Metaheurísticas para el problema de Aprendizaje de Pesos en Características

Stefani Castellanos Erick Silva

Abstract

Dado un conjunto de datos, se desea conocer que atributo es más importante que otro al momento de distingüir la clase a la que pertenece una instancia de otra. Esta es la problemática que plantea Aprendizaje de Pesos en Características. Se procederá a resolver utilizando cuatro metaheurísticas diferentes inicializadas de manera aleatoria o con un algoritmo que aprovecha la información de los datos para guiar la búsqueda en el espacio: Relief. Los mejores resultados serán aquellos que tengan una mejor relación calidad-tiempo para cada conjunto de datos considerado.

Palabras claves: Aprendizaje de Pesos en Características, Metaheurísticas, *Machine Learning*.

Introducción

Una de las preocupaciones más grandes del hombre ha sido crear información, analizarla y permitir que perdure en el tiempo. Al entrar en la tecnológica esta sed de conocimiento se ha afianzado y ha crecido.

Actualmente, se maneja una gran cantidad de información, sin embargo, no toda es relevante para una aplicación especifica. Por esta razón ha surgido la necesidad de reducir el tamaño de los datos, pero ¿Cómo escoger la información de manera adecuada?. En Inteligencia Artificial y Minería de Datos esta ha sido una interrogante constante debido a que se desea evitar consumir demasiado tiempo procesando la información sin pérdida de precisión.

Reducir el tamaño de los datos es posible a través de los siguientes métodos:

- Seleccionando características del conjunto de datos, lo que reduce el tamaño de las columnas.
- Eliminando las instancias del conjunto de datos que aporte poca información.
- Determinando la importancia de las características, ya que proporciona información que permite guiar al clasificador y así, reducir el tiempo de procesamiento.[2]

Este trabajo se enfocará en esta última, este problema de optimización lleva el nombre de Aprendizaje de Pesos en Características. Usualmente, ponderar los atributos para mejorar la tasa de aciertos de un clasificador hace uso de Redes Neurales, sin embargo estas pueden consumir demasiado tiempo. Se utilizará un algoritmo *greedy* (ávido) denominado Relief, dos metaheurísticas de trayectoría y dos poblacionales, para compararlas y determinar cuál de ellos es más adecuado para el problema.

Es indispensable contar con varios conjuntos de datos que permitan realizar la fase experimental y así concretar los resultados. En este caso, fueron seleccionados cinco: Iris, Sonar, WDBC, Spambase y Waveform. El primero constituye un conjunto pequeño, los siguientes dos son medianos y los últimos son consideradas problemas con gran cantidad de datos.

Primeramente se realizará la descripción del problema en términos matemáticos y se explicará en que consiste el clasificador utilizado. Luego, se detallará el funcionamiento del algoritmo Relief y las metaheurísticas Iterated Local Search, Simulated Annealing, Scatter Search y Differential Evolution. Se ofrecerán resultados de acuerdos a los experimentos realizados para cada algoritmo con cada conunto de datos. Finalmente, se concluirá según lo analizado en la sección descrita anteriormente.

1 Descripción del problema

Antes de describir el problema APC, es necesario comprender en qué consiste un problema de clasificación; se dispone de un conjunto de posibles clases (C) y un conjunto de datos (X), en donde una instancia x_i es un vector previamente clasificado y de la forma:

$$x_i = \langle f_1, f_2, \ldots, f_n, c \rangle$$

en donde:

- f_i : el valor de cada característica.
- n : la cantidad de característcas.
- c : la clase a la que pertenece dicha instancia, con $c \in C$.
- $|X| = m$: cantidad de instancias.
- $|C| = p$: cantidad de clases.

Es común particionar X en dos subconjuntos que representen el conjunto de entrenamiento y el de prueba, X_e y X_p respectivamente. X_e es utilizado para que el algoritmo aprenda los parámetros que le permitan clasificar correctamente todas sus instancias y así, utilizar X_p para validar los resultados obtenidos.

El problema del Aprendizaje de Pesos en Características consiste en optimizar el rendimiento de un clasificador, a partir de la inclusión de pesos asociados a las características del conjunto de datos. Los pesos ponderan la relevancia de cada característica y modifican su valor al momento de calcular las distancias entre instancias, de tal forma que los clasificadores que se construyan a partir de estos, sean más certeros y/o más rápidos. En este caso en particular, el clasificador considerado será el *K-Nearest Neighbor* (K-vecinos más cercanos) con $K = 1$ (1-NN).

El algoritmo K-NN asume que todas las instancias corresponden a puntos en un espacio n-dimensional ($\Re^n$), en donde n es la cantidad de características del conjunto de datos. [8]. El algoritmo es el siguiente:

Algorithm 1: K-Nearest Neighbor

Input: Conjunto de entrenamiento X_e, Conjunto de prueba X_p, K
Output: Conjunto X_p clasificados según C

```
foreach x_i ∈ X_p do
    vecinos = [ ];
    foreach x_j ∈ X_e do
        d = distancia(x_i, x_j);
        agregar(x_j, d, vecinos);
    k_vecinos = seleccionar_cercanos(k, vecinos);
    clasificar(x_i, k_vecinos);
return X_p clasificado
```

El proceso de aprendizaje de este clasificador consiste en almacenar una tabla con las instancias correspondientes al X_e junto a la clase asociada a cada uno de ellos. Dado una instancia $x_i \in X_p$, se calcula su distancia a todas las otras que pertenecen al conjunto de entrenamiento y se escogen las k más cercanas [5]; usualmente se determina la proximidad entre dos ejemplos utilizando la distancia Euclideana. Finalmente, x_i se clasifica según la clase mayoritaria grupo y se retorna el conjunto de pruebas clasificado.

1.1 Función objetivo

Al ser APC un problema de optimización, es necesario establecer cuál es la función que se desea mejorar. En este caso, dadas las características del problema, resulta evidente que obtener una "buena" solución está fuertemente relacionado con la cantidad de instancias clasificadas correctamiente usando 1-NN. El objetivo es encontrar el mejor vector de pesos que permita máximizar la tasa de aciertos del clasificador 1-NN. Más específicamente:

$$\max\ tasa(\text{1-NN}(X, W)) = 100 \times \frac{aciertos(X)}{total(X)} \quad (1)$$

sujeto a:

$$w_i = [0, 1] \quad 1 \leq i \leq n$$

donde:

- $W = \langle w_1, \ldots, w_n \rangle$ es una solución al problema.
- 1-NN es el clasificador k-NN con $k = 1$ vecinos, generado a partir del conjunto de datos inicial.
- X es el conjunto de datos sobre el cual se evalúa el clasificador.
- *aciertos* es la cantidad de instancias de X clasificadas correctamente por 1-NN.
- *total* es la cantidad total de instancias de X.

1.2 Representación de la solución

La solución al problema de Aprendizaje de Pesos en Características viene dado por $W = \langle w_1, \ldots, w_n \rangle$, un vector de números reales de tamaño n (cantidad de características) en el que el valor de cada w_i define el peso que pondera a la característica f_i, es decir, representa que tan importante es para distinguir a que clase pertenece un ejemplo.

Cada w_i debe pertenecer al intervalo $[0, 1]$; los valores cercanos a 1 indican que la característica es más importante. En caso de que algún valor quede fuera de este intervalo, se debe normalizar, seleccionando el máximo valor del vector y dividiendo todos los valores entre dicho número.

2 Algoritmo Relief

Las metaheurísticas suelen ser inicializadas con soluciones generadas aleatoriamente, sin embargo, esto puede producir un retardo en la ejecución o desembocar en una convergencia prematura en un óptimo local, por lo que se requiere de un algoritmo eficiente que tome información del problema y guie las búsqueda de manera inteligente. Para el problema considerado en este artículo, existe tal heurística: Relief.

Relief, es un algoritmo de selección de características inspirado en el aprendizaje basado en instancias, detecta aquellos ejemplos que son estadísticamente

relevantes para el concepto objetivo en tiempo lineal ($\Theta(nmp)$). Utiliza un *threshold*, τ entre $0 \leq \tau \leq 1$, que codifica la relevancia de una característica en particular [7]; en esta versión será omitido este parámetro puesto que las metaheurísticas utilizan el vector de pesos y no el vector de relevancias que se obtiene con τ.

Relief utiliza la distancia euclideana n-dimensional para determinar el "amigo más cercano" y el "enemigo más cercano" de una instancia x_i. Se denomina a una instancia "amigo más cercano" o "near-hit" a la instancia que pertenezca a la misma clase de x_i y se encuentre a menor distancia. Se denomina a una instancia "enemigo más cercano" o "near-miss" a la instancia que pertenezca a una clase diferente a x_i y se encuentre a menor distancia [7].

Algorithm 2: Relief

```
Input: Conjunto de entrenamiento X_e
Output: Vector de pesos W
W = 0, 0 ..., 0
foreach x_i ∈ X_e do
    a = amigo_mas_cercano(x_i);
    e = enemigo_mas_cercano(x_i);

    /* Actualizar pesos de W */
    for i ... n do
        dif_amigo = diferencia(x_i, a);
        dif_enemigo = diferencia(x_i, e);
        w_i = w_i - dif_amigo + dif_enemigo;
normalizar(W);
return W
```

La diferencia entre el valor x_i y el amigo/enemigo más cercano está definido por:
- Para los atributos con valores numéricos:

$$\text{diferencia}(a, b) = (a - b)^2$$

- Para los atributos con valores nominales:

$$\text{diferencia}(a, b) = \begin{cases} 0 & \text{son el iguales} \\ 1 & \text{son diferentes} \end{cases}$$

3 Metaheurísticas

Las soluciones a muchos problemas de optimización son intratables, es decir, obtener la mejor respuesta podría tomar un tiempo potencialmente infinito, no ser resoluble o no se dispone de la capacidad computacional suficiente para resolverlo. Las metaheurísticas constituyen un conjunto de estrategías para guiar heurísticas en la tarea de encontrar soluciones aceptables en un tiempo razonable para resolver un problema difícil o del que no se dispone información completa [10]. Usualmente poseen un componente estocástico, por lo que la solución depende de las variables aleatorias generadas y se describen los resultados basados en observaciones empíricas. Existen tres tipos de metaheurísticas: de trayectoria, poblacionales e hibridas.

En general, las meteheurísticas mejoran soluciones obtenidas anteriormente. Sus componentes principales son:

- Inicialización. Se requiere alguna solución inicial según la representación del problema particular, esta puede ser generada de manera aleatoria o utilizando algún algoritmo *greedy*.
- Operador de vecindad. Se debe disponer de un algortimo que, a partir de otra solución, genere un conjunto de soluciones "vecinas". Este operador puede entenderse como una pequeña perturbación en alguna componente de la representación utilizada.
- Criterio de selección. Entre las soluciones que pertenecen a la vecindad, se debe escoger la "mejor". Existen diversas políticas, las más conocidas son: el mejor de toda la vecindad, el primer mejor y el mejor de una porción de la vecindad. Escoger algún criterio sobre otro depende del problema.
- Criterio de convergencia. Al mejorar una solución a partir de la anterior, es indispensable contar con algún mecanismo que permita detener la ejecución de la metaheurística y devolver alguna solución. Los más conocidos son: detenerse al encontrar el óptimo (si este es conocido), luego de un número fijo de iteraciones y luego de un número fijo de iteraciones sin modificar la mejor solución.

