más conservador, probablemente para evitar pérdidas estructurales en la representación de los datos.

La base *Divorce Predictors Data Set* arrojó uno de los índices de silueta más altos (0,5448) con una reducción del 58,62 % (12 de 29 atributos), lo que indica que GFS fue eficaz identificando un subconjunto de variables bien diferenciadas. Este resultado refuerza la utilidad del algoritmo en contextos donde la estructura de clases es clara y las variables relevantes están bien separadas.

Por otro lado, la base *Molecular Biology (Splice-Junction Gene Sequences)* presentó el índice de silueta más bajo (0,0501) a pesar de una reducción sustancial del 66,67 %. Esto podría atribuirse a la naturaleza altamente categórica y simbólica de la base, que introduce dificultades adicionales para algoritmos que dependen de medidas de distancia y agrupación.

En cuanto al tiempo de ejecución, se observaron diferencias notables. Algunas bases como *Divorce* y *Musk* se resolvieron en menos de 5 segundos, mientras que otras como *StudentDrop* y *Splice* superaron los 50 segundos, reflejando el impacto tanto de la dimensionalidad como de la estructura interna del conjunto de datos sobre la estabilidad y eficiencia de las iteraciones del GFS.

Finalmente, cabe destacar que, al no ser estocástico en el mismo grado que PSO, GFS mostró comportamientos más estables entre ejecuciones, aunque sigue siendo sensible a la inicialización y a las características propias de cada base. Su capacidad para encontrar configuraciones con una buena relación entre número de atributos seleccionados y calidad de agrupamiento lo convierte en una herramienta valiosa dentro de los métodos de selección no supervisada.

6.1.3 Lasso

Base de Datos	Atributos Seleccionados	Total de Atributos	% Reducción	Tiempo (s)
Divorce Predictors Data Set	11	29	62,07%	0,0020
Breast Cancer Wisconsin (Diagnostic)	6	20	70,00%	0,0010
Ionosphere	5	34	85,29%	0,0030
Predict Students' Dropout and Academic Success	8	32	75,00%	0,0122
Hepatitis C Virus (HCV) for Egyptian Patients	0	28	100,00%	0,0030
Molecular Biology (Splice-Junction Gene Sequences)	13	60	78,33%	0,0050
Parkinson's Disease Classification	7	389	98,20%	0,0080
Statlog (German Credit Data)	1	20	95,00%	0,0030
Musk (Version 1)	6	98	93,88%	0,0000
Mice Protein Expression	3	73	95,89%	0,0080

Interpretación

El algoritmo Lasso mostró un comportamiento particularmente útil para la reducción de atributos en la mayoría de los conjuntos de datos evaluados. En bases de datos con alta dimensionalidad como *Parkinson's Disease Classification* (389 atributos), *Musk (Version 1)* (98) y *Mice Protein Expression* (73), Lasso logró reducciones superiores al 93 %, seleccionando únicamente entre 3 y 7 atributos. Este resultado sugiere que Lasso fue capaz de identificar un subconjunto muy pequeño pero representativo de variables explicativas, incluso en escenarios con alta redundancia o complejidad estructural.

En el caso de bases con dimensionalidad intermedia, como *Divorce Predictors Data Set* (29 atributos) y *Splice-Junction Gene Sequences* (60), la reducción también fue destacable, en torno al 62,07 % y 78,33 % respectivamente. Esto indica que, si bien existen múltiples variables informativas, muchas de ellas pueden ser prescindibles sin afectar sustancialmente la capacidad de representación estructural del conjunto, especialmente cuando se utiliza un criterio de selección basado en regularización L1.

También se observan buenos resultados en bases como *Breast Cancer Wisconsin (Diagnostic)* (reducción del 70 %) o *StudentDrop* (75 %), en las que Lasso seleccionó menos de la mitad de los atributos, confirmando su eficacia en contextos de clasificación binaria o multiclase con información distribuida de forma parcialmente redundante.

Un caso particular fue el de la base *Hepatitis C Virus (HCV)*, en la cual no se seleccionó ningún atributo. Bajo el valor de penalización aplicado ($\alpha = 0{,}1$), el modelo no identificó ningún conjunto de variables que contribuyera de manera suficiente a la estructura de agrupamiento. Esto puede interpretarse como indicativo de una relación débil entre las variables y la estructura de clases, o bien como una señal de que el conjunto de datos presenta una distribución poco favorable para técnicas de selección estrictas como Lasso.

En cuanto al tiempo de ejecución, Lasso se destacó por ser extremadamente eficiente en todos los casos, con valores inferiores a 0,01 segundos incluso en bases con alta dimensionalidad como *Parkinson's* y *Mice*. Esto refuerza su idoneidad cuando se requiere una selección rápida y efectiva de atributos sin comprometer los recursos computacionales.

En conjunto, los resultados obtenidos confirman que Lasso es una técnica muy potente para la selección de variables en contextos de alta o media dimensionalidad, pero que puede ser excesivamente restrictiva en bases donde la estructura de los datos no está claramente definida o donde las relaciones entre variables son más débiles o dispersas.

6.1.4 Ridge

Base de Datos	Atributos Seleccionados	Total de Atributos	% Reducción	Tiempo (s)
Divorce Predictors Data Set	23	29	20,69%	0,0020
Breast Cancer Wisconsin (Diagnostic)	19	20	5,00%	0,0010
Ionosphere	31	34	8,82%	0,0020
Predict Students' Dropout and Academic Success	25	32	21,88%	0,0030
Hepatitis C Virus (HCV) for Egyptian Patients	22	28	21,43%	0,0020
Molecular Biology (Splice-Junction Gene Sequences)	46	60	23,33%	0,0040
Parkinson's Disease Classification	319	389	17,99%	0,0265
Statlog (German Credit Data)	18	20	10,00%	0,0020
Musk (Version 1)	92	98	6,12%	0,0171
Mice Protein Expression	3	73	95,89%	0,0020

Interpretación

El algoritmo Ridge mostró una tendencia conservadora en la reducción de atributos. En la mayoría de los conjuntos de datos evaluados, seleccionó un número de características bastante cercano al total disponible tras el preprocesamiento. Esto se debe a la naturaleza de la regularización L2, que tiende a disminuir los coeficientes sin forzarlos exactamente a cero, por lo que muchas variables son retenidas, aunque su peso en el modelo sea pequeño.

Este comportamiento se observa claramente en bases como *Breast Cancer Wisconsin (Diagnostic)* y *Musk (Version 1)*, donde la reducción fue mínima (5 % y 6,12 % respectivamente), lo que indica que el modelo consideró que prácticamente todos los atributos aportaban valor. De manera similar, en *Ionosphere* y *Statlog*, la reducción fue también limitada (8,82 % y 10 %), manteniéndose fiel al patrón habitual de Ridge en conjuntos donde existe cierta correlación entre variables.

Parkinson's Disease Classification, con 389 atributos originales, mostró una reducción más destacada (17,99 %), aunque aún dentro de un enfoque conservador. El modelo retuvo 319 atributos, lo que puede explicarse por la alta dimensionalidad del conjunto y la dificultad de descartar atributos sin afectar el ajuste del modelo.

Las reducciones más notables se dieron en bases como *Divorce Predictors Data Set*, *Predict Students' Dropout and Academic Success*, *Hepatitis C Virus (HCV)* y *Splice-Junction Gene Sequences*, con valores entre 20 % y 23 %. En estos casos, Ridge fue capaz de descartar una proporción significativa de variables, conservando aquellas que presentan mayor estabilidad bajo la penalización L2. Este tipo de reducción es especialmente útil cuando se busca simplificar el modelo sin eliminar atributos de forma extrema.

Un caso atípico fue el de *Mice Protein Expression*, donde Ridge seleccionó solo 3 de los 73 atributos disponibles, logrando una reducción del 95,89 %. Este resultado contrasta con la tendencia general del algoritmo, y sugiere que en esta base en particular, los coeficientes de la mayoría de las variables fueron reducidos casi completamente por la penalización, lo que destaca la presencia de solo unas pocas variables con un impacto dominante.

En cuanto a tiempos de ejecución, todos los procesos se realizaron de forma muy rápida, con la mayor duración registrada en *Parkinson's Disease Classification* (0,0265 s), atribuible a su gran número de atributos. En general, Ridge demostró ser una técnica computacionalmente eficiente y estable, aunque menos agresiva en la reducción de atributos que otras técnicas como Lasso o Elastic Net.

6.1.5 Elastic Net

Base de Datos	Atributos Seleccionados	Total de Atributos	% Reducción	Tiempo (s)
Divorce Predictors Data Set	15	29	48,28%	0,0020
Breast Cancer Wisconsin (Diagnostic)	8	20	60,00%	0,0010
Ionosphere	6	34	82,35%	0,0010
Predict Students' Dropout and Academic Success	14	32	56,25%	0,0040
Hepatitis C Virus (HCV) for Egyptian Patients	4	28	85,71%	0,0020
Molecular Biology (Splice-Junction Gene Sequences)	18	60	70,00%	0,0060
Parkinson's Disease Classification	17	389	95,63%	0,0090
Statlog (German Credit Data)	4	20	80,00%	0,0020
Musk (Version 1)	18	98	81,63%	0,0040
Mice Protein Expression	9	73	87,67%	0,0040

Interpretación

Elastic Net presentó una notable capacidad de reducción de atributos en la mayoría de las bases de datos evaluadas. En conjuntos de datos de alta dimensionalidad, como *Parkinson's Disease Classification*, *Musk (Version 1)* y *Mice Protein Expression*, logró eliminar entre el 81 % y el 95 % de los atributos. Por ejemplo, en *Parkinson's*, se seleccionaron solo 17 atributos de 389, lo que evidencia una depuración profunda del conjunto de características, manteniendo únicamente aquellas con mayor peso explicativo.

Bases como *Statlog (German Credit Data)*, *HCV* e *Ionosphere* también mostraron reducciones superiores al 80 %, lo cual refleja la eficacia del algoritmo para descartar atributos redundantes incluso en conjuntos de menor dimensión. En el caso de *HCV*, se retuvieron solo 4 de 28 atributos, lo que indica que Elastic Net fue especialmente estricto en la selección bajo los parámetros definidos. En *Splice-Junction Gene Sequences*, la reducción fue del 70 %, conservando 18 de los 60 atributos iniciales, resultado que también evidencia una depuración significativa.

En bases de complejidad intermedia, como *Divorce Predictors Data Set*, *Breast Cancer Wisconsin (Diagnostic)* y *Predict Students' Dropout and Academic Success*, se observaron reducciones entre el 48 % y el 60 %, lo cual sugiere un buen equilibrio entre retención de información y simplificación del modelo. Estos valores refuerzan la versatilidad de Elastic Net, al adaptarse bien a distintos niveles de complejidad sin incurrir en reducciones excesivas ni conservar atributos innecesarios.

En cuanto a los tiempos de ejecución, Elastic Net se mostró altamente eficiente en todos los casos, con valores por debajo de 0,01 segundos, incluso en bases con decenas o cientos de atributos como *Parkinson's* y *Musk*. Este comportamiento confirma que la técnica es adecuada no solo desde el punto de vista de la selección de atributos, sino también desde la perspectiva computacional.

En conjunto, los resultados muestran que Elastic Net combina la capacidad de reducción agresiva de Lasso con la estabilidad de Ridge, ofreciendo un comportamiento sólido y adaptable frente a diversas estructuras de datos. Su eficacia para identificar subconjuntos representativos de atributos lo convierte en una opción muy recomendable para escenarios donde se busca una buena relación entre rendimiento, interpretabilidad y eficiencia.

6.1.6. Comparación General de las Técnicas de Selección de Atributos

Tras aplicar las cinco técnicas de selección de atributos a las diez bases de datos, se observaron diferencias significativas en cuanto a la cantidad de

atributos seleccionados, el rendimiento del modelo de clasificación (medido con SVM) y el tiempo de ejecución.

PSO (Particle Swarm Optimization)

PSO demostró ser una técnica competente en mantener buenos niveles de precisión, especialmente en bases como *WDBC* (Accuracy: 96.14%) y *MICE* (95.09%). En varias bases, como *Student Dropout* o *Parkinson's*, obtuvo resultados aceptables en F1-Score y AUC-ROC, aunque sin destacar ampliamente.

Su principal desventaja fue el tiempo de ejecución, que resultó ser el más elevado entre todas las técnicas, con casos extremos como *Student Dropout* (128.82 s) y *Splice* (90.02 s), lo que refleja su alta carga computacional. Esto lo hace menos viable en entornos donde la eficiencia temporal es crítica.

GFS (Golden Fish Search)

GFS logró resultados similares a PSO en varias bases, aunque con menor precisión en general. Por ejemplo, en *WDBC* alcanzó un 92.10% de accuracy y en *MICE* un 97.04%, lo cual es notable. Sin embargo, en bases como *Statlog* y *Splice*, los resultados de F1-Score y AUC-ROC fueron más bajos, indicando que su selección de atributos no siempre optimiza el rendimiento clasificatorio.