3.1 Metaheurísticas de trayectoria

Al resolver un problema de optimización, las metaheurísticas de trayectoria, utilizan una solución y realizan mejoras sobre esta, iterativamente; pueden ser entendidas como "caminatas" sobre el espacio de búsqueda del problema. Probablemente la metaheurística de trayectoria más conocida es Local Search, que toma una solución inicial, encuentra sus vecinos y escoge el mejor según un criterio [10].

3.1.1 Búsqueda Local Iterada (Iterated Local Search)

Local Search podría quedar atrapado en un óptimo local y jamás llegar al global ya que se función es

Algorithm 3: Local Search

```
   Output: Mejor solución s*
1  s* = gen_sol();
2  while !criterio_convergencia() do
3  |   V = gen_vecinos(s');
4  |   s' = seleccionar_mejor(V);
5  |   if costo(s) < costo(s) then
6  |   |   s = s';
7  return s
```

intensificar la búsqueda en una vencindad. Para mitigar este riegos se crea una de las metaheurísticas más fáciles de implementar: Iterated Local search, que supone una mejora sobre LS. Primero se aplica LS a la solución inicial; luego, en cada iteración, se realiza una perturbación del óptimo local y se repite el proceso hasta un cumplir el criterio de aceptación [10].

Algorithm 4: Iterated Local Search

```
   Output: Mejor solución s*
1  s = gen_sol();
2  s* = local_search(s);
3  while !criterio_convergencia() do
4  |   s = perturbar(s*);
5  |   s' = local_search(s');
6  |   if s* < s' then
7  |   |   s* = s';
8  return s*
```

3.1.2 Recocido Simulado (Simulated Annealing)

Su nombre proviene del proceso físico de recocer sólidos, en el que un sólido cristalino es calentado y luego se deja enfriar muy lentamente hasta que alcance su configuración más estable y estructuralmente superior, por lo tanto está libre de defectos del cristal [4]. En términos de resolver un problema de optimización, el sólido representa una solución, "calentarlo" es perturbar la solución hasta que la temperatura se estabilice y el "enfriamiento" es una función no-creciente.

En cada iteración, Simulated Annealing, genera dos posibles soluciones: la actual y otra, las cuales son comparadas. Aquellas que mejoren la actual son siempre aceptadas; también existe una función probabilística dependiente de la temperatura que permite aceptar alguna solución considerada "inferior" y así, escapar de un mínimo local. Típicamente es una función no-creciente en cada iteración del algoritmo y al aproximarse la temperatura a cero la probabilidad de aceptar una solución que no mejora es menor [4]. La temperatura sólo decrece cuando se ha alcanzado una condición de equilibrio.

Algorithm 5: Simulated Annealing

```
   Input: T_inicial
   Output: Mejor solución s
1  s = gen_sol();
2  T = T_inicial;
3  while !criterio_convergencia() do
4  |   while !equilibrio do
5  |   |   /* Temperatura inicial */;
6  |   |   s' = gen_sol();
7  |   |   if costo(s) < costo(s') then
8  |   |   |   s = s';
9  |   |   else
10 |   |   |   r = rand();
11 |   |   |   delta = costo(s) - costo(s');
12 |   |   |   acep = 1/(1 + exp^(1+delta/T));
13 |   |   |   if r < acep then
14 |   |   |   |   s = s';
15 |   T = actualizar_temp();
16 return s
```

3.2 Metaheurísticas poblacionales

La mayoría de estos algoritmos están inspirados en fenómenos naturales y el comportamiento de los animales. Estas metaheurísticas toman una población inicial que está constituida por un conjunto de posibles soluciones del problema. Luego, iterativamente, se generan nuevos individuos (soluciones) a través de "cruces" y mutaciones en los genes (componentes del vector), los cuales son seleccionados para reemplazar a algunos de la actual población, creando una nueva. Este proceso se repite hasta alcanzar un criterio de convergencia dado [10].

Los cruces puede ser entendidos como una combinación de dos individuos de la población para crea otros nuevos, preservando algunas características de los originales. El operador de mutación efectúa una perturbación de un hijo generado; el propósito de este es realizar variaciones en la población que permitan explorar nuevos lugares en el espacio de solución, es decir, diversificar la búsqueda.

3.2.1 Scatter Search

Es un metaheurística evolutiva y poblacional que recombina soluciones seleccionadas de un "conjunto de referencia" para construir otras nuevas. El método comienza generando una población inicial cuyos individuos satisfacen un criterio de diversidad y calidad. El conjunto de referencia es construido seleccionando buenas representaciones de la población que se cruzan para proveer otras iniciales y luego mejorarlas con procedimientos basados en metaheurísticas de trayectoria como Local Search. De acuerdo con esto, el conjunto de referencia es actualizado para que contenga soluciones de buena calidad y otras que permitan diversificar la búsqueda. El proceso itera hasta que un criterio de parada se satisface [10].

Scatter Search aprovecha la información proveída por una heurística para crear las soluciones "buenas" e intensificar la búsqueda alrededor de estos espacios.

Algorithm 6: Scatter Search

```
Input: Población P
Output: Mejor solución p_mejor
P = mejorar_pop(P);
refSet = inicializar_buenos_individuos(P);
while !criterio_convergencia() do
    foreach r ∈ refSet do
        foreach r' ∈ refSet do
            s = combinar(r, r');
            s* = mejorar(s);
            P = actualizar(s*);
    P = ordenar(P);
    refSet = mejores(P);
    p_mejor = mejor(P);
return p_mejor
```

3.2.2 Differential Evolution

Es un algoritmo evolutivo que genera su población inicial de manera aleatoria y con un tamaño mayor o igual a 4. Suele utilizarse para problemas de optimización contínuos pues cada individuo tiene componentes con números reales.

El operador de cruce que utiliza está basado en la combinación líneal de las soluciones utilizando la distancia entre las mismas. Dado un padre x y tres otro individuos seleccionados aleatoriamente $r1, r2$ y $r3$, se crea un nuevo vector $u = r1 + F(r2 - r3)$. F representa un factor permite añadir un peso o importancia a la diferencia entre $r2$ y es tal que $r3$ y $F \in (0, 1)$ [4]. Finalmente se reemplaza p, con p', de la siguiente manera:

$$p'_i = \begin{cases} u_i & \text{si } rand < CR \\ p_i & \text{si } rand \geq CR \end{cases}$$

donde:

- $CR \in (0, 1)$: *Crossover Constant*, un parámetro que representa la probabilidad de cruce.
- $rand$: el número aleatorio entre 0 y 1 para escoger un componente u otro.
- $1 \leq i \leq n$: índice para cada característica.

Algorithm 7: Differential Evolution

```
Input: Población P, F, CR
Output: Mejor solución p_mejor
while !criterio_convergencia() do
    foreach p ∈ P do
        for i = 0 ... n − 1 do
            /* Mutar y cruzar */
            r = rand(0, 1)
            if r < CR then
                u_i = r1_i + F(r2_i − r3_i);
            else
                u_i = p_i
            /* Reemplazar según la regla */
            p'_i = reemplazar(p_i, u_i)
    p_mejor = mejor(P);
return p_mejor
```

4 Implementación

Para desarrollar las metaheurísticas y algoritmos explicados anteriormente y obtener los resultados se utilizó en lenguaje de programación C++ que se caracteriza por ser altamente eficiente.

En cuanto a la implementación de 1-NN, se retorna el porcentaje de instancias clasificadas correctamente para el conjunto de datos dado y el cálculo de la cercanía entre dos instancias se realiza utilizando la fórmula de distancia euclideana pesada con la importancia de cada caraterística:

$$\sqrt{\sum_{i=1}^{n} (w_i * (x_i - x'_i))^2}$$

Para modelar la solución al problema planteado se utilizó un vector de números reales con doble precisión

(`double`). El operador para generar vecinos selecciona un número aleatorio, *posToChange*, que representa que posición del vector a modificar, luego se escoge otro número aleatorio $rand \in [-1, 1]$ y se procede de la siguiente manera:

$$w'_{posToChange} = w_{posToChange} + rand$$

$$W' = \begin{cases} w'_{posToChange} = 0 & \text{si } w'_{posToChange} < 0 \\ normalizar(W') & \text{si } w'_{posToChange} \geq 1 \\ W' & \text{de lo contrario} \end{cases}$$

Con respecto a las metaheurísticas, el criterio para seleccionar el mejor vecino es "el mejor de una porción" ya que el "mejor" de todos no era factible puesto que los vecinos de una solución específica son infinitos. Se utilizaron dos criterios de convergencia: "cantidad fija de iteraciones" y "cantidad de iteraciones sin cambios". A continuación se específica el criterio de convergencia de las metaherísticas implementadas:

- Local Search : Cantidad fija de iteraciones. Si la mejor solución no varía después de dos iteraciones se considera que hubo convergencia.
- Iterated Local Search : Cantidad de iteraciones sin cambios.
- Simulated Annealing: Cantidad fija de iteraciones. Si la mejor solución no varía después de dos iteraciones se considera que hubo convergencia
- Scatter Search: ambas.
- Differential Evolution: ambas.

Como método de perturbación de una solución se utilizó la idea para generar un vecino pero no se modifica una posición del vector sino dos; esto se realiza para aquellas metaheurísticas en las que se requiere un operador de vecidad más amplio como ILS.

Simulated Annealing es uno de los algoritmos implementados que posee más sutilezas y variaciones; requiere que se planifique el enfriamiento (*Cooling Schedule*). Para alcanzar el estado de equilibrio (en donde no se varía la temperatura) se realiza un número de transiciones estáticas en las que se fijó una cantidad de iteraciones. Existen diversas funciones para disminuir la temperatura, en este caso se utilizó una función logaritmica que decrece según el número de iteraciones i:

$$\frac{T_{inicial}}{\log(i)}$$

Las metaheurísticas poblacionales requieren de un mecanismo que permita combinar soluciones para crear nuevas, esto se denomina operador de cruce. Debido a que el vector de pesos está compuesto por números reales, se implementó el operador Blend Alpha Crossover que selecciona dos padres, digamos Y y Z, para generar dos hijos a partir de estos ($C1$ y $C2$) y para cada una de sus componentes, i se genera un número aleatorio dentro del intervalo $[\min(y_i, z_i) - \alpha * diff, \max(y_i, z_i) - \alpha * diff]$ en donde α es un parámetro entre $[0, 1]$ y $diff = |y_i - z_i|$. En particular, se utiliza en Scatter Search ya que Diferential Evolution posee su propio operador de cruce.

Para obtener el subconjunto de entrenamiento y el de pruebas para un conjunto de datos específico se procede a desordenar el arreglo de instancias utilizando un generador de números aleatorios con una semilla, de manera que la partición se mantenga en cada corrida del algoritmo. Se seleccionan las primeras para entrenamiento y el resto para pruebas según un número real dado, por ejemplo: 0.7 genera 70% para entrenar y 30% para las pruebas.