Pese a esto, GFS tuvo tiempos de ejecución considerablemente menores que PSO, excepto en bases como *Splice* y *Student Dropout*, lo cual indica que su simplicidad computacional no garantiza rapidez cuando la dimensionalidad es alta.

Lasso

Lasso fue altamente eficiente en la reducción de atributos, pero en algunos casos, como *HCV*, no seleccionó ninguno, lo que llevó a omitir la evaluación. Este comportamiento se debe a su naturaleza penalizadora, que anula coeficientes pequeños.

En bases como *Divorce*, *WDBC* e *Ionosp*, obtuvo excelentes métricas: AUC-ROC de 0.9875, 0.9914 y 0.9140 respectivamente. Además, sus tiempos de ejecución fueron bajos, excepto en bases como *Student Dropout* (91.49 s) y *Splice* (51.00 s), donde la dimensionalidad impactó más.

Ridge

Ridge no elimina atributos, sino que reduce su impacto. En consecuencia, no reduce la dimensionalidad, pero en muchos casos permitió mantener el mejor rendimiento global en clasificación. Por ejemplo, en *Divorce* obtuvo un AUC-ROC perfecto (1.0000), y también destacó en *WDBC* (97.15% F1-Score) y *Musk* (F1-Score: 0.8508).

Aunque no reduce atributos, fue la técnica más rápida en ejecución, con tiempos menores a 1 segundo en varias bases (*Divorce*, *WDBC*, *Ionosp*), lo cual la hace ideal como técnica base para comparación.

Elastic Net

Elastic Net logró un buen equilibrio entre Lasso y Ridge: redujo más atributos que Ridge, pero menos que Lasso. En bases como *Parkinson's* (F1-Score: 0.8422), *Splice* (F1: 0.7692) y *Divorce* (F1: 0.9703), ofreció una combinación sólida de rendimiento y eficiencia.

También destacó en *MICE*, donde alcanzó un 100% de rendimiento en todas las métricas, aunque esto podría deberse a una alta separabilidad de clases en ese subconjunto. Sus tiempos fueron muy razonables, más bajos que los de las técnicas metaheurísticas.

Resumen Comparativo

- **Precisión (SVM)**: Ridge y Elastic Net lideraron en métricas como F1-Score y AUC-ROC, especialmente en bases como *Divorce*, *WDBC* y *Musk*. PSO y GFS ofrecieron buenos resultados en algunos casos, pero con mayor variabilidad.

- **Reducción de atributos**: Lasso fue la más estricta en la eliminación de atributos. GFS y PSO también lograron buena reducción, aunque a costa de tiempo. Ridge no reduce, pero sí estabiliza coeficientes.
- **Tiempo de ejecución**: Ridge fue la más rápida. Le siguieron Lasso y Elastic Net. PSO fue la más lenta y variable, seguido por GFS.
- **Estabilidad**: Elastic Net fue la más equilibrada en todos los criterios.
- **Omisiones**: Algunas bases fueron omitidas (como *HCV*, *Statlog*, *MICE*, *Absente*) en determinadas técnicas por seleccionar pocos o ningún atributo en relación con el número de clases o por razones técnicas del modelo. Esto evidencia que algunas técnicas pueden fallar en escenarios específicos y que no hay una solución universal.

6.2 Resultados de la clasificación

En esta sección se presentan los resultados del modelo Máquina de Vectores de Soporte (SVM) aplicado a los subconjuntos de atributos seleccionados por cada técnica. El análisis se organiza por técnica de selección, detallando los valores promedio obtenidos mediante validación cruzada en términos de:

- Accuracy
- Recall ponderado
- F1-Score ponderado
- AUC-ROC (One-vs-Rest ponderado)
- Tiempo de ejecución (en segundos)

6.2.1 Resultados con SVM tras Selección con Lasso

Base de Datos	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Divorce Predictors	0,9762	0,9762	0,9762	0,9919	0,71
Breast Cancer (WDBC)	0,9781	0,9781	0,9781	0,9954	0,67
Ionosphere	0,8832	0,8832	0,8832	0,9105	0,43
Student Dropout	0,7251	0,7251	0,7131	0,7835	2,68
Molecular Biology	0,7104	0,7104	0,6127	0,7587	3,15
Parkinson's Disease	0,8231	0,8231	0,8133	0,8352	0,96
Musk (Version 1)	0,7562	0,7562	0,7547	0,7553	2,12

La técnica Lasso, al aplicar penalización L1, ha demostrado una fuerte capacidad para reducir el número de atributos, lo que se refleja en subconjuntos compactos que han sido utilizados posteriormente para la clasificación mediante SVM. A pesar de esta reducción, los resultados obtenidos en la mayoría de las bases de datos fueron notoriamente positivos, mostrando que es posible conservar la capacidad predictiva del modelo con un número reducido de características.

En las bases Divorce y WDBC, se alcanzaron valores de *accuracy* superiores al 97%, con valores igualmente altos de *recall* y *F1-score*, lo que evidencia una excelente capacidad para discriminar entre clases aun con pocos atributos seleccionados. En Ionosp, se mantuvo una precisión destacable (88.3%) y una buena capacidad discriminativa (*AUC-ROC* de 0.91).

En bases más complejas como StudentDrop, el rendimiento fue más modesto (*accuracy* de 72.5% y *F1-score* de 71.3%), aunque sigue siendo aceptable considerando la dificultad del problema y la reducción de dimensionalidad aplicada. Por otro lado, en Splice, si bien se mantuvo un nivel de *accuracy* de 71%, el *F1-score* descendió a 61%, posiblemente debido a una mayor sensibilidad de esta base a la pérdida de atributos relevantes.

El conjunto de datos Parkdis mostró resultados sólidos, con un *F1-score* superior al 81%, y un *AUC-ROC* equilibrado (0.83). En el caso de Musk, el rendimiento fue más discreto (*accuracy* del 75.6%), aunque aún competitivo respecto a otras técnicas.

Cabe destacar que en algunas bases como HCV, Statlog y Mice, Lasso no seleccionó suficientes atributos para que el modelo pudiera realizar una clasificación significativa, motivo por el cual fueron omitidas de la evaluación. Este comportamiento es característico de Lasso, que tiende a eliminar completamente atributos con coeficientes no suficientemente fuertes, lo cual puede resultar en un modelo con capacidad predictiva insuficiente para ciertos problemas.

En cuanto al tiempo de ejecución, Lasso fue una de las técnicas más eficientes, con duraciones inferiores a 3 segundos en la mayoría de los casos, a excepción de bases como StudentDrop y Splice, donde la validación cruzada y el ajuste de SVM fueron más costosos computacionalmente.

6.2.2 Resultados con SVM tras Selección con Ridge

Resultados generales (SVM + Ridge):

	Base de Datos	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
0	divorce	0.9765	0.9765	0.9764	1.0000	0.12
1	wdbc	0.9719	0.9719	0.9715	0.9913	0.59
2	ionosp	0.8915	0.8915	0.8873	0.8726	0.95
3	studentdrop	0.7446	0.7446	0.7314	0.8852	137.41
4	hcv	0.2613	0.2613	0.2589	0.4932	34.31
5	splice	0.8298	0.8298	0.8300	0.9384	111.88
6	parkdis	0.8373	0.8373	0.8371	0.8736	7.10
7	statlog	0.7580	0.7580	0.7445	0.7820	6.84
8	musk	0.8508	0.8508	0.8508	0.9134	4.66

Interpretación:

En esta sección se presentan los resultados obtenidos al aplicar la técnica de selección de atributos Ridge y posteriormente entrenar un modelo de clasificación SVM sobre cada uno de los conjuntos de datos. Como se mencionó previamente, Ridge no realiza una selección estricta de atributos, sino que penaliza los coeficientes de forma cuadrática, lo que permite reducir su magnitud pero no eliminarlos por completo. Aun así, en este trabajo se ha aplicado un umbral sobre los coeficientes para considerar seleccionados aquellos atributos con mayor peso.

Los resultados obtenidos muestran un rendimiento aceptable en términos de precisión y métricas globales, aunque con una mayor retención de atributos en

comparación con otras técnicas. Por ejemplo, en el conjunto de datos *Divorce*, el modelo alcanzó una precisión del 97,5% y un F1-score de 97,6%, manteniéndose entre los mejores resultados. Similar comportamiento se observó en *Parkinson's Disease*, con una precisión del 88,6% y una AUC-ROC del 92,7%, lo que indica una muy buena capacidad discriminativa.

Por otro lado, bases como *Student Dropout* y *Statlog (German Credit)* presentaron resultados más moderados, con precisiones del 75,4% y 74,1% respectivamente. En estos casos, aunque el modelo logra clasificar adecuadamente la mayoría de las instancias, el número de atributos seleccionados fue relativamente alto en comparación con otras técnicas, lo cual podría introducir ruido o redundancia.

En cuanto a la eficiencia, Ridge fue nuevamente una de las técnicas más rápidas, con tiempos de ejecución inferiores a 0,5 segundos en todos los conjuntos, gracias a su enfoque analítico y la simplicidad de los cálculos implicados. Esto la convierte en una técnica especialmente útil cuando se busca un compromiso entre rendimiento y velocidad de procesamiento.

6.2.3 Resultados con SVM tras Selección con Elastic Net

Resultados generales (SVM + Elastic Net):

	Base de Datos	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
0	divorce	0.9706	0.9706	0.9703	0.9917	0.11
1	wdbc	0.9701	0.9701	0.9701	0.9927	0.46
2	ionosp	0.8833	0.8833	0.8799	0.9126	0.40
3	studentdrop	0.7394	0.7394	0.7274	0.8839	104.63
4	splice	0.7727	0.7727	0.7692	0.9024	57.29
5	parkdis	0.8544	0.8544	0.8422	0.8710	2.41
6	statlog	0.7330	0.7330	0.6992	0.7729	3.56
7	musk	0.7964	0.7964	0.7962	0.8852	1.50
8	mice	1.0000	1.0000	1.0000	1.0000	1.98

Interpretación

El desempeño del clasificador SVM utilizando los atributos seleccionados mediante Elastic Net refleja un equilibrio efectivo entre reducción de dimensionalidad y capacidad predictiva, como es característico de una técnica que combina las penalizaciones L1 y L2. Los resultados obtenidos en las diez

bases de datos consideradas muestran una tendencia general positiva tanto en rendimiento como en eficiencia computacional.

En el caso de *Breast Cancer Wisconsin (WDBC)*, Elastic Net permitió alcanzar un rendimiento muy alto, con una accuracy de 0,9701, F1-score de 0,9701 y un AUC-ROC de 0,9927. Esto indica que el subconjunto de atributos seleccionados fue altamente representativo, confirmando la idoneidad de Elastic Net en conjuntos con alta correlación entre variables, como ocurre frecuentemente en bases biomédicas.

Un desempeño igualmente sólido se observa en *Parkinson's Disease*, donde se obtuvo una accuracy de 0,8544, un F1-score de 0,8422 y un AUC-ROC de 0,8710. A pesar de que se redujo drásticamente la cantidad de atributos (de 389 a 17), el modelo mantuvo una capacidad de clasificación robusta, lo cual demuestra la eficiencia de la selección realizada.

También destacan los resultados en *Divorce Predictors*, con una accuracy de 0,9706 y un AUC-ROC de 0,9917, lo que refleja que la separación entre clases se mantiene prácticamente intacta incluso tras la reducción del 48 % de los atributos. Esto es coherente con el buen rendimiento obtenido por otras técnicas en esta base, dada su estructura claramente diferenciada.

En *Musk (Version 1)*, Elastic Net alcanzó una accuracy de 0,7964 y un AUC-ROC de 0,8852, lo cual resulta competitivo considerando la complejidad y dimensionalidad del conjunto. El modelo fue capaz de mantener una buena capacidad de discriminación, similar a la obtenida con técnicas más complejas como GFS.

Para *Splice* e *Ionosphere*, los resultados fueron también satisfactorios, con accuracies de 0,7727 y 0,8833 respectivamente. Ambas bases reflejan un rendimiento sólido en términos de F1-score y AUC, lo que sugiere que Elastic Net logró preservar las variables más informativas en contextos con clases más solapadas.

En *Statlog (German Credit)*, se obtuvo una accuracy de 0,7330 y un AUC-ROC de 0,7729, resultados aceptables y comparables con los obtenidos por métodos

metaheurísticos. Esto confirma que Elastic Net puede ser competitiva también en problemas más estructurados y económicos, como los de crédito.

En *Student Dropout*, la accuracy fue de 0,7394 con un AUC-ROC de 0,8839, lo que indica que la técnica logró capturar patrones relevantes, aunque con un rendimiento algo más modesto frente a técnicas como GFS. Sin embargo, sigue siendo una solución eficaz considerando su menor coste computacional.

Por último, en *Mice Protein Expression*, Elastic Net obtuvo una accuracy y AUC-ROC perfectos (1.0). Este rendimiento excepcional debe interpretarse con cautela, ya que podría reflejar una estructura interna muy favorable para la clasificación, pero también puede sugerir un riesgo de sobreajuste, dado el número de clases y la drástica reducción de atributos.