5 Experimento

5.1 Conjunto de datos

Con la expansión del área de Inteligencia Artificial, se ha convertido en una necesidad contar con bases de datos que proporcionen información que permita a los investigadores, educadores y estudiantes del área realizar análisis sobre los algoritmos de *Machine Learning*. Una de las librerias más populares en la actualidad es UCI Machine Learning Repository, que mantiene una colección de conjuntos de datos, teorías de dominio y generadores de datos disponibles para toda la comunidad.

Para efectuar el análisis de las metaheurísticas a utilizar para resolver el problema APC se utilizan cuatro librerias disponibles en UCI:

Iris: este es probablemente el conjunto de datos mejor conocido y citado en la literatura de reconocimiento de patrones. Contiene tres clases de 50 instancias cada una, en donde cada clase corresponde a un tipo de planta Iris. Cada instancia posee las siguientes características: largo del sépalo, ancho del sépalo, largo del pétalo y ancho del pétalo en centímetros. Las clases a predecir son: "Iris-Setosa", "Iris-Versicolor" e "Iris-Virginica". [3].

Sonar: es una base de datos de detección de materiales mediante señales de sónar que "rebotan" en los objetos desde diferentes ángulos y bajo condiciones varias, discriminando entre cilindros metálicos(*mines*) y rocas(*rocks*). Cuenta con

111 *mines* y 97 *rocks* donde cada instancia es un conjunto de 60 números entre 0.0 y 1.0 que representan la energía entre una banda en particular, integrada sobre un periodo dispuestas en orden cresciente según el ángulo. Las clases a predecir son: "R" (*rocks*) y "M" (*mines*). [9].

Winsconsin Diagnostic Breast Cancer: es una base de datos contiene atributos calculados a partir de una imagen digitalizada de una aspiración con aguja fina de una masa en la mama. Se describen las características de los núcleos de las células presentes en la imagen. La tarea consiste en determinar si un tumor encontrado es benigno o maligno. Cuenta con 357 tumores beningnos y 212 malignos en donde cada instancia posee 30 caracteristicas, 10 valores reales computados por cada célula: radio, textura, perímetro, área, suavidad, compacto, concavidad, puntos cóncavos, simetría y dimensión fractal. [11].

Spambase: es una base de datos de detección de SPAM (correo basura) frente a correo electrónico seguro. Cuenta de 4601 ejemplos y 57 atributos en donde los primeros 48 corresponden al porcentaje de frecuencia de una palabra en particular en un correo, 6 corresponden al porcentaje de frecuencia de símbolos de puntuación y las últimas 3 son el promedio, máximo y la suma de la cantidad de letras en mayúsculas. Las clases a predecir son: 1 (spam), 0 (non-spam) que representan [6].

Waveform: contiene datos de ondas que fueron generadas con un programa en C. Dispone de 5000 instancias, 40 atributos con valores contínuos entre [0, 6], y 3 clases de ondas que represetan la característica a predecir. [1].

5.2 Particionamiento de los datos

Los conjuntos de datos mencionados anteriormente fueron particionados en dos subconjuntos: entrenamiento y prueba; se utilizó el 60% y 40% de los datos respectivamente. Cuando se analiza el comportamiento de una metaheurística probabilística, se desearía que el resultado obtenido no estuviera sesgado por una secuencia aleatoria concreta que pueda influir positiva o negativamente en las decisiones tomadas durante su ejecución. Para minimizar el impacto de esto, se realizaron varias particiones utilizando diferentes semillas para desordenar los datos.

5.3 Descripción del experimento

Se realizaron dos tipos de experimentos. El primero de ellos consistió en utilizar diferentes parámetros para cada una de las metaheurísticas y con sólo dos particiones, con el objetivo de encontrar los parámetros más adecuados (entre los probados) para cada uno de los conjuntos de datos. En las tablas 1-45 se encuentran los resultados del promedio de las corridas.

El segundo experimento se basa en los mejores parámetros obtenidos según los resultados promedio del primero y se realizan 5 particiones en las que se obtiene la media. En las tablas 46-100 se encuentran los resultados. En concreto, se consideraron los siguientes vectores de pesos (solución):

- Sin pesos: todas las características tienen el mismo peso (igual a 1).
- Relief: el resultado de ejecutar el algoritmo Relief.
- ILS aleatorio: el resultado de ejecutar Iterated Local Search tomando como solución inicial un vector aleatorio.
- ILS relief: el resultado de ejecutar Iterated Local Search tomando como solución inicial el resultado de relief.
- SA aleatorio: el resultado de ejecutar Simulated Annealing tomando como solución inicial un vector aleatorio.
- SA relief: el resultado de ejecutar Simulated Annealing tomando como solución inicial el resultado de relief.
- SS aleatorio: el resultado de ejecutar Scatter Search en donde todos los individuos de la población son aleatorios.
- SS relief: el resultado de ejecutar Scatter Search en donde uno de los individuos es el resultado de Relief y el resto de la población es aleatorios.
- DE: el resultado de ejecutar Diferential Evolution con una población aleatoria.

Los resultados de estos experimentos fueron obtenidos en una computadora portátil de 64 bits con un procesador Intel Core 2 Duo de 2.00 ghz con 3GB de memoria RAM.

6 Análisis de resultados

Primeramente, se desea conocer que mejora supone el uso de Relief para guiar el aprendizaje del clasificador 1-NN, por esta razón se comparan los resultados obtenidos para cada conjunto de datos utilizando la heurística y "sin pesos", es decir, cada componente de W es igual a 1.

Figura 1. Conjunto de datos: Iris. Comparación entre 1-NN sin considerar pesos y con la heurística Relief.

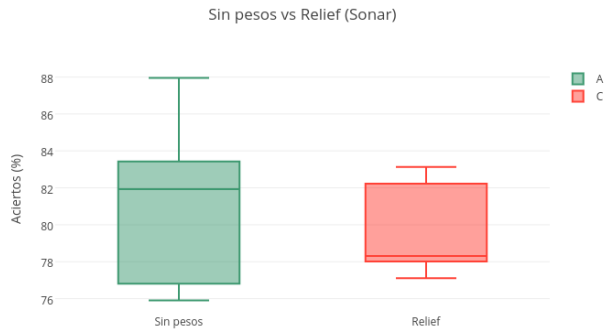

Figura 2. Conjunto de datos: Sonar. Comparación entre 1-NN sin considerar pesos y con la heurística Relief.

En el caso del conjunto de datos Iris y Sonar (Figura 1 y 2), el porcentaje de aciertos obtenidos al excluir los pesos presentan una mejora poco significativa con respecto a Relief. La diferencia entre ellos es cercano al 2% en promedio.

Figura 3. Conjunto de datos: WDBC. Comparación entre 1-NN sin considerar pesos y con la heurística Relief.

Para Wisconsin Diagnostic Breast Cancer Relief incrementa el promedio del porcentaje de aciertos en apróximadamente 6% con respecto a "sin pesos"

Figura 4. Conjunto de datos: Spambase. Comparación entre 1-NN sin considerar pesos y con la heurística Relief.

En cuanto a Spambase es evidente que el uso de Relief perjudica la obtención de aciertos, esto se debe probablemente a que contiene datos muy dispersos con varios atributo igual a 0, lo que impide que la heurística diferencie claramente un clase de otra.

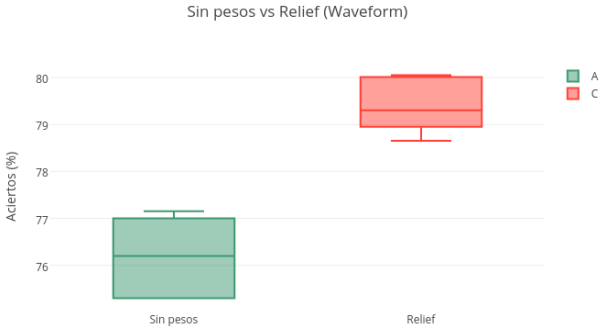

Figura 5. Conjunto de datos: Waveform. Comparación entre 1-NN sin considerar pesos y con la heurística Relief.

Se puede obeservar que el promedio de aciertos mejora con el uso de Relief en un 4%. En general, la metaheurística proporciona buenos resultados para los conjuntos de datos utilizados en tiempos aceptables; alrededor de 1 segundo para los conjuntos considerados pequeños (Iris, Sonar y WDBC) y 5 segundos para los grandes (Spambase y Waveform).

Con relación a las metaheurísticas, se realizaron ejecuciones de cada algoritmo con los párametros que se consideraron mejores según el primer experimento. Los resultados para Iris estos se obtuvieron con los siguientes parámetros:

- LS: 50 iteraciones, 20 vecinos.
- ILS: 50 iteraciones, 20 vecinos, 3 iteraciones sin

cambios.

- SA: 2 iteraciones, 8 vecinos, 100 de temperatura inicial y 100 iteraciones de estabilización.
- SS: 10 iteraciones, 10 tamaño de la población, 6 iteraciones sin cambios
- DE: 2 iteraciones, 5 tamaño de la poblacion, 0.5 CR, 0.7 de Factor de Escalamiento.

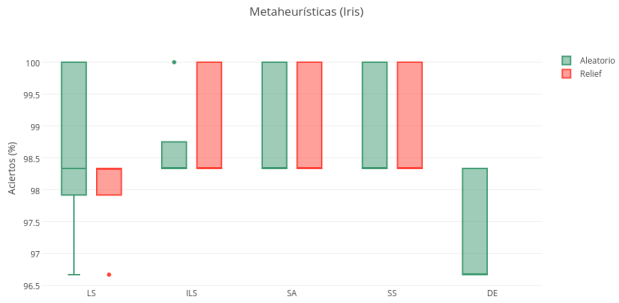

Figura 6. Conjunto de datos: Iris. Comparación entre todas las metaheurísticas.

Se puede observar en la Figura 6 que cualquier meteheurística permite clasificar con alta precisión, alrededero de 97%, los diferentes tipos de flores que componen a este conjunto de datos. Con respecto a las metaheurísticas de trayectoria, en ILS aleatorio se observa un resultado atípico puesto que una de las particiones consideradas permitió obtener un 100% de aciertos. Por su parte, SA presenta resultados (99% en promedio) similares, sin importar como se inicialice la solución que se optimizará.

Las metaheurísticas de trayectoria no varían de manera importante con respecto a las anteriores, sin embargo, DE converge más rápidamente por lo que su precesión es cerca de 2% peor a las demás. En cuanto al tiempo, ninguna sobrepasa los 0.5 segundos. La mejor de todas es Simulated Annealing con Relief ya que su tiempo es menor al de Scatter Search y obtienen los mismos reultados en promedio.

Para el conjunto de datos Sonar se utilizaron los siguientes parámetros:

- LS: 250 iteraciones, 150 vecinos.
- ILS: 250 iteraciones, 150 vecinos, 3 iteraciones sin cambios.
- SA: 250 iteraciones, 4 vecinos, 50 de temperatura inicial y 100 iteraciones de estabilización.
- SS: 250 iteraciones, 10 tamaño de la población, 8 iteraciones sin cambios
- DE: 150 iteraciones, 10 tamaño de la poblacion, 0.5 CR, 0.5 de Factor de Escalamiento.