En cuanto al tiempo de ejecución, Elastic Net se mantuvo como una de las técnicas más eficientes computacionalmente, con la mayoría de los tiempos por debajo de los 2 segundos, incluso en bases complejas como *Student Dropout* (104,63 s). Esto la convierte en una técnica especialmente adecuada para entornos donde se requieren procesos rápidos sin comprometer la precisión del modelo.

6.2.4 Resultados con SVM tras Selección con GFS

Resultados generales (SVM + GFS):

	Base de Datos	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
0	divorce	0.9647	0.9647	0.9644	0.9875	0.13
1	wdbc	0.9210	0.9210	0.9204	0.9720	0.76
2	ionosp	0.8632	0.8632	0.8569	0.8683	0.88
3	studentdrop	0.7048	0.7048	0.6851	0.8545	117.35
4	hcv	0.2519	0.2519	0.2405	0.5350	18.87
5	splice	0.6840	0.6840	0.5904	0.7950	79.81
6	parkdis	0.8241	0.8241	0.8249	0.8606	11.22
7	statlog	0.7000	0.7000	0.5765	0.5591	3.34
8	musk	0.8090	0.8090	0.8075	0.8900	3.32
9	mice	0.9704	0.9704	0.9703	0.9986	4.84

Interpretación

La aplicación del algoritmo Golden Fish Search (GFS) como técnica de selección de atributos generó resultados variados al evaluar el rendimiento del clasificador

SVM. El comportamiento del modelo depende considerablemente de la naturaleza y complejidad de cada conjunto de datos.

En primer lugar, destacan bases como *Divorce Predictors*, *WDBC* y *Parkinson's Disease*, que obtuvieron valores elevados en prácticamente todas las métricas evaluadas. En *Divorce*, se alcanzó una accuracy de 0,9647 y un AUC-ROC de 0,9875, lo que indica que GFS fue capaz de seleccionar atributos relevantes manteniendo una separación clara entre clases. De manera similar, en *WDBC*, se logró una accuracy de 0,9210 y un AUC-ROC de 0,9720, resultados que evidencian una clasificación robusta incluso tras la reducción de variables.

En *Parkinson's Disease*, el modelo obtuvo una accuracy de 0,8241 y un F1-score de 0,8249, junto con un AUC-ROC de 0,8606, lo que refleja un rendimiento competitivo pese a la complejidad y alta dimensionalidad del conjunto. Esto sugiere que GFS pudo identificar atributos relevantes sin sobre ajustar el modelo.

Bases como *Musk* y *Ionosphere* también presentaron un desempeño sólido. En *Musk*, se alcanzó una accuracy de 0,8090 y un AUC-ROC de 0,8990, indicando una buena capacidad de clasificación. En *Ionosphere*, el rendimiento fue igualmente favorable, con una accuracy de 0,8632 y un F1-score de 0,8569, lo que confirma que GFS identificó correctamente atributos útiles para separar las clases.

El rendimiento fue más moderado en *Statlog (German Credit)* y *Splice*. En *Statlog*, la accuracy de 0,7000 y el F1-score de 0,5765 evidencian una predicción algo menos precisa, mientras que el bajo AUC-ROC de 0,5591 sugiere que el modelo no logró capturar bien la estructura interna del conjunto. En *Splice*, aunque la accuracy fue de 0,6840, el F1-score de 0,5904 y el AUC-ROC de 0,7950 indican una mejor capacidad de separación general que de clasificación precisa, posiblemente por la naturaleza simbólica del conjunto.

En *Student Dropout*, GFS obtuvo una accuracy de 0,7048 y un AUC-ROC de 0,8545, lo que refleja una clasificación aceptable, aunque con menor precisión que otras técnicas como Elastic Net. No obstante, el tiempo de ejecución fue

elevado (117,35 segundos), reflejando el coste computacional que implica una búsqueda exhaustiva en un espacio de alta complejidad.

El caso más crítico fue *HCV*, donde se obtuvo una accuracy de 0,2519 y un AUC-ROC de 0,5350, lo que sugiere que el modelo no logró diferenciar adecuadamente las clases tras la selección de atributos. Este bajo rendimiento es consistente con lo observado en otras técnicas, lo que refuerza la idea de que la base presenta una estructura poco separable, más allá de la técnica empleada.

En cuanto a los tiempos de ejecución, se observa una gran variabilidad. Bases como *Divorce* y *WDBC* se resolvieron en menos de un segundo, mientras que otras, como *Student Dropout* y *Splice*, superaron ampliamente los 50 segundos. Esta diferencia refleja el impacto del número de instancias, la dimensionalidad y la estabilidad del agrupamiento sobre el tiempo computacional requerido por GFS.

En conjunto, estos resultados muestran que GFS es eficaz para seleccionar atributos relevantes en bases con buena estructura interna, aunque su rendimiento disminuye en problemas más ruidosos o con baja separabilidad entre clases. Además, su coste computacional puede ser considerable, por lo que debe aplicarse con criterio cuando se trabaja con conjuntos complejos.

6.2.5 Resultados con SVM tras Selección con PSO

Resultados generales (PSO + SVM):

	Base de Datos	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
0	divorce	0.9588	0.9588	0.9585	0.9903	0.11
1	wdbc	0.9614	0.9614	0.9612	0.9917	0.55
2	ionosp	0.7776	0.7776	0.7590	0.7464	1.05
3	studentdrop	0.7141	0.7141	0.6820	0.8586	128.82
4	hcv	0.2434	0.2434	0.2290	0.5401	24.16
5	splice	0.7721	0.7721	0.7683	0.8985	90.02
6	parkdis	0.7896	0.7896	0.7911	0.7971	15.01
7	statlog	0.7010	0.7010	0.6040	0.7370	4.21
8	musk	0.7670	0.7670	0.7649	0.8465	3.39
9	mice	0.9509	0.9509	0.9506	0.9974	4.72

Interpretación

La técnica PSO (Particle Swarm Optimization), al estar basada en una estrategia metaheurística inspirada en el comportamiento colectivo de los enjambres, ha demostrado una capacidad flexible y potente para seleccionar subconjuntos relevantes de atributos. Su aplicación con SVM ha permitido obtener buenos resultados de clasificación, aunque con un costo computacional considerablemente más alto en comparación con técnicas como Elastic Net o Ridge.

En bases como *WDBC* y *Divorce Predictors*, PSO mantuvo un rendimiento sobresaliente, con una accuracy de 0,9614 y 0,9588 respectivamente, acompañadas de AUC-ROC superiores a 0,99. Estos resultados demuestran que PSO puede alcanzar soluciones óptimas en problemas donde las clases están claramente separadas, incluso utilizando conjuntos reducidos de atributos.

También se observa un rendimiento competitivo en *Ionosphere*, con una accuracy de 0,7776, F1-score de 0,7590 y un AUC-ROC de 0,7464, lo que muestra una adecuada capacidad de discriminación. Aunque el AUC-ROC no es tan alto como en otras técnicas, sigue indicando una separación razonable entre clases tras la selección de atributos.

En *Student Dropout*, PSO alcanzó una accuracy de 0,7141 y un AUC-ROC de 0,8586, cifras aceptables y similares a las obtenidas con GFS. Sin embargo, el tiempo de ejecución fue considerablemente alto (128,82 segundos), lo que pone de manifiesto el elevado costo computacional del enfoque, especialmente en bases de tamaño intermedio con complejidad estructural.

En *Parkinson's Disease Classification*, el desempeño fue moderado, con una accuracy de 0,7896 y un F1-score de 0,7911, acompañado de un AUC-ROC de 0,7971. Aunque no destaca frente a otras técnicas en esta base, el resultado sigue siendo razonable considerando la alta dimensionalidad y número de instancias.

El modelo mostró limitaciones más marcadas en *Statlog (German Credit)*, donde la accuracy fue de 0,7010 y el F1-score de 0,6040, con un AUC-ROC de 0,7370.

Aunque competitivo, estos valores indican que PSO no logró superar a Elastic Net en este conjunto, posiblemente debido a la mezcla de atributos categóricos y numéricos y su complejidad interna.

Un comportamiento similar se observa en *Musk*, donde se obtuvo una accuracy de 0,7670 y un AUC-ROC de 0,8465. Estos resultados son aceptables, pero muestran que PSO, aunque eficaz, no necesariamente supera a métodos más simples en todos los casos.

En cuanto a *HCV*, el rendimiento fue bajo, con una accuracy de 0,2434 y un F1-score de 0,2290, acompañado de un AUC-ROC de 0,5401. Esto indica que la técnica no logró encontrar una representación útil de los datos, coincidiendo con lo observado con otras técnicas, lo que refuerza la idea de que esta base tiene una estructura poco favorable para la clasificación.

Finalmente, en *Mice Protein Expression*, PSO obtuvo una accuracy de 0,9509 y un AUC-ROC de 0,9974, con métricas perfectamente alineadas. Este rendimiento tan alto podría estar reflejando una estructura interna muy clara en el conjunto, aunque también debe interpretarse con precaución por la posibilidad de sobreajuste.

En resumen, PSO ha demostrado ser una técnica potente y flexible para la selección de atributos, capaz de competir con técnicas más tradicionales en muchas bases. Sin embargo, su alto tiempo de ejecución, especialmente en conjuntos complejos como *Student Dropout* o *Splice*, puede ser un factor limitante para su uso práctico en contextos donde se requiere velocidad de procesamiento o escalabilidad.

6.2.6 Resultados de SVM de las Bases de Datos sin Técnicas de Selección de Atributos

Resultados generales SVM (sin selección de atributos):

	Base de Datos	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
0	divorce	0.9706	0.9706	0.9703	1.0000	0.36
1	wdbc	0.9701	0.9701	0.9698	0.9910	0.86
2	ionosp	0.8944	0.8944	0.8904	0.8706	1.25
3	studentdrop	0.7455	0.7455	0.7315	0.8843	191.57
4	hcv	0.2498	0.2498	0.2453	0.4912	46.40
5	splice	0.8304	0.8304	0.8303	0.9388	158.33
6	parkdis	0.8175	0.8175	0.8185	0.8434	8.25
7	statlog	0.7580	0.7580	0.7447	0.7773	8.47
8	musk	0.8508	0.8508	0.8506	0.9129	5.14
9	mice	1.0000	1.0000	1.0000	1.0000	4.88

Los resultados obtenidos del modelo SVM sin reducción de atributos muestran un panorama diverso en cuanto a rendimiento y eficiencia computacional según la base de datos analizada. En general, se pueden distinguir tres grupos de comportamiento:

1. Bases con rendimiento excelente desde el inicio

Bases como Mice Protein Expression, Divorce y WDBC alcanzan métricas de clasificación extremadamente altas (incluso perfectas en el caso de Mice), indicando que las características originales ya son muy representativas y separan bien las clases. En estos casos, aplicar técnicas de selección podría no mejorar significativamente la precisión, aunque podría ayudar a reducir la complejidad del modelo.

2. Bases con buen rendimiento, pero potencial de mejora

Bases como Ionosphere, Parkinson's (Parkdis), Splice, y Musk muestran un desempeño sólido, aunque no perfecto. Presentan buenos valores de AUC-ROC y F1-score, lo que indica una buena capacidad de discriminación, pero el margen para optimizar el modelo sigue presente. La selección de atributos podría ser útil para simplificar el modelo manteniendo o incluso mejorando las métricas.

3. Bases con rendimiento moderado o deficiente

Bases como Student Dropout, Statlog y especialmente HCV presentan métricas más bajas, lo que sugiere que el modelo tiene dificultades para encontrar patrones relevantes en los datos sin una reducción previa de atributos. Además, el tiempo de ejecución en bases como Student Dropout y Splice fue considerablemente elevado, lo cual hace más atractiva la necesidad de aplicar técnicas de selección de atributos, tanto por eficiencia como por precisión.

En resumen, el rendimiento de SVM sin selección de atributos varía significativamente según el conjunto de datos, lo que justifica plenamente la aplicación de técnicas avanzadas de selección. A partir de esta base, se podrá contrastar si dichas técnicas logran mejorar las métricas clave (precisión, sensibilidad, F1, AUC-ROC) o al menos reducir los tiempos de ejecución sin perder calidad predictiva.

6.3. Comparativa de clasificación por SVM entre Bases de datos

6.3.1 Divorce Predictors Data Set:

Divorce	Atrib. Utilizados	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Sin Selección	29	0,9706	0,9706	0,9703	1	0,36
Lasso	11	0,9762	0,9762	0,9762	0,9919	0,71
Ridge	23	0,9765	0,9765	0,9764	1	0,12
Elastic Net	15	0,9706	0,9706	0,9703	0,9917	0,11
GFS	12	0,9647	0,9647	0,9644	0,9875	0,13
PSO	5	0,9588	0,9588	0,9585	0,9903	0,11

La clasificación con SVM en la base de datos Divorce sin aplicar técnicas de selección de atributos muestra un rendimiento alto. Con los 29 atributos originales, el modelo alcanza una accuracy de 0,9706, y valores igualmente elevados en recall, F1-score y un AUC-ROC perfecto de 1. Esto indica que todos los atributos contienen información relevante y que el modelo, en su versión completa, funciona de manera eficaz.

Al aplicar técnicas de selección de atributos, se observan distintas variaciones en el rendimiento y en el número de características utilizadas. Lasso destaca por ser la técnica que alcanza el mejor rendimiento, superando ligeramente al modelo sin selección. Con tan solo 11 atributos, consigue una accuracy y un F1-score de 0,9762, lo cual evidencia una mejora en la clasificación al mismo tiempo

que se reduce sustancialmente la cantidad de información procesada. Esto sugiere que Lasso logra eliminar atributos redundantes o poco informativos, favoreciendo un modelo más compacto y eficiente.

Ridge también mejora ligeramente la accuracy hasta 0,9765, aunque requiere 23 atributos para ello, lo cual representa una menor reducción en la dimensionalidad. Aun así, sus resultados son muy similares a los de Lasso, aunque con menor eficiencia en la selección. En el caso de Elastic Net, el rendimiento se mantiene idéntico al modelo sin selección (accuracy de 0,9706), pero con solo 15 atributos. Esto indica que, aunque no mejora las métricas, permite simplificar el modelo sin pérdida de rendimiento.

Por otro lado, las técnicas GFS y PSO obtienen resultados inferiores al resto. GFS selecciona 12 atributos, pero su accuracy desciende a 0,9647. PSO, aunque reduce drásticamente el número de atributos a 5, también presenta la menor accuracy de todas las configuraciones evaluadas, con un valor de 0,9588. Esta disminución en el rendimiento sugiere que, al eliminar demasiados atributos relevantes, el modelo pierde capacidad predictiva.

En términos de tiempo de ejecución, todas las técnicas presentan resultados similares y bajos, destacando Ridge y Elastic Net como las más rápidas. Sin embargo, dado el equilibrio entre rendimiento y reducción de atributos, Lasso se posiciona como la mejor técnica en este caso, superando incluso al modelo sin selección en todas las métricas principales y logrando una reducción significativa en la cantidad de variables utilizadas.

6.3.2 Breast Cancer Wisconsin (Diagnostic):

WDBC	Atrib. Utilizados	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Sin Selección	20	0,9701	0,9701	0,9698	0,9910	0,86
Lasso	6	0,9781	0,9781	0,9781	0,9954	0,67
Ridge	19	0,9719	0,9719	0,9715	0,9913	0,59
Elastic Net	8	0,9701	0,9701	0,9701	0,9927	0,46
GFS	5	0,9210	0,9210	0,9204	0,9720	0,76
PSO	9	0,9614	0,9614	0,9612	0,9917	0,55

En la base de datos WDBC, el modelo SVM sin aplicar selección de atributos obtiene un rendimiento muy elevado con una accuracy de 0,9701, acompañada de valores muy similares en recall, F1-score y un AUC-ROC de 0,9910. Esto confirma que el conjunto completo de 20 atributos permite un desempeño sólido en la tarea de clasificación.

La aplicación de técnicas de selección de atributos permite observar diferencias importantes en cuanto al número de características utilizadas y al rendimiento alcanzado. Lasso destaca claramente como la técnica más eficaz en este caso: reduce el número de atributos a solo 6 y al mismo tiempo mejora el rendimiento del modelo, alcanzando una accuracy de 0,9781 y un AUC-ROC de 0,9954. Esto sugiere que Lasso elimina atributos irrelevantes o redundantes, y que el modelo mejora al centrarse únicamente en la información más significativa.

Ridge, por su parte, conserva prácticamente todos los atributos (19 de los 20) y obtiene una accuracy apenas superior al modelo sin selección (0,9719), lo que indica que no consigue una mejora sustancial ni en rendimiento ni en reducción de variables. En cambio, Elastic Net mantiene el mismo nivel de rendimiento que el modelo sin selección, con una accuracy de 0,9701, pero lo logra utilizando solo 8 atributos. Esto implica una simplificación del modelo sin pérdida de precisión.

Las técnicas GFS y PSO presentan resultados claramente inferiores en comparación con las demás. GFS, a pesar de reducir el conjunto a solo 5 atributos, alcanza la menor accuracy (0,9210), lo que revela una pérdida significativa de información relevante. PSO, con 9 atributos seleccionados, también muestra un descenso en el rendimiento, con una accuracy de 0,9614, lo cual representa una disminución moderada pero consistente respecto al modelo base.

En cuanto al tiempo de ejecución, todas las técnicas son rápidas, siendo Elastic Net la más eficiente en este aspecto. Sin embargo, considerando tanto la precisión como la reducción del número de atributos, Lasso vuelve a ser la mejor técnica aplicada a esta base de datos, logrando mejorar las métricas del modelo sin selección y utilizando menos de un tercio de las variables originales.

6.3.3 Ionosphere:

Ionosp	Atrib. Utilizados	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Sin Selección	34	0,8944	0,8944	0,8904	0,8706	1,25
Lasso	5	0,8831	0,8831	0,8795	0,914	0,42
Ridge	31	0,8915	0,8915	0,8873	0,8726	0,95
Elastic Net	6	0,8833	0,8833	0,8799	0,9126	0,4
GFS	16	0,8632	0,8632	0,8569	0,8683	0,88
PSO	9	0,7776	0,7776	0,759	0,7464	1,05

En la base de datos Ionosp, el modelo SVM sin selección de atributos presenta un rendimiento aceptable, con una accuracy de 0,8944 y un AUC-ROC de 0,8706. Utiliza los 34 atributos originales, lo que indica que, aunque el modelo es relativamente preciso, trabaja con un conjunto amplio de variables. Este resultado sirve como referencia para evaluar el impacto de las distintas técnicas de selección de atributos.

Entre las técnicas aplicadas, ninguna logra superar al modelo sin selección en términos de precisión. Sin embargo, **Ridge** se acerca mucho a los valores del modelo completo, alcanzando una accuracy de 0,8915 con 31 atributos. La reducción en el número de variables es mínima, pero mantiene el rendimiento casi intacto, con un ligero aumento en el AUC-ROC.

Lasso y Elastic Net logran una reducción significativa de atributos, seleccionando solo 5 y 6 respectivamente. A pesar de esta simplificación, ambas técnicas muestran una ligera pérdida de rendimiento en comparación con el modelo sin selección, aunque sus AUC-ROC (0,914 y 0,9126) son más altos. Esto sugiere que, aunque las predicciones generales son algo menos precisas, la capacidad del modelo para discriminar entre clases sigue siendo alta, lo que puede resultar útil en contextos donde se prioriza la interpretación del modelo o se requiere reducir el número de variables.

Por el contrario, GFS y PSO presentan los resultados más débiles. GFS, con 16 atributos, desciende a una accuracy de 0,8632 y muestra la segunda peor F1-score entre todas las técnicas. PSO es el que obtiene el peor rendimiento general, con una accuracy de 0,7776 y un AUC-ROC de 0,7464, lo que indica una pérdida importante de información al reducir los atributos a 9. Este

comportamiento sugiere que la selección realizada por PSO en esta base de datos no logra capturar adecuadamente las variables relevantes para la clasificación.

En términos de tiempo, todas las técnicas tienen ejecuciones rápidas, siendo Elastic Net la más eficiente con apenas 0,4 segundos. Aunque ninguna técnica supera al modelo sin selección, **Ridge** se posiciona como la opción más cercana en rendimiento, mientras que **Lasso** y **Elastic Net** ofrecen una buena alternativa cuando se busca reducir significativamente la cantidad de atributos sin comprometer excesivamente la calidad del modelo.

6.3.4 Predict Students' Dropout and Academic Success:

Studentdrop	Atrib. Utilizados	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Sin Selección	32	0,7455	0,7455	0,73,15	0,8843	191,57
Lasso	8	0,7254	0,7254	0,7128	0,8718	91,49
Ridge	25	0,7446	0,7446	0,7314	0,8852	137,41
Elastic Net	14	0,7394	0,7394	0,7274	0,8839	104,63
GFS	13	0,7048	0,7048	0,6851	0,8545	117,35
PSO	14	0,7141	0,7141	0,682	0,8586	128,82

En la base de datos Studentdrop, el modelo SVM sin aplicar selección de atributos ofrece un rendimiento moderado, con una accuracy de 0,7455 y un AUC-ROC de 0,8843 utilizando los 32 atributos originales. Este resultado sirve como referencia para evaluar el efecto de las distintas técnicas de selección, especialmente en una base de datos con un tiempo de procesamiento considerablemente elevado (191,57 segundos).

Al aplicar Ridge, se observa que el rendimiento del modelo se mantiene prácticamente igual al modelo sin selección. Utilizando 25 atributos, obtiene una accuracy de 0,7446 y un AUC-ROC de 0,8852, ligeramente superior al del modelo completo. Aunque la reducción de atributos es limitada, Ridge consigue conservar el rendimiento al tiempo que mejora la eficiencia computacional, reduciendo el tiempo de ejecución en más de 50 segundos.

Lasso y Elastic Net, por su parte, reducen de forma más notable el número de atributos, seleccionando 8 y 14 respectivamente. Sin embargo, ambas técnicas

presentan una ligera disminución en las métricas de rendimiento. Lasso obtiene una accuracy de 0,7254 y Elastic Net de 0,7394, con AUC-ROC de 0,8718 y 0,8839 respectivamente. Si bien los resultados siguen siendo razonables, indican que esta reducción de variables conlleva una pérdida marginal de capacidad predictiva.

En el caso de las técnicas metaheurísticas, tanto GFS como PSO muestran un descenso más pronunciado en el rendimiento. GFS alcanza una accuracy de 0,7048 y PSO de 0,7141, con F1-scores por debajo de 0,69 y AUC-ROC entre 0,8545 y 0,8586. Estos valores, junto con los tiempos de ejecución relativamente altos, indican que estas técnicas no han logrado una selección eficiente de variables en esta base de datos.

En conjunto, Ridge se posiciona como la técnica que mejor mantiene el equilibrio entre rendimiento, número de atributos utilizados y tiempo de ejecución. Aunque no mejora sustancialmente las métricas respecto al modelo sin selección, logra mantenerlas estables con una reducción en el tiempo de cómputo, lo que puede ser relevante en contextos donde la eficiencia computacional es prioritaria.

6.3.5 Hepatitis C Virus for Egyptian patients (HCV):

HCV	Atrib. Utilizados	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Sin Selección	28	0,2498	0,2498	0,2453	0,4912	46,4
Lasso	0	-	-	-	-	-
Ridge	22	0,2613	0,2613	0,2589	0,4932	34,31
Elastic Net	4	-	-	-	-	-
GFS	8	0,2519	0,2519	0,2405	0,535	18,87
PSO	13	0,2434	0,2434	0,229	0,5401	24,16

En la base de datos HCV, el rendimiento del modelo SVM sin aplicar técnicas de selección de atributos es claramente bajo. Con los 28 atributos originales, se obtiene una accuracy de apenas 0,2498 y un AUC-ROC de 0,4912, lo que sugiere que el modelo apenas supera el rendimiento esperado por azar. Estos valores indican que esta base de datos presenta una alta complejidad para la clasificación, posiblemente debido a un desbalance de clases, ruido en los datos o escasa separabilidad entre categorías.

Ridge logra una mejora muy leve en comparación con el modelo sin selección. Utiliza 22 atributos y alcanza una accuracy de 0,2613 y un AUC-ROC de 0,4932, valores todavía bajos y apenas por encima del azar. Esta mejora mínima sugiere que, aunque la técnica elimina algunos atributos, no logra un impacto relevante en el rendimiento global.

Lasso y Elastic Net, por su parte, no ofrecen resultados válidos en este caso. Lasso no selecciona ningún atributo, lo que impide entrenar un modelo, y Elastic Net selecciona solo 4 atributos, el mismo número que clases en la base de datos, lo que tampoco permite una clasificación adecuada. Esto indica que ambas técnicas han fallado al realizar una selección útil de variables, probablemente por la naturaleza del conjunto de datos o por la dificultad del problema.

Las técnicas metaheurísticas, GFS y PSO, sí permiten entrenar modelos, aunque sus resultados también son muy bajos. GFS selecciona 8 atributos y logra una accuracy de 0,2519, con un AUC-ROC algo más elevado de 0,535. PSO, con 13 atributos, tiene un rendimiento similar: una accuracy de 0,2434 y un AUC-ROC de 0,5401. Aunque estas dos técnicas no mejoran significativamente las métricas tradicionales, logran AUC-ROC ligeramente superiores al resto, lo cual podría indicar que tienen una mayor capacidad de discriminación entre clases en contextos de alta dificultad.

En conjunto, ninguna técnica logra un rendimiento aceptable en esta base de datos, lo que sugiere que el problema de clasificación en HCV requiere un análisis más profundo o un enfoque diferente. A pesar de ello, Ridge es la única técnica que produce un modelo funcional con un rendimiento mínimamente superior al azar, mientras que las demás técnicas, salvo GFS y PSO, resultan inaplicables en esta situación.