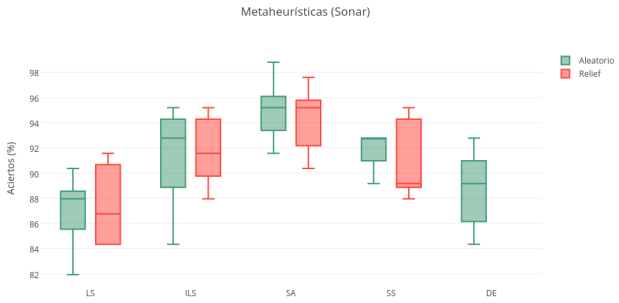

Figura 7. Conjunto de datos: Sonar. Comparación entre todas las metaheurísticas.

Sobre las metaheurísticas de trayectoria, SA es indiscutiblemente mejor a ILS y puesto que supera el promedio por apróximadamente 4% en prácticamente el mismo tiempo de ejecución. También es mejor que LS por 7% a pesar de que el tiempo que toma es mayor: 3 segundos de diferencia.

Es difícil definir claramente cuál de las dos metaheurísticas poblacionales presentan mejores resultados debido a que SS supera a DE en un 3% pero su tiempo de ejecución es el triple. La mejora de Simulated Annealing con respecto a Scatter Search es del 3%, sin embargo tarda 3 segundos más, por lo que se considera mejor SS.

Para el conjunto de datos Winsconsin Diagnostic Breast Cancer se utilizaron los siguientes parámetros:

- LS: 50 iteraciones, 30 vecinos.
- ILS: 50 iteraciones, 30 vecinos, 3 iteraciones sin cambios.
- SA: 40 iteraciones, 4 vecinos, 100 de temperatura inicial y 100 iteraciones de estabilización.
- SS: 40 iteraciones, 10 tamaño de la población, 8 iteraciones sin cambios
- DE: 50 iteraciones, 10 tamaño de la poblacion, 0.7 CR, 0.3 de Factor de Escalamiento.

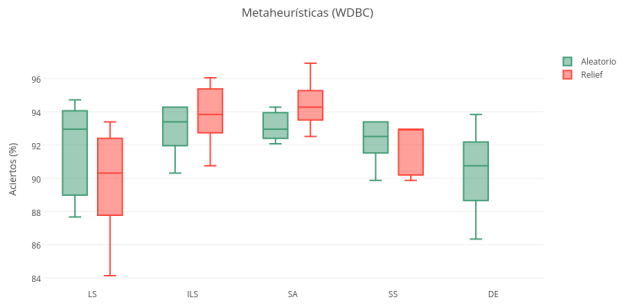

Figura 8. Conjunto de datos: WDBC. Comparación entre todas las metaheurísticas.

La metehaurística de trayectoria que determina con mayor certeza si una mujer tiene o no cáncer de mama según el conjunto de datos es SA, ya que muestra una tasa de aciertos de apróximadamente 94% y los resultados se obtuvieron en un tiempo relativamente corto, de 5 segundos apóximadamente. ILS presenta un porcentaje mayo de errores, además consume más tiempo.

En este caso SS tarda 4 veces más que DE y en promedio obtiene un 2% más de aciertos, por lo que DE es más apropiado para este problema; sin embargo Simulated Annealing con Relief es claramente superior, tanto en calidad como en tiempo.

Spambase fue la instancia más complicada de clasificar y que consumió más tiempo por lo que en el experimento 1 se realizaron menos pruebas; se escogieron los siguientes parámetros para cada metaheurística:

- LS: 20 iteraciones, 15 vecinos.
- ILS: 10 iteraciones, 5 vecinos, 3 iteraciones sin cambios.
- SA: 10 iteraciones, 4 vecinos, 100 de temperatura inicial y 100 iteraciones de estabilización.
- SS: 2 iteraciones, 5 tamaño de la población, 6 iteraciones sin cambios.
- DE: 15 iteraciones, 5 tamaño de la poblacion, 0.7 CR, 0.3 de Factor de Escalamiento.

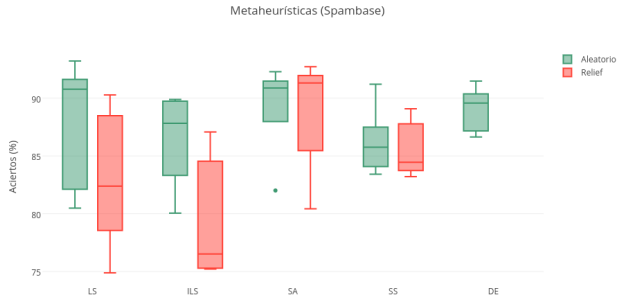

Figura 9. Conjunto de datos: Spambase. Comparación entre todas las metaheurísticas.

En est problema particular, ninguna de las metaheurísticas inicializadas con Relief presentaron un incremento considerable en la cantidad de aciertos, por el contrario, disminuyeron. En cuanto a las metaheurísticas de trayectoria, LS fue la que obtuvo mejores resultados, 87% en promedio, en la mitad del tiempo que SA.

DE supera el promedio de SS en un 4% y de LS en 2% en apróximadamente el mismo tiempo de cómputo que ambos, por lo que es el mejor para clasificar si un correo determinado es considerado "basura".

Los resultados obtenidos para Waveform utilizaron los siguientes parámetros:

- LS: 15 iteraciones, 10 vecinos.
- ILS: 2 iteraciones, 3 vecinos, 4 iteraciones sin cambios.
- SA: 2 iteraciones, 2 vecinos, 100 de temperatura inicial y 100 iteraciones de estabilización.
- SS: 2 iteraciones, 5 tamaño de la población, 2 iteraciones sin cambios.
- DE: 2 iteraciones, 5 tamaño de la poblacion, 0.5 CR, 0.7 de Factor de Escalamiento.

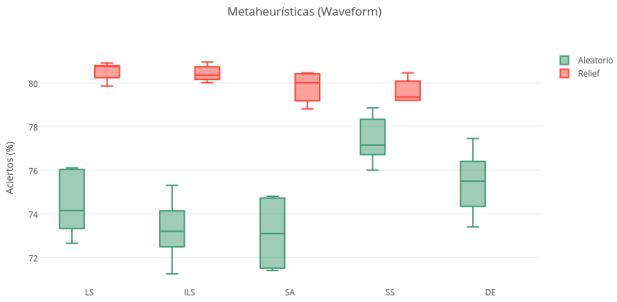

Figura 10. Conjunto de datos: Waveform. Comparación entre todas las metaheurísticas.

Es evidente el algoritmo Relief mejora los resultados de las todas las metheurísticas de manera considerable, en promedio 6% apróximadamente por el mismo tiempo de ejecución. A pesar de que ILS obtiene resultados un poco mejores que SA el tiempo que consume llegando a esta tasa de aciertos es más del triple que SA. Por otro lado, DE obtiene resultados con un 2% más de errores que SS, sin embargo tarda casi 6 veces menos. Simulated Annealing con Relief es la metaheurística como mejor relación calidad-tiempo para el para clasificar los tres tipos de olas.

Finalmente, es imperativo resaltar que ninguna de las metaheurísticas propuestas incrementa la tasa

de aciertos de manera significativa con respecto al algoritmo Relief, a excepción de Spambase.

Conclusiones

Se exploraron cuatro metaheurísticas para resolver el problema de Aprendizaje de Pesos en Características, típicamente resuelto utilizando redes neurales. En este caso se utilizó el clasificador 1-NN y algún algoritmo que permita ponderar los atributos de un conjunto de datos dado.

Las metaheurísticas consideradas incluyeron dos de trayectoria y dos poblacionales: Iterated Local Search, Simulated Annealing, Scatter Search y Differential Evolution. Adicionalmente se realizaron inicializaciones diferentes, primero se utilizó una solución generada aleatoriamnte y luego una creada con el algoritmo Relief.

En términos generales, la metaheurística que presentó mejores resultados según su tiempo y calidad de la solución fue Simulated Annealing. Sin embargo, al analizar más profudamente los resultados, se puede notar que no constituye una mjora considerable con respecto a Relief.

Dadas las características de los conjuntos de datos estudiados, se puede concluir que el uso de alguna de las metaheurísticas propuestas está justificado cuando los datos son dispersos, puesto que Relief no genera una solución los suficientemente confiable.

Por otro lado, es necesario realizar más experimentos para lograr obtener una caracterización de los problemas. Una de los posibles experimentos a realizar es perturbar más la solución en aquellos algoritmos que así lo requieran, como ILS. Otro posible experimento sería modificar los valores del α en Blend Alpha Crossover y analizar como varían los resultados.

Por último, se propone llevar a cabo estos experimentos en un computador con mejores recursos, ya que los resultados podrían mejorar considerablemente.

References

[1] Breiman, L. (1988). UCI Machine Learning Repository [`https://archive.ics.uci.edu/ml/machine-learning-databases/waveform/`]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Cano, J. Herrera, F. Lozano. M Using evolutionary algorithms as Instance Selection for data reduction in KDD: an experimental study. *IEEE Transaction on Evolutionary computation*, 2003.

[3] Fisher, R.A. (1988). UCI Machine Learning Repository [`https://archive.ics.uci.edu/ml/machine-learning-databases/iris/`]. Irvine, CA: University of California, School of Information and Computer Science.

[4] Glover, F Handbook of metaheuristics. *Operation resarch and management science*, 2003.

[5] Herrera, F. Metaheurísticas. *Seminario 2: Problemas de optimización con técnicas basadas en búsqueda local*, 2017. Disponible en: `http://sci2s.ugr.es/sites/default/files/files/Teaching/GraduatesCourses/Metaheuristicas/Sem02-Problemas-BusquedaLocal-MHs-16-17.pdf`

[6] Hopkins, M. (1999). UCI Machine Learning Repository [`https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/`]. Irvine, CA: University of California, School of Information and Computer Science.

[7] Kira, K., Rendell, A. A practical approach to feature selection, 1992.

[8] Mitchell, T. Machine Learning. *From Book News, Inc*, 1997.

[9] Sejnowski, T (año). UCI Machine Learning Repository [`http://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connectionist-bench/sonar/`]. Irvine, CA: University of California, School of Information and Computer Science.

[10] Talbi E. Metaheuristics: From desing to implementation *University of Lille*, 2009.

[11] Wolberg, W. (1995). UCI Machine Learning Repository [`https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/`]. Irvine, CA: University of California, School of Information and Computer Science.