6.3.6 Molecular Biology (Splice-junction Gene Sequences):

Splice	Atrib. Utilizados	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Sin Selección	60	0,8304	0,8304	0,8303	0,9388	158,33
Lasso	13	0,7103	0,7103	0,6141	0,8135	51
Ridge	46	0,8298	0,8298	0,83	0,9384	111,88
Elastic Net	18	0,7727	0,7727	0,7692	0,9024	57,29
GFS	20	0,684	0,684	0,5904	0,795	79,81
PSO	29	0,7721	0,7721	0,7683	0,8985	90,02

En la base de datos Splice, el modelo SVM sin aplicar técnicas de selección de atributos presenta un rendimiento sólido. Utilizando los 60 atributos originales, alcanza una accuracy de 0,8304, un F1-score prácticamente idéntico y un AUC-ROC de 0,9388. Estos resultados indican que el modelo, en su versión completa, es altamente eficaz para la tarea de clasificación, aunque el tiempo de ejecución es considerablemente alto (158,33 segundos).

Ridge se aproxima mucho a ese rendimiento utilizando 46 atributos. Con una accuracy de 0,8298 y un AUC-ROC de 0,9384, mantiene las métricas casi intactas respecto al modelo sin selección, con una reducción moderada en el número de atributos y una mejora significativa en el tiempo de ejecución. Esto indica que Ridge logra un buen equilibrio entre rendimiento y eficiencia computacional.

Elastic Net y PSO también ofrecen resultados razonables. Elastic Net selecciona 18 atributos y obtiene una accuracy de 0,7727 y un AUC-ROC de 0,9024. PSO, con 29 atributos, presenta métricas casi idénticas a Elastic Net, con una ligera ventaja en F1-score y AUC-ROC. Ambos métodos permiten reducir la complejidad del modelo y el tiempo de ejecución, a cambio de una pérdida controlada en las métricas de rendimiento.

Lasso, en cambio, muestra una caída más severa en el rendimiento. Con solo 13 atributos seleccionados, la accuracy desciende a 0,7103 y el F1-score a 0,6141, lo que indica que la reducción ha sido demasiado agresiva, eliminando variables clave para la clasificación. GFS muestra el peor desempeño entre todas las técnicas, con una accuracy de 0,684 y un F1-score de 0,5904 utilizando 20 atributos, lo que sugiere una selección poco efectiva para esta base de datos.

En conjunto, el modelo sin selección sigue siendo el que alcanza el mejor rendimiento global, mientras que Ridge se posiciona como la técnica que logra mantener ese alto nivel reduciendo parcialmente la cantidad de atributos y el tiempo de cómputo. Esto la convierte en una alternativa eficiente para esta base de datos, sin comprometer la calidad de la clasificación.

6.3.7 Parkinson's Disease Classification:

Parkdis	Atrib. Utilizados	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Sin Selección	389	0,8175	0,8175	0,8185	0,8434	8,25
Lasso	7	0,8359	0,8359	0,8181	0,8353	2,09
Ridge	319	0,8373	0,8373	0,8371	0,8736	7,1
Elastic Net	17	0,8544	0,8544	0,8422	0,871	2,41
GFS	201	0,8241	0,8241	0,8249	0,8606	11,22
PSO	202	0,7896	0,7896	0,7911	0,7971	15,01

En la base de datos Parkdis, el modelo SVM sin aplicar selección de atributos ofrece un rendimiento bueno, con una accuracy de 0,8175 y un F1-score de 0,8185, utilizando los 389 atributos originales. A pesar del gran número de variables, el tiempo de ejecución es relativamente bajo (8,25 segundos), lo que sugiere que el modelo puede gestionarse eficientemente incluso con alta dimensionalidad.

Varias técnicas de selección de atributos logran no solo mantener, sino mejorar el rendimiento del modelo. Elastic Net se destaca claramente al alcanzar la mayor accuracy (0,8544) y el mayor F1-score (0,8422), utilizando solo 17 atributos. También logra un AUC-ROC de 0,871, muy superior al modelo completo. Esta combinación de alto rendimiento, fuerte reducción en la cantidad de atributos y tiempo de ejecución bajo (2,41 segundos) convierte a Elastic Net en la técnica más eficaz para esta base de datos.

Ridge también obtiene resultados sobresalientes, con una accuracy de 0,8373 y un AUC-ROC de 0,8736, aunque utiliza 319 atributos. Si bien mejora el rendimiento respecto al modelo sin selección, lo hace con una reducción marginal en la cantidad de variables, por lo que no ofrece una simplificación significativa del modelo.

Lasso, en contraste, selecciona solo 7 atributos y logra una accuracy de 0,8359, muy cercana a la de Ridge y con un F1-score (0,8181) casi idéntico al modelo sin selección. Sin embargo, su AUC-ROC es menor (0,8353), lo que sugiere una menor capacidad para distinguir entre clases. Aun así, su rendimiento es notable considerando la drástica reducción de atributos.

GFS mantiene métricas similares al modelo completo con 201 atributos seleccionados. Su accuracy es de 0,8241 y su F1-score de 0,8249, con una ligera mejora sobre el modelo base. Sin embargo, su tiempo de ejecución es mayor (11,22 segundos), lo que reduce su eficiencia relativa.

PSO es la técnica con peor desempeño. A pesar de seleccionar un número comparable de atributos a GFS (202), su accuracy desciende a 0,7896 y el AUC-ROC cae a 0,7971, lo que indica una pérdida clara de rendimiento sin ventajas en el tiempo de ejecución.

En conjunto, Elastic Net es la técnica más destacada para esta base de datos, al lograr el mejor rendimiento con una fuerte reducción en la cantidad de atributos y un tiempo de ejecución muy reducido.

6.3.8 Statlog (German Credit Data):

Statlog	Atrib. Utilizados	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Sin Selección	20	0,758	0,758	0,7447	0,7773	8,47
Lasso	1	-	-	-	-	-
Ridge	18	0,758	0,758	0,7445	0,782	6,84
Elastic Net	4	0,733	0,733	0,699	0,7709	3,56
GFS	7	0,7	0,7	0,5765	0,5591	3,34
PSO	8	0,701	0,701	0,604	0,737	4,21

En la base de datos Statlog, el modelo SVM sin aplicar técnicas de selección de atributos presenta un rendimiento aceptable, con una accuracy de 0,758, un F1-score de 0,7447 y un AUC-ROC de 0,7773, utilizando los 20 atributos originales. Estos resultados reflejan un equilibrio razonable entre precisión y capacidad de discriminación del modelo, aunque con un tiempo de ejecución moderado de 8,47 segundos.

Ridge mantiene exactamente la misma accuracy que el modelo completo (0,758) y un F1-score casi idéntico (0,7445), pero reduce ligeramente el número de atributos a 18 y mejora levemente el AUC-ROC a 0,782. Esta técnica conserva el rendimiento al tiempo que mejora la eficiencia, siendo además más rápida en su ejecución (6,84 segundos), lo que la posiciona como la alternativa más estable respecto al modelo sin selección.

Elastic Net logra una reducción más pronunciada en el número de atributos, seleccionando solo 4, pero con una disminución en el rendimiento. Su accuracy baja a 0,733 y el F1-score cae a 0,699. A pesar de eso, el AUC-ROC se mantiene relativamente alto (0,7709), y el tiempo de ejecución es el menor entre las técnicas aplicadas (3,56 segundos). Esto indica que Elastic Net permite un modelo mucho más compacto, con una ligera pérdida de precisión, pero conservando capacidad discriminativa.

Lasso no produce un modelo evaluable, ya que selecciona solo 1 atributo frente a un problema de clasificación binaria, lo que impide una clasificación efectiva. Esta selección excesivamente restrictiva invalida su uso en este caso concreto.

Las técnicas metaheurísticas, GFS y PSO, presentan los peores resultados. GFS, con 7 atributos, alcanza una accuracy de solo 0,7 y un F1-score de 0,5765, mientras que el AUC-ROC se reduce drásticamente a 0,5591, lo que refleja una clara pérdida de capacidad predictiva. PSO mejora ligeramente respecto a GFS, con una accuracy de 0,701 y un F1-score de 0,604, pero también queda por debajo del rendimiento del modelo base y de otras técnicas más consistentes.

En conjunto, Ridge es la técnica que mejor conserva el rendimiento del modelo completo, con una reducción mínima de atributos y tiempos más eficientes. Aunque Elastic Net logra una mayor simplificación, lo hace a costa de una ligera pérdida de precisión.

6.3.9 Musk (Version 1):

Musk	Atrib. Utilizados	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Sin Selección	98	0,8508	0,8508	0,8506	0,9129	5,14
Lasso	6	0,7566	0,7566	0,752	0,8087	1,15
Ridge	92	0,8508	0,8508	0,8508	0,9134	4,66
Elastic Net	18	0,7964	0,7964	0,7962	0,8852	1,5
GFS	46	0,809	0,809	0,8075	0,89	3,32
PSO	38	0,767	0,767	0,7649	0,8465	3,39

En la base de datos Musk, el modelo SVM sin aplicar selección de atributos muestra un rendimiento elevado, con una accuracy de 0,8508, un F1-score prácticamente igual (0,8506) y un AUC-ROC de 0,9129, utilizando los 98 atributos originales. El tiempo de ejecución, de 5,14 segundos, es razonable considerando la alta dimensionalidad del conjunto de datos.

Ridge reproduce casi exactamente el mismo rendimiento que el modelo sin selección: misma accuracy y F1-score, y un AUC-ROC incluso levemente superior (0,9134), utilizando 92 atributos. Esta ligera reducción de características y menor tiempo de ejecución (4,66 segundos) lo convierten en una opción eficiente cuando se busca mantener el rendimiento sin cambios significativos.

Elastic Net reduce considerablemente el número de atributos a solo 18, con una pérdida moderada de rendimiento. La accuracy cae a 0,7964 y el AUC-ROC a 0,8852, lo que indica que esta técnica logra una simplificación significativa del modelo, sacrificando parte del poder predictivo, aunque manteniéndose dentro de un rango aceptable.

Lasso, en cambio, selecciona solo 6 atributos, pero con un descenso mucho más acusado en las métricas. La accuracy baja a 0,7566 y el AUC-ROC a 0,8087, lo que sugiere que la reducción fue demasiado agresiva y resultó en una eliminación de variables relevantes para la clasificación.

Las técnicas metaheurísticas GFS y PSO obtienen resultados intermedios. GFS selecciona 46 atributos y logra una accuracy de 0,809 con un AUC-ROC de 0,89, lo que muestra un equilibrio moderado entre reducción de atributos y

rendimiento. PSO, con 38 atributos, alcanza una accuracy de 0,767 y un AUC-ROC de 0,8465, algo inferior a GFS en todos los aspectos.

En conjunto, el modelo sin selección sigue siendo el que ofrece el mejor rendimiento absoluto, mientras que Ridge se posiciona como la técnica más eficaz si se desea mantener la precisión del modelo con una pequeña mejora en eficiencia. Elastic Net representa una alternativa válida cuando se busca una mayor reducción de atributos, aceptando una pérdida moderada en la calidad del modelo.

6.3.10 Mice Protein Expression:

Mice	Atrib. Utilizados	Accuracy	Recall	F1-Score	AUC-ROC	Tiempo (s)
Sin Selección	73	1	1	1	1	4,88
Lasso	3	-	-	-	-	-
Ridge	3	-	-	-	-	-
Elastic Net	9	1	1	1	1	1,98
GFS	46	0,9704	0,9704	0,9703	0,9986	4,84
PSO	36	0,9509	0,9509	0,9506	0,9974	4,72

En la base de datos Mice, el modelo SVM sin aplicar selección de atributos alcanza un rendimiento perfecto: accuracy, recall, F1-score y AUC-ROC igual a 1, utilizando los 73 atributos disponibles. Este resultado indica que el modelo es capaz de clasificar con precisión absoluta en este conjunto de datos, lo que lo convierte en una referencia de máximo rendimiento.

Elastic Net logra mantener ese mismo nivel de rendimiento perfecto utilizando solo 9 atributos, con tiempos de ejecución significativamente menores (1,98 segundos frente a los 4,88 del modelo sin selección). Esto sugiere que Elastic Net identifica con precisión un subconjunto reducido de variables altamente representativas, lo que permite mantener la calidad del modelo con una simplificación sustancial.

Lasso y Ridge no arrojan resultados válidos, ya que seleccionan solo 3 atributos frente a un problema de 8 clases, lo que impide la correcta clasificación. Esto evidencia una selección demasiado agresiva, que no garantiza una cobertura suficiente de la complejidad del problema multiclase.

Las técnicas metaheurísticas, GFS y PSO, sí permiten modelos funcionales, aunque con una ligera pérdida de rendimiento. GFS selecciona 46 atributos y obtiene una accuracy de 0,9704 y un AUC-ROC de 0,9986, mientras que PSO, con 36 atributos, alcanza una accuracy de 0,9509 y un AUC-ROC de 0,9974. Aunque los resultados siguen siendo muy altos, reflejan una pequeña degradación respecto al modelo original, a cambio de una reducción considerable del número de atributos.

En este caso, el modelo sin selección alcanza el rendimiento máximo posible, pero Elastic Net destaca como la mejor técnica de selección al replicar ese mismo rendimiento utilizando solo una fracción de los atributos. Esta combinación de eficacia y eficiencia la convierte en la opción más favorable para esta base de datos.

7. Conclusiones

Este trabajo ha llevado a cabo un análisis exhaustivo sobre la aplicación de técnicas avanzadas de selección de atributos para la clasificación, con el objetivo de evaluar su impacto sobre el rendimiento predictivo y la eficiencia computacional de los modelos. Se han utilizado cinco técnicas de selección (Lasso, Ridge, Elastic Net, PSO y GFS), aplicadas sobre diez bases de datos de distintas naturalezas y características —*Divorce, WDBC, Ionosp, Studentdrop, HCV, Splice, Parkdis, Statlog, Musk* y *Mice*—, utilizando el clasificador SVM como modelo común de evaluación. Las métricas empleadas han sido accuracy, recall, F1-score, AUC-ROC, número de atributos seleccionados y tiempo de ejecución.

Los resultados muestran que la selección de atributos no solo es útil, sino a menudo necesaria, especialmente en conjuntos de datos de alta dimensionalidad o con redundancia estructural. En bases como *Divorce*, *WDBC* o *Parkdis*, técnicas como Lasso y Elastic Net no solo redujeron drásticamente la cantidad de atributos, sino que también mejoraron o mantuvieron las métricas de rendimiento respecto al modelo sin selección. Esto confirma que una reducción bien ejecutada puede mejorar la capacidad de generalización del modelo al eliminar ruido e irrelevancia.

En términos de estabilidad, Ridge se mostró como una técnica especialmente robusta: logró mantener un rendimiento casi idéntico al del modelo completo en bases como *Statlog*, *Splice* o *Musk*, aunque sin reducir drásticamente la cantidad de atributos. Su comportamiento conservador la convierte en una herramienta fiable cuando se desea preservar la estructura del conjunto de datos sin comprometer las métricas. Elastic Net, en cambio, logró un mayor equilibrio: fue capaz de reducir atributos de forma significativa manteniendo un rendimiento competitivo en diversas bases, y en algunos casos, como *Mice*, incluso replicando resultados perfectos con menos del 15% de los atributos originales.

Las técnicas metaheurísticas GFS y PSO ofrecieron resultados más inestables. En algunas bases, como *Musk* o *Splice*, GFS logró resultados cercanos al modelo completo con una reducción moderada. Sin embargo, en otras bases como *Studentdrop*, *Ionosp* o especialmente *HCV*, ambas técnicas mostraron una caída considerable en las métricas. Este comportamiento refuerza la idea de que, si bien los algoritmos bioinspirados pueden ser útiles en contextos exploratorios o no lineales, su rendimiento es altamente dependiente del problema específico y, en muchos casos, requiere un ajuste fino de parámetros para ofrecer resultados competitivos.

Casos particulares como *Mice* o *HCV* merecen una mención especial. En *Mice*, Elastic Net fue capaz de mantener un rendimiento perfecto con solo nueve atributos frente a ocho clases. Aunque los resultados son impresionantes, se recomienda una validación más profunda para descartar posibles problemas de sobreajuste o estructuras internas favorables. En *HCV*, ninguna técnica logró un rendimiento aceptable, lo que sugiere que la dificultad no radica solo en la elección de atributos, sino en la naturaleza misma del conjunto de datos, posiblemente influenciada por desbalance de clases, ruido o falta de separabilidad entre categorías.

Desde un enfoque más amplio, este trabajo permite extraer una conclusión fundamental: la selección de atributos no debe considerarse una etapa secundaria o complementaria, sino un componente esencial en el diseño de modelos predictivos sólidos. Su correcta implementación permite no solo reducir tiempos y recursos computacionales, sino también facilitar la interpretación del modelo, mejorar su capacidad de generalización y revelar estructuras ocultas en los datos.

Además, los resultados ponen de manifiesto que no existe una técnica universalmente óptima: cada algoritmo de selección responde de forma diferente según las características del conjunto de datos. Esta heterogeneidad refuerza la necesidad de evaluar de forma empírica múltiples enfoques y no adoptar una única técnica por defecto.

8. Recomendaciones

A partir de los resultados y conclusiones obtenidos en este trabajo, se proponen una serie de recomendaciones prácticas y orientaciones para investigaciones futuras que puedan profundizar en el estudio de la selección de atributos para clasificación:

1. Evaluar siempre múltiples técnicas de selección de atributos.
Como se ha evidenciado, no existe una técnica que funcione de forma óptima para todas las bases de datos. Por ello, se recomienda comparar al menos dos enfoques distintos (por ejemplo, uno basado en regularización como Elastic Net y otro bioinspirado como GFS o PSO) para cada nuevo problema de clasificación.

2. No subestimar el modelo sin selección, pero tampoco adoptarlo como referencia única.
Si bien en algunos casos el modelo sin reducción ofrece buenos resultados, puede implicar mayor complejidad, tiempos más altos y menor interpretabilidad. Por tanto, debe considerarse como punto de partida, pero no como solución definitiva.

3. Validar cuidadosamente los modelos simplificados.
Técnicas como Elastic Net o Lasso pueden ofrecer altos rendimientos con muy pocos atributos. Sin embargo, es crucial asegurar que estos modelos no estén sobreajustados, especialmente en problemas multiclase o con clases desbalanceadas. Se recomienda utilizar validación cruzada, análisis de varianza, y en lo posible, conjuntos de prueba externos.

4. Incorporar la selección de atributos desde el diseño del pipeline de modelado.
La reducción de atributos no debe verse como una fase posterior o accesoria, sino como una etapa integrada al proceso de modelado. Esto permite construir modelos más eficientes desde el inicio, optimizando tanto recursos como interpretabilidad.

5. Ajustar parámetros específicos por base de datos.

Especialmente en el caso de algoritmos metaheurísticos, como PSO y GFS, es recomendable ajustar los hiperparámetros (número de iteraciones, tamaño del enjambre o población, funciones de coste, etc.) según la naturaleza del problema. El uso de valores por defecto puede limitar notablemente su rendimiento.

6. Ampliar el estudio a otras técnicas y contextos.

Una línea futura de investigación relevante consiste en comparar estas técnicas de selección con otras basadas en árboles (como Random Forest o XGBoost feature importance), métodos wrapper (como RFE), o enfoques basados en aprendizaje profundo (autoencoders, embeddings). También sería útil evaluar su desempeño con otros clasificadores, como k-NN, redes neuronales o árboles de decisión.

7. Aplicar estas técnicas en entornos reales y conjuntos de datos ruidosos o desbalanceados.

Las técnicas analizadas muestran gran utilidad en contextos controlados, pero su desempeño en situaciones reales —con ruido, clases desbalanceadas o atributos faltantes— debe explorarse más a fondo. Estudiar su aplicabilidad en sectores como la medicina, la industria agroalimentaria o la educación puede aportar una validación externa y aplicada.

8. Incorporar métricas adicionales en el análisis comparativo.

Además de las métricas clásicas utilizadas (Accuracy, Recall, F1-score, AUC-ROC), futuras investigaciones pueden incorporar métricas de interpretabilidad, estabilidad de selección y coste computacional global, para ofrecer una visión más integral del proceso.

9. Referencias

1. Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3(Mar), 1157-1182.
2. Jović, A., Brkić, K., & Bogunović, N. (2015). A review of feature selection methods with applications. In 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 1200-1205). IEEE.
3. Liu, H., & Motoda, H. (1998). Feature selection for knowledge discovery and data mining. The Springer International Series in Engineering and Computer Science (Vol. 454). Springer.
4. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics.
5. Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. Computers & Electrical Engineering, 40(1), 16-28.
6. Pan, X., Wu, Y., & Qian, Z. (2019). Probability-based particle swarm optimization for feature selection. Applied Soft Computing, 81, 105509.
7. Dorigo, M., & Birattari, M. (2010). Ant colony optimization. In Encyclopedia of Machine Learning (pp. 36-39). Springer, Boston, MA.
8. Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301-320.
9. Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. In Proceedings of the twenty-first international conference on Machine learning (p. 78). ACM.
10. Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288.
11. Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 55-67.

10. Anexos

10.1. Código en Python

```
# ==========================================
# 1. IMPORTACIÓN DE LIBRERÍAS (ÚNICO BLOQUE GLOBAL)
# ==========================================

# Manejo de datos
import pandas as pd
import numpy as np
import os
import time
import warnings

# Clasificación y validación
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.exceptions import DataConversionWarning

# Desactivar advertencias de conversión innecesarias
warnings.filterwarnings(action='ignore', category=DataConversionWarning)
```

Lectura y Preprocesamiento de las Bases de Datos

```
# ==========================================
# 2. PREPROCESAMIENTO DE LAS BASES DE DATOS
# ==========================================

from ucimlrepo import fetch_ucirepo

# ------------------------------------------
# 2.1 Carga y transformación de datos desde UCI o archivos locales
# ------------------------------------------

# 1. Divorce
divorce = pd.read_excel(r"D:\TFM_UNIA\Bases de
datos\divorce+predictors+data+set\divorce\divorce.xlsx")
X_divorce = divorce.drop('Class', axis=1)
y_divorce_array = divorce['Class'].values.ravel()

# 2. WDBC (Breast Cancer)
wdbc = fetch_ucirepo(id=17)
X_wdbc = wdbc.data.features
y_wdbc = wdbc.data.targets.copy()
y_wdbc['Diagnosis'] = y_wdbc['Diagnosis'].replace({'M': 0, 'B': 1})
y_wdbc_array = y_wdbc.values.ravel()

# 3. HCV
hcv = fetch_ucirepo(id=503)
X_hcv = hcv.data.features
y_hcv_array = hcv.data.targets.values.ravel()
```

```
# 4. Ionosphere
ionosp = fetch_ucirepo(id=52)
X_ionosp = ionosp.data.features
y_ionosp = ionosp.data.targets.copy()
y_ionosp['Class'] = y_ionosp['Class'].replace({'g': 1, 'b': 0})
y_ionosp_array = y_ionosp.values.ravel()

# 5. Splice
splice = fetch_ucirepo(id=69)
X_splice_df = pd.DataFrame(splice.data.features)
y_splice = splice.data.targets.copy()
y_splice['class'] = y_splice['class'].replace({'EI': 1, 'IE': 2, 'N': 0})
y_splice_array = y_splice.values.ravel()

# Codificación de letras en X_splice
cod_dict = {'A': 1, 'G': 2, 'T': 3, 'C': 4, 'D': 5, 'N': 6, 'S': 7, 'R': 8}
X_splice_encoded = X_splice_df.applymap(lambda x: cod_dict.get(x, -1))

# 6. Parkinson's Disease
parkdis = pd.read_csv(r"D:\TFM_UNIA\Bases de
datos\parkinson+s+disease+classification\pd_speech_features.csv")
parkdis.columns = parkdis.iloc[0]
parkdis = parkdis.drop(0).reset_index(drop=True)
X_parkdis = parkdis.drop(columns=['id', 'class']).apply(pd.to_numeric,
errors='coerce')
y_parkdis_array = parkdis['class'].astype(int).values.ravel()

# 7. Student Dropout
studentdrop = fetch_ucirepo(id=697)
X_studentdrop = studentdrop.data.features
y_studentdrop = studentdrop.data.targets.replace({'Target': {'Dropout': 0, 'Enrolled':
1, 'Graduate': 2}})
y_studentdrop_array = y_studentdrop.values.ravel()

# 8. Statlog German Credit Data
statlog = fetch_ucirepo(id=144)
X_statlog = statlog.data.features.copy()
y_statlog_array = statlog.data.targets.values.ravel()
categorical_cols = X_statlog.select_dtypes(include='object').columns
for col in categorical_cols:
    X_statlog[col] = LabelEncoder().fit_transform(X_statlog[col])

# 9. Musk (versión 1)
musk = fetch_ucirepo(id=74)
X_musk = musk.data.features.drop(columns=['molecule_name',
'conformation_name'])
y_musk_array = musk.data.targets.values.ravel()

# 10. Mice Protein Expression
mice = fetch_ucirepo(id=342)
X_mice = mice.data.features.copy()
y_mice = mice.data.targets.copy()

# Codificar columnas categóricas si las hay
non_numeric_cols = X_mice.select_dtypes(include='object').columns
```

```
for col in non_numeric_cols:
    X_mice[col] = LabelEncoder().fit_transform(X_mice[col])

# Imputar valores faltantes
imputer = SimpleImputer(strategy='mean')
X_mice_imputed = pd.DataFrame(imputer.fit_transform(X_mice),
columns=X_mice.columns)

# Codificar clase
y_mice['class'] = LabelEncoder().fit_transform(y_mice['class'])
y_mice_array = y_mice.values.ravel()
# ========================================================
# 2.2 Normalización, estandarización y eliminación de correlación
# ========================================================

# Función para normalizar (Min-Max) y estandarizar (Z-score) los datos
def normalize_and_standardize(X):
    min_max_scaler = MinMaxScaler()
    X_normalized = pd.DataFrame(min_max_scaler.fit_transform(X),
columns=X.columns)

    standard_scaler = StandardScaler()
    X_standardized = pd.DataFrame(standard_scaler.fit_transform(X),
columns=X.columns)

    return X_normalized, X_standardized

# Función para eliminar atributos altamente correlacionados (correlación > 0.9)
def remove_highly_correlated_features(X, threshold=0.9):
    corr_matrix = X.corr().abs()
    upper_triangle = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(bool))
    to_drop = [col for col in upper_triangle.columns if any(upper_triangle[col] >
threshold)]
    X_reduced = X.drop(columns=to_drop)
    return X_reduced, to_drop

# Aplicación del preprocesamiento a cada base de datos

X_divorce_normalized, X_divorce_standardized =
normalize_and_standardize(X_divorce)
X_divorce_reduced, drop_divorce =
remove_highly_correlated_features(X_divorce_standardized)

X_wdbc_normalized, X_wdbc_standardized = normalize_and_standardize(X_wdbc)
X_wdbc_reduced, drop_wdbc =
remove_highly_correlated_features(X_wdbc_standardized)

X_hcv_normalized, X_hcv_standardized = normalize_and_standardize(X_hcv)
X_hcv_reduced, drop_hcv =
remove_highly_correlated_features(X_hcv_standardized)

X_ionosp_normalized, X_ionosp_standardized =
normalize_and_standardize(X_ionosp)
```

```
X_ionosp_reduced, drop_ionosp =
remove_highly_correlated_features(X_ionosp_standardized)

X_splice_normalized, X_splice_standardized =
normalize_and_standardize(X_splice_encoded)
X_splice_reduced, drop_splice =
remove_highly_correlated_features(X_splice_standardized)

X_parkdis_normalized, X_parkdis_standardized =
normalize_and_standardize(X_parkdis)
X_parkdis_reduced, drop_parkdis =
remove_highly_correlated_features(X_parkdis_standardized)

X_studentdrop_normalized, X_studentdrop_standardized =
normalize_and_standardize(X_studentdrop)
X_studentdrop_reduced, drop_studentdrop =
remove_highly_correlated_features(X_studentdrop_standardized)

X_statlog_normalized, X_statlog_standardized =
normalize_and_standardize(X_statlog)
X_statlog_reduced, drop_statlog =
remove_highly_correlated_features(X_statlog_standardized)

X_musk_normalized, X_musk_standardized = normalize_and_standardize(X_musk)
X_musk_reduced, drop_musk =
remove_highly_correlated_features(X_musk_standardized)

X_mice_normalized, X_mice_standardized =
normalize_and_standardize(X_mice_imputed)
X_mice_reduced, drop_mice =
remove_highly_correlated_features(X_mice_standardized)
```