Apéndice

Experimento 1

maxIter	NumGen	%Aciertos	Tiempo
2	3	95.8333	0
5	4	96.6667	0
10	8	97.5	0.5
50	20	98.3333	0
100	60	97.5	0

Tabla 1. Conjunto de Datos: Iris, Algoritmo: LS aleatorio.

maxIter	NumGen	%Aciertos	Tiempo
2	3	95.8333	0
5	4	95.8333	0
10	8	97.5	0
50	20	97.5	0
100	60	97.5	0

Tabla 2. Conjunto de Datos: Iris, Algoritmo: LS relief.

maxIter	NumGen	maxIterWC	%Aciertos	Tiempo
2	3	3	97.5	0.0072405
5	4	3	98.3333	0.0069645
10	8	3	97.5	0.013895
50	20	3	97.5	0.0254675
2	3	4	97.5	0.0072445
5	4	4	98.3333	0.011036
10	8	4	97.5	0.014299
50	20	4	97.5	0.036013

Tabla 3. Conjunto de Datos: Iris, Algoritmo: ILS aleatorio.

maxIter	NumGen	maxIterWC	%Aciertos	Tiempo
2	3	3	97.5	0.0056145
5	4	3	96.6667	0.005287
10	8	3	97.5	0.0106905
50	20	3	97.5	0.025469
2	3	4	97.5	0.007122
5	4	4	96.6667	0.006225
10	8	4	97.5	0.0178455
50	20	4	97.5	0.03602

Tabla 4. Conjunto de Datos: Iris, Algoritmo: ILS relief.

maxIter	NumGen	temp	estable	%Aciertos	Tiempo
2	2	100	100	95.8333	0.001733
2	4	100	100	97.5	0.004499
2	8	100	100	99.1667	0.032917
2	16	100	100	99.1667	0.516997
5	2	100	100	97.5	0.006576
5	4	100	100	97.5	0.0093115
5	8	100	100	99.1667	0.107548
5	16	100	100	99.1667	1.33503
10	2	100	100	98.3333	0.0093325
10	4	100	100	98.3333	0.0238445
10	8	100	100	99.1667	0.195268
10	16	100	100	99.1667	2.53034
50	2	100	100	99.1667	0.0323935
50	4	100	100	99.1667	0.1334
50	8	100	100	99.1667	1.05187
2	2	50	100	95.8333	0.00215
2	4	50	100	97.5	0.0045365
2	8	50	100	99.1667	0.0338565
5	2	50	100	97.5	0.0064175
5	4	50	100	97.5	0.0088875
5	8	50	100	99.1667	0.0691405
10	2	50	100	98.3333	0.0094815
10	4	50	100	98.3333	0.023769
10	8	50	100	99.1667	0.178455
50	2	50	100	98.3333	0.0351035
50	4	50	100	99.1667	0.135018
50	8	50	100	99.1667	0.916302
2	2	50	50	95.8333	0.002976
2	4	50	50	97.5	0.0052245
2	8	50	50	99.1667	0.0337035
5	2	50	50	97.5	0.0064165
5	4	50	50	97.5	0.0110045
5	8	50	50	99.1667	0.069088
10	2	50	50	98.3333	0.012835
10	4	50	50	98.3333	0.0270595
10	8	50	50	99.1667	0.180071
50	2	50	50	98.3333	0.0356615
50	4	50	50	99.1667	0.135627
50	8	50	50	99.1667	0.846995

Tabla 5. Conjunto de Datos: Iris, Algoritmo: SA aleatorio.

maxIter	NumGen	temp	estable	%Aciertos	Tiempo
2	2	100	100	96.6667	0.0019035
2	4	100	100	97.5	0.0048015
2	8	100	100	99.1667	0.035221
2	16	100	100	99.1667	0.540506
5	2	100	100	97.5	0.004302
5	4	100	100	99.1667	0.017063
5	8	100	100	99.1667	0.117603
5	16	100	100	99.1667	1.18248
10	2	100	100	97.5	0.0094085
10	4	100	100	99.1667	0.025932
10	8	100	100	99.1667	0.23537
10	16	100	100	99.1667	2.24277
50	2	100	100	99.1667	0.0350055
50	4	100	100	99.1667	0.150114
50	8	100	100	99.1667	1.15374
2	2	50	100	96.6667	0.001921
2	4	50	100	97.5	0.0048055
2	8	50	100	98.3333	0.054577
5	4	50	100	99.1667	0.0171175
5	8	50	100	99.1667	0.165833
10	2	50	100	97.5	0.0094955
5	2	50	100	97.5	0.004304
10	4	50	100	93.3333	0
10	8	50	100	99.1667	0.23754
50	2	50	100	98.3333	0.038803
50	4	50	100	99.1667	0.145719
50	8	50	100	99.1667	1.26785
2	2	50	50	96.6667	0.0018985
2	4	50	50	97.5	0.0048055
2	8	50	50	98.3333	0.05585
5	2	50	50	97.5	0.0043505
5	4	50	50	99.1667	0.0171715
5	8	50	50	99.1667	0.165803
10	2	50	50	97.5	0.0085295
10	4	50	50	99.1667	0.0274365
10	8	50	50	99.1667	0.230878
50	2	50	50	98.3333	0.0389235
50	4	50	50	99.1667	0.146285
50	8	50	50	99.1667	1.06618

Tabla 6. Conjunto de Datos: Iris, Algoritmo: SA relief.

maxIter	popSize	maxIterWC	%Aciertos	Tiempo
2	5	2	97.5	0.0225335
2	5	4	97.5	0.0227095
2	5	6	97.5	0.022474
2	5	8	97.5	0.023055
10	5	2	97.5	0.033398
10	5	4	98.3333	0.0550625
10	5	6	97.5	0.076187
10	5	8	98.3333	0.0983225
50	5	2	98.3333	0.0336085
50	5	4	98.3333	0.055009
50	5	6	98.3333	0.075022
50	5	8	98.3333	0.0965235
2	10	4	97.5	0.101311
2	10	6	98.3333	0.099992
2	10	8	98.3333	0.101543
10	10	2	98.3333	0.148722
10	10	4	98.3333	0.243982
10	10	6	99.1667	0.34275
10	10	8	99.1667	0.43498
50	10	2	98.3333	0.150023
50	10	4	98.3333	0.246687
50	10	6	99.1667	0.345515
50	10	8	99.1667	0.437731
2	20	4	99.1667	0.419829
2	20	6	99.1667	0.420353
2	20	8	99.1667	0.425347
10	20	2	99.1667	0.630875
10	20	4	99.1667	1.04929
10	20	6	99.1667	1.46079
10	20	8	99.1667	1.86167
50	20	2	99.1667	0.623293
50	20	4	99.1667	1.03611
50	20	6	99.1667	1.4464
50	20	8	98.3333	1.84269

Tabla 7. Conjunto de Datos: Iris, Algoritmo: SS aleatorio.

maxIter	popSize	maxIterWC	%Aciertos	Tiempo
2	5	2	97.5	0.023397
2	5	4	97.5	0.022993
2	5	6	97.5	0.0253065
2	5	8	97.5	0.026229
10	5	2	98.3333	0.0330275
10	5	4	98.3333	0.0551815
10	5	6	98.3333	0.076385
10	5	8	98.3333	0.0992485
50	5	2	97.5	0.0334965
50	5	4	98.3333	0.0561185
50	5	6	98.3333	0.0768475
50	5	8	99.1667	0.098724
2	10	4	98.3333	0.100441
2	10	6	98.3333	0.101094
2	10	8	98.3333	0.100044
10	10	2	99.1667	0.148192
10	10	4	99.1667	0.245998
10	10	6	99.1667	0.342861
10	10	8	99.1667	0.444935
50	10	2	99.1667	0.151731
50	10	4	98.3333	0.246533
50	10	6	98.3333	0.346445
50	10	8	99.1667	0.444733
2	20	4	99.1667	0.423689
2	20	6	98.3333	0.420434
2	20	8	99.1667	0.420574
10	20	2	99.1667	0.628231
10	20	4	99.1667	1.03705
10	20	6	99.1667	1.43244
10	20	8	99.1667	1.84992
50	20	2	99.1667	0.627933
50	20	4	99.1667	1.03112
50	20	6	99.1667	1.4523
50	20	8	99.1667	1.84809

Tabla 8. Conjunto de Datos: Iris, Algoritmo: SS relief.

maxIter	popSize	CR	F	%Aciertos	Tiempo
2	5	0.5	0.3	97.5	0.0045245
2	5	0.7	0.3	97.5	0.005204
2	5	0.3	0.5	97.5	0.0044515
2	5	0.5	0.5	98.3333	0.005593
2	5	0.7	0.5	98.3333	0.00463
2	5	0.3	0.7	97.5	0.0056275
2	5	0.5	0.7	97.5	0.0053335
2	5	0.7	0.7	97.5	0.0053775
2	10	0.3	0.3	96.6667	0.009656
2	10	0.5	0.3	96.6667	0.00874
2	10	0.7	0.3	96.6667	0.009636
2	10	0.3	0.5	96.6667	0.009811
2	10	0.5	0.5	96.6667	0.009863
2	10	0.7	0.5	96.6667	0.009428
2	10	0.3	0.7	96.6667	0.0088645
2	10	0.5	0.7	96.6667	0.0098275
2	10	0.7	0.7	97.5	0.0096955
2	20	0.3	0.3	97.5	0.017315
2	20	0.5	0.3	97.5	0.018251
2	20	0.7	0.3	97.5	0.0183435
2	20	0.3	0.5	97.5	0.018105
2	20	0.5	0.5	97.5	0.018347
2	20	0.7	0.5	98.3333	0.017279
2	20	0.3	0.7	97.5	0.0173245
2	20	0.5	0.7	97.5	0.017426
2	20	0.7	0.7	97.5	0.017672
10	5	0.3	0.3	97.5	0.005817
10	5	0.5	0.3	97.5	0.0093115
10	5	0.7	0.3	97.5	0.0088175
10	5	0.3	0.5	97.5	0.011418
10	5	0.5	0.5	98.3333	0.007922
10	5	0.7	0.5	98.3333	0.0147545
10	5	0.3	0.7	98.3333	0.0087395
10	5	0.5	0.7	98.3333	0.011561
10	5	0.7	0.7	97.5	0.0086975
10	10	0.3	0.3	97.5	0.0309365
10	10	0.5	0.3	97.5	0.0275555
10	10	0.7	0.3	98.3333	0.034218
10	10	0.3	0.5	97.5	0.03083
10	10	0.5	0.5	97.5	0.0322265
10	10	0.7	0.5	98.3333	0.031213
10	10	0.3	0.7	97.5	0.031183
10	10	0.5	0.7	97.5	0.0360255
10	20	0.3	0.3	97.5	0.0646025
10	20	0.5	0.3	98.3333	0.057857
10	20	0.7	0.3	98.3333	0.061248
10	20	0.3	0.5	98.3333	0.071442
10	20	0.5	0.5	98.3333	0.0644825
10	20	0.7	0.5	97.5	0.0545015
10	20	0.3	0.7	97.5	0.071423
10	20	0.5	0.7	98.3333	0.0611295
10	20	0.7	0.7	98.3333	0.0646335

Tabla 9. Conjunto de Datos: Iris, Algoritmo: DE.

maxIter	NumGen	%Aciertos	Tiempo
150	80	81.9277	0
250	150	89.759	1
350	250	84.3373	0.5
500	450	84.9398	3.5

Tabla 10. Conjunto de Datos: Sonar, Algoritmo: LS

aleatorio.

maxIter	NumGen	%Aciertos	Tiempo
150	80	84.3373	0.5
250	150	84.3373	0.5
350	250	84.9398	1.5
500	450	86.747	3

Tabla 11. Conjunto de Datos: Sonar, Algoritmo: LS relief.

maxIter	NumGen	maxIterWC	%Aciertos	Tiempo
150	80	3	87.9518	1.94728
250	150	3	89.1566	5.30416
350	250	3	90.3614	10.9038
500	450	3	89.1566	20.5411
150	80	4	88.5542	1.95119
250	150	4	87.9518	6.23948
350	250	4	88.5542	9.16794
500	450	4	90.3614	11.2005