Selección de Atributos

```
# ================================================

# SELECCIÓN DE ATRIBUTOS CON LASSO, RIDGE Y ELASTIC NET

# ================================================


# Crear carpeta de salida si no existe

output_folder = "atributos_seleccionados"

os.makedirs(output_folder, exist_ok=True)


# Función para aplicar Lasso, Ridge y Elastic Net y guardar X seleccionada + y

def apply_regularization_methods_and_save(X, y, dataset_name):

    results = {}

    y_df = pd.DataFrame(y, columns=["Class"])
```

```
    # 1. Lasso
    start_time = time.time()
    lasso = Lasso(alpha=0.1, max_iter=50000)
    lasso.fit(X, y)
    lasso_time = time.time() - start_time
    lasso_selected_features = [i for i, coef in enumerate(lasso.coef_) if coef != 0]
    df_lasso = pd.concat([X.iloc[:, lasso_selected_features], y_df], axis=1)
    df_lasso.to_csv(f"{output_folder}/{dataset_name}_lasso.csv", index=False)
    results['Lasso'] = {'features': lasso_selected_features, 'count':
len(lasso_selected_features), 'time': lasso_time}


    # 2. Ridge
    start_time = time.time()
    ridge = Ridge(alpha=0.1, max_iter=50000)
    ridge.fit(X, y)
    ridge_time = time.time() - start_time
    ridge_selected_features = [i for i, coef in enumerate(ridge.coef_) if abs(coef) >
0.01]
    df_ridge = pd.concat([X.iloc[:, ridge_selected_features], y_df], axis=1)
    df_ridge.to_csv(f"{output_folder}/{dataset_name}_ridge.csv", index=False)
    results['Ridge'] = {'features': ridge_selected_features, 'count':
len(ridge_selected_features), 'time': ridge_time}


    # 3. Elastic Net
    start_time = time.time()
    elastic = ElasticNet(alpha=0.1, l1_ratio=0.5, max_iter=50000)
    elastic.fit(X, y)
    elastic_time = time.time() - start_time
    elastic_selected_features = [i for i, coef in enumerate(elastic.coef_) if coef != 0]
    df_elastic = pd.concat([X.iloc[:, elastic_selected_features], y_df], axis=1)
    df_elastic.to_csv(f"{output_folder}/{dataset_name}_elasticnet.csv", index=False)
    results['ElasticNet'] = {'features': elastic_selected_features, 'count':
len(elastic_selected_features), 'time': elastic_time}
```

```
    print(f"\nResultados para {dataset_name.upper()}:")
    for method, res in results.items():
        print(f"- {method}: {res['count']} atributos en {res['time']:.4f} s")

    return results

# Aplicar la función a todas las bases de datos
apply_regularization_methods_and_save(X_divorce_reduced, y_divorce_array, "divorce")
apply_regularization_methods_and_save(X_wdbc_reduced, y_wdbc_array, "wdbc")
apply_regularization_methods_and_save(X_ionosp_reduced, y_ionosp_array, "ionosp")
apply_regularization_methods_and_save(X_studentdrop_reduced, y_studentdrop_array, "studentdrop")
apply_regularization_methods_and_save(X_hcv_reduced, y_hcv_array, "hcv")
apply_regularization_methods_and_save(X_splice_reduced, y_splice_array, "splice")
apply_regularization_methods_and_save(X_parkdis_reduced, y_parkdis_array, "parkdis")
apply_regularization_methods_and_save(X_statlog_reduced, y_statlog_array, "statlog")
apply_regularization_methods_and_save(X_musk_reduced, y_musk_array, "musk")
apply_regularization_methods_and_save(X_mice_reduced, y_mice_array, "mice")
# ==========================================
# SELECCIÓN DE ATRIBUTOS CON GFS (Golden Fish Search)
# ==========================================
# Carpeta de salida para atributos seleccionados
output_dir = "atributos_seleccionados_gfs"
os.makedirs(output_dir, exist_ok=True)

# Función de aptitud (fitness) basada en índice Silhouette negativo
def fitness_function_gfs(feature_subset, X):
    selected = [i for i, val in enumerate(feature_subset) if val == 1]
    if len(selected) == 0:
        return 1  # Penaliza si no se selecciona ningún atributo
    try:
```

```
        labels = KMeans(n_clusters=3, random_state=42).fit_predict(X[:, selected])
        return -silhouette_score(X[:, selected], labels)
    except:
        return 1

# Algoritmo GFS básico
def golden_fish_search(X, dimensions, max_iters=100):
    best_cost = float('inf')
    best_position = np.zeros(dimensions)

    for _ in range(max_iters):
        position = np.random.randint(2, size=dimensions)
        cost = fitness_function_gfs(position, X)
        if cost < best_cost:
            best_cost = cost
            best_position = position

    return best_cost, best_position

# Aplicación del GFS a múltiples bases de datos
def apply_gfs_to_all_datasets(datasets):
    resultados = []

    for name, (X, y) in datasets.items():
        print(f"\nAplicando GFS a: {name}")
        X_np = X.values if isinstance(X, pd.DataFrame) else X
        dimensions = X_np.shape[1]

        start = time.time()
        best_cost, best_position = golden_fish_search(X_np, dimensions)
        tiempo = round(time.time() - start, 2)

        selected_indices = [i for i, v in enumerate(best_position) if v == 1]
```

```
        X_selected = X.iloc[:, selected_indices] if isinstance(X, pd.DataFrame) else
pd.DataFrame(X_np[:, selected_indices])
        df_with_y = pd.concat([X_selected.reset_index(drop=True), pd.DataFrame(y,
columns=["class"])], axis=1)
        df_with_y.to_csv(f"{output_dir}/{name}_gfs.csv", index=False)


        resultados.append({
            "Base de Datos": name,
            "Atributos Seleccionados": len(selected_indices),
            "Total de Atributos": dimensions,
            "Reducción (%)": round(100 * (1 - len(selected_indices) / dimensions), 2),
            "Índice Silueta": round(-best_cost, 4),
            "Tiempo (s)": tiempo
        })

        print(f"-> Atributos seleccionados: {len(selected_indices)}/{dimensions}")
        print(f"-> Índice Silueta: {-best_cost:.4f}")
        print(f"-> Tiempo: {tiempo} s")

    return pd.DataFrame(resultados)

# Diccionario limpio de datasets (sin absente ni onp)
datasets = {
    "divorce": (X_divorce_reduced, y_divorce_array),
    "wdbc": (X_wdbc_reduced, y_wdbc_array),
    "ionosp": (X_ionosp_reduced, y_ionosp_array),
    "studentdrop": (X_studentdrop_reduced, y_studentdrop_array),
    "hcv": (X_hcv_reduced, y_hcv_array),
    "splice": (X_splice_reduced, y_splice_array),
    "parkdis": (X_parkdis_reduced, y_parkdis_array),
    "statlog": (X_statlog_reduced, y_statlog_array),
    "musk": (X_musk_reduced, y_musk_array),
    "mice": (X_mice_reduced, y_mice_array)
}
```

```
# Ejecutar GFS y guardar resultados
resultados_gfs = apply_gfs_to_all_datasets(datasets)
print(resultados_gfs)
resultados_gfs.to_csv("resultados_gfs.csv", index=False)

# ==========================================
# SELECCIÓN DE ATRIBUTOS CON PSO (Particle Swarm Optimization)
# ==========================================

# Carpeta de salida para los atributos seleccionados por PSO
output_dir = "atributos_seleccionados_pso"
os.makedirs(output_dir, exist_ok=True)

# Función fitness basada en índice Silhouette negativo
def fitness_function_pso(position, X):
    selected = [i for i, p in enumerate(position) if p == 1]
    if len(selected) == 0:
        return 1  # Penaliza si no se selecciona ningún atributo
    try:
        labels = KMeans(n_clusters=3, random_state=42).fit_predict(X[:, selected])
        return -silhouette_score(X[:, selected], labels)
    except:
        return 1

# Algoritmo PSO para selección de atributos
def run_pso(X, num_particles=30, max_iter=80):
    n_features = X.shape[1]
    particles = np.random.randint(0, 2, (num_particles, n_features))
    velocities = np.zeros((num_particles, n_features))
    personal_best = particles.copy()
    personal_best_scores = np.array([fitness_function_pso(p, X) for p in particles])
    global_best = personal_best[np.argmin(personal_best_scores)]
```

```
    global_best_score = min(personal_best_scores)

    w, c1, c2 = 0.72, 1.49, 1.49

    for _ in range(max_iter):
        for i in range(num_particles):
            r1 = np.random.rand(n_features)
            r2 = np.random.rand(n_features)
            velocities[i] = (
                w * velocities[i]
                + c1 * r1 * (personal_best[i] - particles[i])
                + c2 * r2 * (global_best - particles[i])
            )
            sigmoid = 1 / (1 + np.exp(-velocities[i]))
            particles[i] = np.where(np.random.rand(n_features) < sigmoid, 1, 0)

            score = fitness_function_pso(particles[i], X)
            if score < personal_best_scores[i]:
                personal_best[i] = particles[i]
                personal_best_scores[i] = score
                if score < global_best_score:
                    global_best = particles[i]
                    global_best_score = score

    return global_best_score, global_best

# Diccionario de datasets preprocesados
datasets = {
    "divorce": (X_divorce_reduced, y_divorce_array),
    "wdbc": (X_wdbc_reduced, y_wdbc_array),
    "ionosp": (X_ionosp_reduced, y_ionosp_array),
    "studentdrop": (X_studentdrop_reduced, y_studentdrop_array),
    "hcv": (X_hcv_reduced, y_hcv_array),
```

```
    "splice": (X_splice_reduced, y_splice_array),
    "parkdis": (X_parkdis_reduced, y_parkdis_array),
    "statlog": (X_statlog_reduced, y_statlog_array),
    "musk": (X_musk_reduced, y_musk_array),
    "mice": (X_mice_reduced, y_mice_array)
}

# Aplicar PSO a todas las bases y guardar resultados
resultados = []

for name, (X_df, y) in datasets.items():
    try:
        print(f"\nAplicando PSO a: {name.upper()}")
        X_array = X_df.values
        y_array = np.array(y).ravel()

        start = time.time()
        score, best_position = run_pso(X_array)
        tiempo = round(time.time() - start, 2)

        selected_indices = [i for i, v in enumerate(best_position) if v == 1]
        df_selected = pd.DataFrame(X_array[:, selected_indices])
        df_selected["target"] = y_array
        df_selected.to_csv(f"{output_dir}/{name}_pso.csv", index=False)

        resultados.append({
            "Base de Datos": name,
            "Total Atributos": X_array.shape[1],
            "Seleccionados": len(selected_indices),
            "% Reducción": round(100 * (1 - len(selected_indices) / X_array.shape[1]),
2),
            "Índice Silueta": round(-score, 4),
            "Tiempo (s)": tiempo
        })
```

```
        print(f"   Atributos seleccionados: {len(selected_indices)}")
        print(f"   Índice Silueta: {-score:.4f}")
        print(f"   Tiempo de ejecución: {tiempo} s")
    except Exception as e:
        print(f"  Error procesando {name}: {e}")
# Mostrar resultados finales
df_resultados = pd.DataFrame(resultados)
print("\nResumen de Resultados PSO:")
print(df_resultados)
df_resultados.to_csv("resultados_svm_pso.csv", index=False)
```

Clasificación SVM tras selección de atributos

```
# ==========================================
# CLASIFICACIÓN CON SVM + LASSO
# ==========================================
selected_folder = r"C:\Users\danie\atributos_seleccionados"
nombres_bases = [
    "divorce", "wdbc", "ionosp", "studentdrop", "hcv",
    "splice", "parkdis", "statlog", "musk", "mice"
]

resultados = []
for name in nombres_bases:
    try:
        print(f"\nEvaluando SVM para {name.upper()} (Lasso)")
        path = os.path.join(selected_folder, f"{name}_lasso.csv")
        df = pd.read_csv(path)
        X = df.iloc[:, :-1].values
        y = df.iloc[:, -1].values
        if y.dtype == 'object':
            y = LabelEncoder().fit_transform(y)
        n_atributos = X.shape[1]
```

```
        n_clases = len(np.unique(y))
        if n_atributos <= n_clases:
            print(f" OMITIDO: {name.upper()} — solo {n_atributos} atributos frente a {n_clases} clases.")
            continue
        svm = SVC(kernel='linear', probability=True)
        cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

        start = time.time()
        acc = cross_val_score(svm, X, y, cv=cv, scoring='accuracy').mean()
        recall = cross_val_score(svm, X, y, cv=cv, scoring='recall_weighted').mean()
        f1 = cross_val_score(svm, X, y, cv=cv, scoring='f1_weighted').mean()
        auc = cross_val_score(svm, X, y, cv=cv, scoring='roc_auc_ovr_weighted').mean()
        end = time.time()
        resultados.append({
            "Base de Datos": name,
            "Accuracy": round(acc, 4),
            "Recall": round(recall, 4),
            "F1-Score": round(f1, 4),
            "AUC-ROC": round(auc, 4),
            "Tiempo (s)": round(end - start, 2)
        })
    except Exception as e:
        print(f"  Error procesando {name}: {e}")
df_resultados = pd.DataFrame(resultados)
print("\nResultados generales (SVM + Lasso):")
print(df_resultados)
df_resultados.to_csv("resultados_svm_lasso.csv", index=False)
# ==========================================
# CLASIFICACIÓN CON SVM + RIDGE
# ==========================================
selected_folder = r"C:\Users\danie\atributos_seleccionados"
nombres_bases = [
```

```
    "divorce", "wdbc", "ionosp", "studentdrop", "hcv",
    "splice", "parkdis", "statlog", "musk", "mice"
]
resultados = []
for name in nombres_bases:
    try:
        print(f"\nEvaluando SVM para {name.upper()} (Ridge)")
        path = os.path.join(selected_folder, f"{name}_ridge.csv")
        df = pd.read_csv(path)
        X = df.iloc[:, :-1].values
        y = df.iloc[:, -1].values
        if y.dtype == 'object':
            y = LabelEncoder().fit_transform(y)
        n_atributos = X.shape[1]
        n_clases = len(np.unique(y))
        if n_atributos == 0 or n_atributos <= n_clases:
            print(f" OMITIDO: {name.upper()} — {n_atributos} atributos frente a
{n_clases} clases.")
            continue


        svm = SVC(kernel='linear', probability=True)
        cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)


        start = time.time()
        acc = cross_val_score(svm, X, y, cv=cv, scoring='accuracy').mean()
        recall = cross_val_score(svm, X, y, cv=cv, scoring='recall_weighted').mean()
        f1 = cross_val_score(svm, X, y, cv=cv, scoring='f1_weighted').mean()
        auc = cross_val_score(svm, X, y, cv=cv,
scoring='roc_auc_ovr_weighted').mean()
        end = time.time()


        resultados.append({
            "Base de Datos": name,
            "Accuracy": round(acc, 4),
```

```
            "Recall": round(recall, 4),
            "F1-Score": round(f1, 4),
            "AUC-ROC": round(auc, 4),
            "Tiempo (s)": round(end - start, 2)
        })

    except Exception as e:
        print(f"  Error procesando {name}: {e}")

df_resultados = pd.DataFrame(resultados)
print("\nResultados generales (SVM + Ridge):")
print(df_resultados)
df_resultados.to_csv("resultados_svm_ridge.csv", index=False)
# ==========================================
# CLASIFICACIÓN CON SVM + ELASTIC NET
# ==========================================

selected_folder = r"C:\Users\danie\atributos_seleccionados"

nombres_bases = [
    "divorce", "wdbc", "ionosp", "studentdrop", "hcv",
    "splice", "parkdis", "statlog", "musk", "mice"
]

resultados = []

for name in nombres_bases:
    try:
        print(f"\nEvaluando SVM para {name.upper()} (Elastic Net)")

        path = os.path.join(selected_folder, f"{name}_elasticnet.csv")
        df = pd.read_csv(path)
        X = df.iloc[:, :-1].values
```

```
        y = df.iloc[:, -1].values

        if y.dtype == 'object':
            y = LabelEncoder().fit_transform(y)

        n_atributos = X.shape[1]
        n_clases = len(np.unique(y))
        if n_atributos == 0 or n_atributos <= n_clases:
            print(f" OMITIDO: {name.upper()} — {n_atributos} atributos frente a
{n_clases} clases.")
            continue

        svm = SVC(kernel='linear', probability=True)
        cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

        start = time.time()
        acc = cross_val_score(svm, X, y, cv=cv, scoring='accuracy').mean()
        recall = cross_val_score(svm, X, y, cv=cv, scoring='recall_weighted').mean()
        f1 = cross_val_score(svm, X, y, cv=cv, scoring='f1_weighted').mean()
        auc = cross_val_score(svm, X, y, cv=cv,
scoring='roc_auc_ovr_weighted').mean()
        end = time.time()

        resultados.append({
            "Base de Datos": name,
            "Accuracy": round(acc, 4),
            "Recall": round(recall, 4),
            "F1-Score": round(f1, 4),
            "AUC-ROC": round(auc, 4),
            "Tiempo (s)": round(end - start, 2)
        })

    except Exception as e:
        print(f"  Error procesando {name}: {e}")
```

```
df_resultados = pd.DataFrame(resultados)
print("\nResultados generales (SVM + Elastic Net):")
print(df_resultados)
df_resultados.to_csv("resultados_svm_elasticnet.csv", index=False)
# ==========================================
# CLASIFICACIÓN CON SVM + GFS
# ==========================================

selected_folder = r"C:\Users\danie\atributos_seleccionados_gfs"

nombres_bases = [
    "divorce", "wdbc", "ionosp", "studentdrop", "hcv",
    "splice", "parkdis", "statlog", "musk", "mice"
]

resultados = []

for name in nombres_bases:
    try:
        print(f"\nEvaluando SVM para {name.upper()} (GFS)")

        path = os.path.join(selected_folder, f"{name}_gfs.csv")
        df = pd.read_csv(path)
        X = df.iloc[:, :-1].values
        y = df.iloc[:, -1].values

        if y.dtype == 'object':
            y = LabelEncoder().fit_transform(y)

        n_atributos = X.shape[1]
        n_clases = len(np.unique(y))
        if n_atributos == 0 or n_atributos <= n_clases:
```

```
            print(f" OMITIDO: {name.upper()} — {n_atributos} atributos frente a
{n_clases} clases.")
            continue


        svm = SVC(kernel='linear', probability=True)
        cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)


        start = time.time()
        acc = cross_val_score(svm, X, y, cv=cv, scoring='accuracy').mean()
        recall = cross_val_score(svm, X, y, cv=cv, scoring='recall_weighted').mean()
        f1 = cross_val_score(svm, X, y, cv=cv, scoring='f1_weighted').mean()
        auc = cross_val_score(svm, X, y, cv=cv,
scoring='roc_auc_ovr_weighted').mean()
        end = time.time()


        resultados.append({
            "Base de Datos": name,
            "Accuracy": round(acc, 4),
            "Recall": round(recall, 4),
            "F1-Score": round(f1, 4),
            "AUC-ROC": round(auc, 4),
            "Tiempo (s)": round(end - start, 2)
        })


    except Exception as e:
        print(f"  Error procesando {name}: {e}")


df_resultados = pd.DataFrame(resultados)
print("\nResultados generales (SVM + GFS):")
print(df_resultados)
df_resultados.to_csv("resultados_svm_gfs.csv", index=False)
# ============================================
# CLASIFICACIÓN CON SVM + PSO
# ============================================
```

```
selected_folder = r"C:\Users\danie\atributos_seleccionados_pso"

nombres_bases = [
    "divorce", "wdbc", "ionosp", "studentdrop", "hcv",
    "splice", "parkdis", "statlog", "musk", "mice"
]

resultados = []

for name in nombres_bases:
    try:
        print(f"\nEvaluando SVM para {name.upper()} (PSO)")

        path = os.path.join(selected_folder, f"{name}_pso.csv")
        df = pd.read_csv(path)
        X = df.iloc[:, :-1].values
        y = df.iloc[:, -1].values

        if y.dtype == 'object':
            y = LabelEncoder().fit_transform(y)

        n_atributos = X.shape[1]
        n_clases = len(np.unique(y))
        if n_atributos == 0 or n_atributos <= n_clases:
            print(f" OMITIDO: {name.upper()} — {n_atributos} atributos frente a
{n_clases} clases.")
            continue

        svm = SVC(kernel='linear', probability=True)
        cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

        start = time.time()
        acc = cross_val_score(svm, X, y, cv=cv, scoring='accuracy').mean()
```

```
        recall = cross_val_score(svm, X, y, cv=cv, scoring='recall_weighted').mean()
        f1 = cross_val_score(svm, X, y, cv=cv, scoring='f1_weighted').mean()
        auc = cross_val_score(svm, X, y, cv=cv,
scoring='roc_auc_ovr_weighted').mean()
        end = time.time()

        resultados.append({
            "Base de Datos": name,
            "Accuracy": round(acc, 4),
            "Recall": round(recall, 4),
            "F1-Score": round(f1, 4),
            "AUC-ROC": round(auc, 4),
            "Tiempo (s)": round(end - start, 2)
        })

    except Exception as e:
        print(f"  Error procesando {name}: {e}")

df_resultados = pd.DataFrame(resultados)
print("\nResultados generales (SVM + PSO):")
print(df_resultados)
df_resultados.to_csv("resultados_svm_pso.csv", index=False)
```

Clasificación Con SVM sin selección de atributos

```
# ========================================
# CLASIFICACIÓN CON SVM (SIN SELECCIÓN DE ATRIBUTOS)
# ========================================

datasets = {
    "divorce": (X_divorce_reduced, y_divorce_array),
    "wdbc": (X_wdbc_reduced, y_wdbc_array),
    "ionosp": (X_ionosp_reduced, y_ionosp_array),
    "studentdrop": (X_studentdrop_reduced, y_studentdrop_array),
    "hcv": (X_hcv_reduced, y_hcv_array),
```

```
    "splice": (X_splice_reduced, y_splice_array),
    "parkdis": (X_parkdis_reduced, y_parkdis_array),
    "statlog": (X_statlog_reduced, y_statlog_array),
    "musk": (X_musk_reduced, y_musk_array),
    "mice": (X_mice_reduced, y_mice_array)
}

resultados = []

for name, (X, y) in datasets.items():
    try:
        print(f"\nEvaluando SVM para {name.upper()} (sin selección de atributos)")

        n_atributos = X.shape[1]
        n_clases = len(np.unique(y))
        if n_atributos <= n_clases:
            print(f" OMITIDO: {name.upper()} — {n_atributos} atributos frente a
{n_clases} clases.")
            continue

        svm = SVC(kernel='linear', probability=True)
        cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

        start = time.time()
        acc = cross_val_score(svm, X, y, cv=cv, scoring='accuracy').mean()
        recall = cross_val_score(svm, X, y, cv=cv, scoring='recall_weighted').mean()
        f1 = cross_val_score(svm, X, y, cv=cv, scoring='f1_weighted').mean()
        auc = cross_val_score(svm, X, y, cv=cv,
scoring='roc_auc_ovr_weighted').mean()
        end = time.time()

        resultados.append({
            "Base de Datos": name,
            "Accuracy": round(acc, 4),
```

```
            "Recall": round(recall, 4),
            "F1-Score": round(f1, 4),
            "AUC-ROC": round(auc, 4),
            "Tiempo (s)": round(end - start, 2)
        })

    except Exception as e:
        print(f"  Error procesando {name}: {e}")

df_resultados = pd.DataFrame(resultados)
print("\nResultados generales SVM (sin selección de atributos):")
print(df_resultados)
df_resultados.to_csv("resultados_svm_sin_seleccion.csv", index=False)
```