Tabla 12. Conjunto de Datos: Sonar, Algoritmo: ILS aleatorio.

maxIter	NumGen	maxIterWC	%Aciertos	Tiempo
150	80	3	85.5422	1.83963
250	150	3	88.5542	5.61831
350	250	3	88.5542	9.85058
500	450	3	88.5542	15.5378
150	80	4	86.747	3.83648
250	150	4	89.759	8.40371
350	250	4	89.1566	15.8955
500	450	4	89.759	23.308

Tabla 13. Conjunto de Datos: Sonar, Algoritmo: ILS relief.

maxIter	NumGen	temp	estable	%Aciertos	Tiempo
150	2	100	100	87.9518	0.925362
150	4	100	100	91.5663	3.81631
150	8	100	100	95.1807	28.8836
150	16	100	100	97.5904	265.156
250	2	100	100	90.3614	1.47956
250	4	100	100	92.7711	5.7066
250	8	100	100	95.1807	43.3814
250	16	100	100	96.988	439.596
350	2	100	100	89.759	2.08988
350	5	100	100	92.1687	8.11332
350	8	100	100	93.9759	64.7297
350	16	100	100	95.7831	646.528
500	2	100	100	90.9639	2.96337
500	4	100	100	94.5783	11.696
500	8	100	100	95.7831	86.6341
150	2	50	100	88.5542	0.883367
150	4	50	100	91.5663	3.46104
150	8	50	100	95.1807	23.866
250	2	50	100	90.9639	1.48774
250	4	50	100	94.5783	5.45391
250	8	50	100	94.5783	34.4061
350	2	50	100	92.1687	1.9737
350	5	50	100	94.5783	7.49999
350	8	50	100	95.1807	48.6543
500	2	50	100	90.3614	2.88556
500	4	50	100	93.3735	10.7061
500	8	50	100	95.1807	65.1663
150	2	50	50	88.5542	0.89294
150	4	50	50	91.5663	3.43418
150	8	50	50	95.1807	22.7999
250	2	50	50	90.9639	1.48147
250	4	50	50	94.5783	5.4483
250	8	50	50	94.5783	33.6006
350	2	50	50	92.1687	1.9824
350	4	50	50	94.5783	7.50827
350	8	50	50	93.9759	45.7193
500	2	50	50	90.3614	2.89191
500	4	50	50	93.3735	10.7094
500	8	50	50	95.7831	65.4803

Tabla 14. Conjunto de Datos: Sonar, Algoritmo: SA aleatorio.

maxIter	NumGen	temp	estable	%Aciertos	Tiempo
150	2	100	100	87.9518	0.921946
150	4	100	100	92.7711	3.5407
150	8	100	100	94.5783	28.7262
150	16	100	100	96.3855	258.56
250	2	100	100	90.3614	1.47956
250	4	100	100	89.1566	1.44483
250	8	100	100	92.1687	5.82396
250	16	100	100	96.3855	422.046
350	2	100	100	92.1687	2.13963
350	5	100	100	93.3735	8.47296
350	8	100	100	94.5783	57.33
350	16	100	100	97.5904	667.649
500	2	100	100	90.9639	3.04349
500	4	100	100	93.3735	11.7638
500	8	100	100	96.3855	86.0461
150	2	50	100	90.3614	0.904235
150	4	50	100	90.3614	3.17135
150	8	50	100	95.7831	23.187
250	2	50	100	91.5663	1.43878
250	4	50	100	93.3735	5.51446
250	8	50	100	95.7831	34.9927
350	2	50	100	89.1566	1.95869
350	5	50	100	92.7711	7.45297
350	8	50	100	95.7831	48.7565
500	2	50	100	94.5783	2.8903
500	4	50	100	93.9759	10.3589
500	8	50	100	95.7831	68.334
150	2	50	50	90.3614	0.913408
150	4	50	50	90.3614	3.16557
150	8	50	50	93.9759	22.5214
250	2	50	50	91.5663	1.43654
250	4	50	50	93.3735	5.47237
250	8	50	50	96.3855	34.4863
350	2	50	50	89.1566	1.95436
350	4	50	50	92.7711	7.39022
350	8	50	50	93.3735	46.9276
500	2	50	50	94.5783	2.91869
500	4	50	50	93.9759	10.2619
500	8	50	50	95.1807	66.1648

Tabla 15. Conjunto de Datos: Sonar, Algoritmo: SA relief.

maxIter	popSize	maxIterWC	Aciertos	Tiempo
150	5	2	84.9398	0.1415
150	5	4	83.1325	0.2335
150	5	6	84.9398	0.323
150	5	8	84.9398	0.438
250	5	2	84.3373	0.151
250	5	4	88.5542	0.2365
250	5	6	83.7349	0.342
250	5	8	87.3494	0.4265
350	5	2	85.5422	0.1505
350	5	4	84.3373	0.2425
350	5	6	86.1446	0.33
350	5	8	85.5422	0.428
150	10	4	87.9518	1.0595
150	10	6	89.1566	1.4825
150	10	8	88.5542	1.9335
250	10	2	86.747	0.6465
250	10	4	87.9518	1.0675
250	10	6	86.747	1.501
250	10	8	89.1566	1.8915
350	10	2	87.3494	0.633
350	10	4	87.9518	1.0615
350	10	6	89.1566	1.4995
350	10	8	88.5542	1.947
150	20	4	89.759	4.736
150	20	6	91.5663	6.4605
150	20	8	91.5663	8.393
250	20	2	87.3494	2.779
250	20	4	90.3614	4.6155
250	20	6	89.759	6.3905
250	20	8	90.9639	8.17
350	20	2	86.747	2.7755
350	20	4	89.1566	4.5665
350	20	6	90.9639	6.405
350	20	8	89.759	8.1315

Tabla 16. Conjunto de Datos: Sonar, Algoritmo: SS aleatorio.

maxIter	popSize	maxIterWC	Aciertos	Tiempo
150	5	2	83.7349	0.145
150	5	4	85.5422	0.233
150	5	6	83.7349	0.3275
150	5	8	84.9398	0.4145
250	5	2	86.1446	0.1435
250	5	4	86.1446	0.2325
250	5	6	86.747	0.3335
250	5	8	84.3373	0.4305
350	5	2	84.9398	0.147
350	5	4	86.1446	0.244
350	5	6	86.747	0.358
350	5	8	86.1446	0.4485
150	10	4	88.5542	1.0865
150	10	6	88.5542	1.5075
150	10	8	87.3494	1.96
250	10	2	86.747	0.6515
250	10	4	87.3494	1.0915
250	10	6	89.1566	1.511
250	10	8	89.1566	1.9465
350	10	2	85.5422	0.66
350	10	4	86.747	1.086
350	10	6	86.747	1.5155
350	10	8	90.3614	1.9515
150	20	4	89.759	4.5745
150	20	6	89.759	6.3445
150	20	8	90.9639	8.217
250	20	2	87.9518	2.743
250	20	4	89.759	4.595
250	20	6	89.759	6.404
250	20	8	91.5663	8.157
350	20	2	87.9518	2.7595
350	20	4	89.759	4.597
350	20	6	90.9639	6.38
350	20	8	89.759	8.1795

Tabla 17. Conjunto de Datos: Sonar, Algoritmo: SS relief.

maxIter	popSize	CR	F	%Aciertos	Tiempo
150	5	0.3	0.3	86.747	0.072
150	5	0.5	0.3	85.5422	0.0815
150	5	0.7	0.3	86.747	0.0605
150	5	0.3	0.5	85.5422	0.065
150	5	0.5	0.5	84.3373	0.062
150	5	0.7	0.5	86.1446	0.057
150	5	0.3	0.7	85.5422	0.0505
150	5	0.5	0.7	86.747	0.074
150	5	0.7	0.7	84.3373	0.071
150	10	0.3	0.3	86.747	0.2485
150	10	0.5	0.3	89.1566	0.3595
150	10	0.7	0.3	86.1446	0.29
150	10	0.3	0.5	87.9518	0.2235
150	10	0.5	0.5	89.1566	0.1865
150	10	0.7	0.5	88.5542	0.259
150	10	0.3	0.7	87.3494	0.152
150	10	0.5	0.7	88.5542	0.23
150	10	0.7	0.7	88.5542	0.2455
150	20	0.3	0.3	89.759	1.2115
150	20	0.5	0.3	90.3614	1.1
150	20	0.7	0.3	89.759	0.6515
150	20	0.3	0.5	87.9518	0.954
150	20	0.5	0.5	89.1566	0.8625
150	20	0.7	0.5	87.9518	0.6055
150	20	0.3	0.7	89.1566	0.864
150	20	0.5	0.7	88.5542	0.591
150	20	0.7	0.7	89.759	0.5215
350	5	0.3	0.3	86.747	0.079
350	5	0.5	0.3	85.5422	0.0485
350	5	0.7	0.3	83.1325	0.0555
350	5	0.3	0.5	84.3373	0.0865
350	5	0.5	0.5	83.1325	0.045
350	5	0.7	0.5	86.747	0.0635
350	5	0.3	0.7	83.7349	0.056
350	5	0.5	0.7	84.9398	0.0815
350	5	0.7	0.7	85.5422	0.0835
350	10	0.3	0.3	85.5422	0.2145
350	10	0.5	0.3	86.1446	0.179
350	10	0.7	0.3	86.1446	0.1635
350	10	0.3	0.5	86.747	0.2025
350	10	0.5	0.5	87.3494	0.2955
350	10	0.7	0.5	87.9518	0.275
350	10	0.3	0.7	85.5422	0.133
350	10	0.5	0.7	88.5542	0.2365
350	20	0.3	0.3	88.5542	1.58317
350	20	0.5	0.3	89.1566	1.46866
350	20	0.7	0.3	89.1566	1.99203
350	20	0.3	0.5	88.5542	0.916451
350	20	0.5	0.5	89.1566	1.28393
350	20	0.7	0.5	87.3494	0.888943
350	20	0.3	0.7	89.759	1.35783
350	20	0.5	0.7	87.9518	1.41385
350	20	0.7	0.7	87.9518	1.16448

Tabla 18. Conjunto de Datos: Sonar, Algoritmo: DE.

maxIter	NumGen	%Aciertos	Tiempo
15	10	83.2599	0.5
40	20	81.7181	0.5
50	30	90.7489	1.5
100	60	92.2907	4
150	80	90.7489	3

Tabla 19. Conjunto de Datos: WDBC, Algoritmo: LS aleatorio.

maxIter	NumGen	%Aciertos	Tiempo
15	10	89.4273	0
40	20	93.1718	1
50	30	92.0705	1
100	60	91.63	1.5
150	80	93.1718	2.5

Tabla 20. Conjunto de Datos: WDBC, Algoritmo: LS relief.

maxIter	NumGen	maxIterWC	%Aciertos	Tiempo
15	10	3	89.4273	1.33782
40	20	3	89.6476	3.04014
50	30	3	94.4934	8.26167
100	60	3	96.2555	30.1906
150	80	3	92.7313	22.1498
15	10	4	85.9031	1.81677
40	20	4	93.1718	4.35872
50	30	4	91.8502	6.88045
100	60	4	94.4934	17.574
150	80	4	93.6123	34.4008

Tabla 21. Conjunto de Datos: WDBC, Algoritmo: ILS aleatorio.

maxIter	NumGen	maxIterWC	%Aciertos	Tiempo
15	10	3	92.2907	1.65303
40	20	3	92.511	3.18228
50	30	3	92.511	6.70977
100	60	3	94.9339	12.0594
150	80	3	94.4934	13.2045
15	10	4	91.4097	1.04342
40	20	4	93.3921	4.72845
50	30	4	94.9339	6.89954
100	60	4	94.4934	10.692
150	80	4	95.3744	26.3753

Tabla 22. Conjunto de Datos: WDBC, Algoritmo: ILS relief.

maxIter	NumGen	temp	estable	%Aciertos	Tiempo
15	2	100	100	81.9383	0.393873
15	4	100	100	88.1057	1.70711
15	8	100	100	95.815	14.3292
15	16	100	100	96.2555	137.921
40	2	100	100	92.2907	1.27919
40	4	100	100	94.2731	5.7194
40	8	100	100	96.0352	39.2137
40	16	100	100	96.9163	354.368
50	2	100	100	94.2731	1.48484
50	4	100	100	94.2731	6.12073
50	8	100	100	96.696	52.372
50	16	100	100	96.9163	445.535
100	2	100	100	93.6123	3.03059
100	4	100	100	94.7137	11.665
100	8	100	100	96.2555	90.7215
15	2	50	100	84.141	0.453956
15	4	50	100	92.0705	2.06082
15	8	50	100	95.1542	13.1658
40	2	50	100	90.0881	1.25744
40	4	50	100	95.1542	5.39478
40	8	50	100	96.696	27.4816
50	2	50	100	92.9515	1.43874
50	4	50	100	93.6123	5.73546
50	8	50	100	95.815	50.0533
100	2	50	100	94.4934	2.77218
100	4	50	100	95.3744	12.2462
100	8	50	100	96.696	96.8671
15	2	50	50	84.141	0.45967
15	4	50	50	92.0705	2.06338
15	8	50	50	95.1542	12.5151
40	2	50	50	90.0881	1.26065
40	4	50	50	94.2731	5.51484
40	8	50	50	96.0352	26.7602
50	2	50	50	92.9515	1.45356
50	4	50	50	93.6123	5.70265
50	8	50	50	96.696	42.8088
100	2	50	50	94.4934	2.79473
100	4	50	50	95.3744	12.1826
100	8	50	50	96.9163	85.8242

Tabla 23. Conjunto de Datos: WDBC, Algoritmo: SA aleatorio.

maxIter	NumGen	temp	estable	%Aciertos	Tiempo
15	2	100	100	87.8855	0.524754
15	4	100	100	91.1894	1.81186
15	8	100	100	94.2731	10.9386
15	16	100	100	96.9163	133.598
40	2	100	100	92.0705	1.16754
40	4	100	100	93.6123	4.88903
40	8	100	100	96.2555	34.258
40	16	100	100	96.9163	342.471
50	2	100	100	92.0705	1.42255
50	4	100	100	94.9339	6.2816
50	8	100	100	96.4758	51.9605
50	16	100	100	97.3568	480.319
100	2	100	100	91.63	3.02778
100	4	100	100	95.1542	10.7134
100	8	100	100	96.696	94.1997
15	2	50	100	88.7665	0.460307
15	4	50	100	94.2731	1.7092
15	8	50	100	96.0352	16.529
40	2	50	100	93.6123	1.07361
40	4	50	100	95.1542	5.98671
40	8	50	100	96.696	34.518
50	2	50	100	92.9515	1.41514
50	4	50	100	92.9515	5.1211
50	8	50	100	95.815	49.413
100	2	50	100	93.1718	3.08979
100	4	50	100	94.7137	12.062
100	8	50	100	96.9163	86.0422
15	2	50	50	88.7665	0.459081
15	4	50	50	94.2731	1.71408
15	8	50	50	95.5947	16.5173
40	2	50	50	93.6123	1.0719
40	4	50	50	95.1542	5.93842
40	8	50	50	95.1542	28.8965
50	2	50	50	92.9515	1.41327
50	4	50	50	92.9515	5.08278
50	8	50	50	95.815	35.1078
100	2	50	50	93.1718	3.10792
100	4	50	50	94.7137	11.9579
100	8	50	50	96.0352	78.6358

Tabla 24. Conjunto de Datos: WDBC, Algoritmo: SA relief.

maxIter	popSize	maxIterWC	%Aciertos	Tiempo
15	5	2	87.4449	0.5365
15	5	4	83.7004	0.9135
15	5	6	84.5815	1.2865
15	5	8	89.207	1.682
40	5	2	86.3436	0.572
40	5	4	88.7665	0.9405
40	5	6	85.6828	1.287
40	5	8	85.2423	1.64
50	5	2	87.4449	0.5625
50	5	4	83.9207	0.94
50	5	6	87.2247	1.307
50	5	8	87.2247	1.6985
15	10	2	86.5639	2.5285
15	10	4	89.4273	4.099
15	10	6	91.63	5.865
15	10	8	89.4273	7.427
40	10	2	88.1057	2.5265
40	10	4	88.7665	4.168
40	10	6	89.207	5.6785
40	10	8	91.1894	7.323
50	10	2	89.6476	2.5165
50	10	4	89.207	4.0875
50	10	6	89.6476	5.734
50	10	8	91.1894	7.413

Tabla 25. Conjunto de Datos: WDBC, Algoritmo: SS aleatorio.

maxIter	popSize	maxIterWC	%Aciertos	Tiempo
15	5	2	82.3789	0.5355
15	5	4	85.022	0.894
15	5	6	87.0044	1.2035
15	5	8	85.9031	1.651
40	5	2	85.2423	0.5985
40	5	4	86.7841	0.961
40	5	6	87.8855	1.3635
40	5	8	88.7665	1.619
50	5	2	85.6828	0.561
50	5	4	85.2423	0.948
50	5	6	88.326	1.263
50	5	8	89.6476	1.661
15	10	2	87.2247	2.429
15	10	4	89.6476	4.051
15	10	6	90.9692	5.6175
15	10	8	90.0881	7.438
40	10	2	88.5463	2.525
40	10	4	90.0881	4.1405
40	10	6	92.2907	5.6865
40	10	8	90.7489	7.424
50	10	2	88.5463	2.5145
50	10	4	88.7665	4.0895
50	10	6	91.4097	5.728
50	10	8	89.6476	7.3905

Tabla 26. Conjunto de Datos: WDBC, Algoritmo: SS relief.

maxIter	popSize	CR	F	%Aciertos	Tiempo
15	5	0.3	0.3	84.8018	0.4335
15	5	0.5	0.3	87.0044	0.4265
15	5	0.7	0.3	88.326	0.408
15	5	0.3	0.5	87.4449	0.412
15	5	0.5	0.5	90.3084	0.4365
15	5	0.7	0.5	89.6476	0.404
15	5	0.3	0.7	86.7841	0.424
15	5	0.5	0.7	90.0881	0.3305
15	5	0.7	0.7	88.9868	0.448
15	10	0.3	0.3	87.8855	0.827
15	10	0.5	0.3	89.207	0.9295
15	10	0.7	0.3	89.6476	0.929
15	10	0.3	0.5	88.326	0.8995
15	10	0.7	0.5	90.7489	0.849
15	10	0.5	0.7	89.4273	0.807
15	10	0.3	0.7	86.5639	0.8545
15	10	0.5	0.7	89.207	0.851
15	10	0.7	0.7	90.0881	0.8435
15	20	0.3	0.3	88.326	1.7105
15	20	0.5	0.3	91.63	1.73
15	20	0.7	0.3	90.7489	1.703
15	20	0.3	0.5	89.6476	1.715
15	20	0.5	0.5	90.7489	1.699
15	20	0.7	0.5	90.5286	1.7085
15	20	0.3	0.7	90.3084	1.793
15	20	0.5	0.7	89.207	1.7165
15	20	0.7	0.7	89.8678	1.7015
50	5	0.3	0.3	86.3436	0.6145
50	5	0.5	0.3	89.6476	0.582
50	5	0.7	0.3	86.1233	0.3485
50	5	0.3	0.5	87.0044	0.4985
50	5	0.3	0.5	90.0881	0.5835
50	5	0.5	0.5	85.022	0.329
50	5	0.7	0.5	86.5639	0.334
50	5	0.3	0.7	87.6652	0.532
50	5	0.5	0.7	87.8855	0.4585
50	5	0.7	0.7	85.4626	0.543
50	10	0.3	0.3	88.9868	1.8325
50	10	0.5	0.3	89.207	1.0195
50	10	0.7	0.3	91.8502	1.6535
50	10	0.3	0.5	91.4097	2.077
50	10	0.5	0.5	90.9692	1.7965
50	10	0.7	0.5	91.63	1.2105
50	10	0.3	0.7	92.511	2.1145
50	10	0.5	0.7	93.6123	2.512
50	20	0.3	0.3	93.1718	13.2436
50	20	0.5	0.3	92.2907	10.7609
50	20	0.7	0.3	94.2731	13.2424
50	20	0.3	0.5	92.9515	13.2435
50	20	0.5	0.5	92.511	12.5972
50	20	0.7	0.5	92.511	12.8516
50	20	0.3	0.7	92.2907	12.9884
50	20	0.5	0.7	92.7313	13.2423
50	20	0.7	0.7	93.3921	12.328

Tabla 27. Conjunto de Datos: WDBC, Algoritmo: DE.

maxIter	NumGen	%Aciertos	Tiempo
2	3	79.5109	3.5
10	5	82.4728	8.5
15	10	80.9511	15.5
20	15	90.2174	140.5

Tabla 28. Conjunto de Datos: Spambase, Algoritmo: LS aleatorio.

maxIter	NumGen	%Aciertos	Tiempo
2	3	79.5109	3.5
10	5	82.4728	8.5
15	10	80.9511	15.5
20	15	90.2174	140.5

Tabla 29. Conjunto de Datos: Spambase, Algoritmo: LS relief.

maxIter	NumGen	maxIterWC	%Aciertos	Tiempo
2	3	3	86.3043	16.355
10	5	3	86.4674	42.871
15	10	3	90.0543	257.036
20	15	3	92.663	273.822
2	3	4	81.7935	25.3435
10	5	4	90.2989	44.516
15	10	4	91.3315	261.387
20	15	4	92.2554	555.618

Tabla 30. Conjunto de Datos: Spambase, Algoritmo: ILS aleatorio.

maxIter	NumGen	maxIterWC	%Aciertos	Tiempo
2	3	3	75.462	12.2865
10	5	3	81.1413	69.0075
15	10	3	87.6087	214.852
20	15	3	90.625	375.861
2	3	4	76.3859	23.97
10	5	4	77.7989	60.8375
15	10	4	80.5978	203.894
20	15	4	91.2772	759.406

Tabla 31. Conjunto de Datos: Spambase, Algoritmo: ILS relief.

maxIter	NumGen	temp	estable	%Aciertos	Tiempo
2	2	100	100	82.3913	11.4089
2	4	100	100	81.3043	74.5867
10	2	100	100	80.5707	52.0373
10	4	100	100	91.7935	309.738
15	2	100	100	83.6957	65.5999
15	4	100	100	91.7663	408.804
20	2	100	100	87.4185	99.1111
2	2	50	100	83.3424	10.7511
2	4	50	100	84.9185	39.1893
10	2	50	100	84.538	37.9376
10	4	50	100	91.9837	338.508
15	2	50	100	83.6957	65.5943
20	2	50	100	83.4239	95.617

Tabla 32. Conjunto de Datos: Spambase, Algoritmo: SA aleatorio.

maxIter	NumGen	temp	estable	%Aciertos	Tiempo
2	2	100	100	76.7935	19.0017
2	4	100	100	78.0978	44.2843
10	2	100	100	79.0489	47.2527
10	4	100	100	88.8043	282.788
15	2	100	100	78.0163	64.0012
15	4	100	100	90.3533	417.151
20	2	100	100	77.7174	67.2651
2	2	50	100	75.7337	10.1409
2	4	50	100	76.2228	39.5184
10	2	50	100	77.5815	39.4676
10	4	50	100	88.8043	282.948
15	2	50	100	78.0163	64.9265
20	2	50	100	77.7174	67.4936

Tabla 33. Conjunto de Datos: Spambase, Algoritmo: SA relief.

maxIter	popSize	maxIterWC	%Aciertos	Tiempo
2	5	2	86.5489	50.5535
2	5	4	89.4293	53.107
2	5	6	86.1413	52.2165
2	5	8	85.0543	50.5195
10	5	2	84.375	77.623
10	5	4	84.3207	126.447
10	5	6	90.1359	174.495
10	5	8	90.2174	221.416
15	5	2	86.413	77.7435
15	5	4	89.2935	125.811
15	5	6	86.8478	171.815
15	5	8	89.837	219.449
2	10	2	89.1033	209.662
2	10	4	87.962	212.627

Tabla 34. Conjunto de Datos: Spambase, Algoritmo: SS aleatorio.

maxIter	popSize	maxIterWC	%Aciertos	Tiempo
2	5	2	85.5163	51.933
2	5	4	85.0543	50.375
2	5	6	82.7717	50.977
2	5	8	86.6033	50.209
10	5	2	85.0272	77.497
10	5	4	83.288	119.693
10	5	6	88.9402	173.063
10	5	8	90.0815	217.65
15	5	2	85.4891	71.4505
15	5	4	86.7391	117.708
15	5	6	89.0489	171.178
15	5	8	90.4076	213.377
2	10	2	88.8315	225.037
2	10	4	89.2663	220.651

Tabla 35. Conjunto de Datos: Spambase, Algoritmo: SS relief.

maxIter	popSize	CR	F	%Aciertos	Tiempo
2	5	0.3	0.3	81.7391	9.086
2	5	0.5	0.3	81.9837	8.949
2	5	0.7	0.3	87.038	9.0155
2	5	0.3	0.5	87.1467	9.729
2	5	0.5	0.5	83.75	9.6045
2	5	0.7	0.5	87.6359	9.64
2	5	0.3	0.7	85.0815	9.757
2	5	0.5	0.7	85.8152	9.7085
2	5	0.7	0.7	86.1413	9.573
2	10	0.3	0.3	85.6522	18.357
2	10	0.5	0.3	87.7446	18.326
2	10	0.7	0.3	86.2772	18.3675
2	10	0.3	0.5	86.3043	18.314
2	10	0.5	0.5	88.4783	18.341
2	10	0.7	0.5	86.8207	18.609
2	10	0.3	0.7	88.288	18.573
2	10	0.5	0.7	88.3696	18.299
2	10	0.7	0.7	87.1739	18.3285
2	20	0.3	0.3	86.7391	36.074
2	20	0.5	0.3	87.4185	36.155
2	20	0.7	0.3	89.0761	36.1285
2	20	0.3	0.5	89.7283	36.025
2	20	0.5	0.5	89.6739	36.1575
2	20	0.7	0.5	86.4946	35.941
2	20	0.3	0.7	86.712	36.871
2	20	0.5	0.7	88.3696	36.173
2	20	0.7	0.7	86.5489	36.7635
15	5	0.3	0.3	86.3859	55.419
15	5	0.5	0.3	87.9076	56.584
15	5	0.7	0.3	90.7609	56.1965
15	5	0.3	0.5	86.8207	56.567
15	5	0.5	0.5	87.337	55.6105
15	5	0.7	0.5	91.0054	55.6065
15	5	0.3	0.7	89.9728	50.3865
15	5	0.5	0.7	89.5109	48.6895
15	5	0.7	0.7	91.2772	55.333
15	10	0.3	0.3	90.2717	106.697
15	10	0.5	0.3	90.1087	104.912
15	10	0.7	0.3	91.2772	105.558
15	10	0.3	0.5	91.1685	111.144
15	10	0.5	0.5	90.462	111.415
15	10	0.7	0.5	89.0217	109.785
15	10	0.3	0.7	90.5978	111.839
15	10	0.5	0.7	90.4348	111.267

Tabla 36. Conjunto de Datos: Spambase, Algoritmo: DE.

maxIter	NumGen	%Aciertos	Tiempo
2	3	72.7	2.502
10	5	73.9	7.55
15	10	73.875	6.7405
20	15	77.175	40.233

Tabla 37. Conjunto de Datos: Waveform, Algoritmo: LS aleatorio.

maxIter	NumGen	%Aciertos	Tiempo
2	3	79.55	2.358
10	5	79.325	3.7325
15	10	80.35	16.2985
20	15	80.825	35.447

Tabla 38. Conjunto de Datos: Waveform, Algoritmo: LS relief.

maxIter	NumGen	maxIterWC	%Aciertos	Tiempo
2	3	3	74.5	19.6815
10	5	3	76.125	71.5155
15	10	3	76.3	68.486
20	15	3	78.725	192.809
2	3	4	74.65	14.0385
10	5	4	74.425	34.826
15	10	4	76.25	163.803
20	15	4	77.15	185.736

Tabla 39. Conjunto de Datos: Waveform, Algoritmo: ILS aleatorio.

maxIter	NumGen	maxIterWC	%Aciertos	Tiempo
2	3	3	79.9	13.1505
10	5	3	80.075	30.7075
15	10	3	80.875	94.9805
20	15	3	80.8	102.454
2	3	4	80.525	13.2005
10	5	4	80.425	30.4605
15	10	4	80.95	92.4885
20	15	4	80.65	116.255

Tabla 40. Conjunto de Datos: Waveform, Algoritmo: ILS relief.

maxIter	NumGen	temp	estable	%Aciertos	Tiempo
2	2	100	100	72.725	4.778
2	4	100	100	73.875	10.123
10	2	100	100	76.6	18.3215
10	4	100	100	76.35	92.693
15	2	100	100	75.925	29.6205
15	4	100	100	78.325	145.279
20	2	100	100	75.975	38.55
2	2	50	100	72.525	7.8405
2	4	50	100	75.975	24.7035
10	2	50	100	73.125	24.609
10	4	50	100	78	116.287
15	2	50	100	75.975	34.5645
20	2	50	100	73.275	40.3585

Tabla 41. Conjunto de Datos: Waveform, Algoritmo: SA aleatorio.

maxIter	NumGen	temp	estable	%Aciertos	Tiempo
2	2	100	100	72.725	4.778
2	4	100	100	73.875	10.123
10	2	100	100	76.6	18.3215
10	4	100	100	76.35	92.693
15	2	100	100	75.925	29.6205
15	4	100	100	78.325	145.279
20	2	100	100	75.975	38.55
2	2	50	100	72.525	7.8405
2	4	50	100	75.975	24.7035
10	2	50	100	73.125	24.609
10	4	50	100	78	116.287
15	2	50	100	75.975	34.5645
20	2	50	100	73.275	40.3585

Tabla 42. Conjunto de Datos: Waveform, Algoritmo: SA relief.

maxIter	popSize	maxIterWC	%Aciertos	Tiempo
2	5	2	75.9	46.931
2	5	4	77.15	51.694
2	5	6	77.275	48.224
2	5	8	75.8	48.769
10	5	2	76.4	69.2705
10	5	4	77.4	111.748
10	5	6	78.45	150.754
10	5	8	78.5	195.281
15	5	2	77.3	68.6605
15	5	4	77.075	111.191
15	5	6	78.425	148.372
15	5	8	78.6	192.824
2	10	2	77.825	198.541
2	10	4	77.925	200.025

Tabla 43. Conjunto de Datos: Waveform, Algoritmo: SS aleatorio.

maxIter	popSize	maxIterWC	%Aciertos	Tiempo
2	5	2	75.9	46.931
2	5	4	77.15	51.694
2	5	6	77.275	48.224
2	5	8	75.8	48.769
10	5	2	76.4	69.2705
10	5	4	77.4	111.748
10	5	6	78.45	150.754
10	5	8	78.5	195.281
15	5	2	77.3	68.6605
15	5	4	77.075	111.191
15	5	6	78.425	148.372
15	5	8	78.6	192.824
2	10	2	77.825	198.541
2	10	4	77.925	200.025

Tabla 44. Conjunto de Datos: Waveform, Algoritmo: SS relief.

maxIter	popSize	CR	F	%Aciertos	Tiempo
2	5	0.3	0.3	74.1	7.825
2	5	0.5	0.3	75.075	7.9195
2	5	0.7	0.3	75.025	7.805
2	5	0.3	0.5	76.25	7.8385
2	5	0.5	0.5	75.675	8.225
2	5	0.7	0.5	75.45	8.0515
2	5	0.3	0.7	75.55	7.904
2	5	0.5	0.7	74.875	7.6895
2	5	0.7	0.7	74.85	7.614
2	10	0.3	0.3	75.875	14.8795
2	10	0.5	0.3	75.7	14.9525
2	10	0.7	0.3	76.425	14.8395
2	10	0.3	0.5	75.25	14.9235
2	10	0.5	0.5	75.475	14.8285
2	10	0.7	0.5	76.25	14.9455
2	10	0.3	0.7	76.05	14.9135
2	10	0.5	0.7	76.25	14.9685
2	10	0.7	0.7	76	14.8885
2	20	0.3	0.3	76.9	30.215
2	20	0.5	0.3	75.95	30.3315
2	20	0.7	0.3	76.3	30.0685
2	20	0.3	0.5	76.625	30
2	20	0.5	0.5	76.275	30.1685
2	20	0.7	0.5	75.475	30.0535
2	20	0.3	0.7	76.175	30.1715
2	20	0.5	0.7	75.85	30.1775
2	20	0.7	0.7	75.625	30.1455
15	5	0.3	0.3	77.25	46.796
15	5	0.5	0.3	77.825	45.5315
15	5	0.7	0.3	76.925	45.443
15	5	0.3	0.5	77.825	47.232
15	5	0.5	0.5	76.65	43.14
15	5	0.7	0.5	76.925	47.933
15	5	0.3	0.7	77	32.5205
15	5	0.5	0.7	76.775	44.626
15	5	0.7	0.7	76.1	29.344
15	10	0.3	0.3	78.275	94.8235
15	10	0.5	0.3	78.375	94.9545
15	10	0.7	0.3	78.75	96.0775
15	10	0.3	0.5	77.925	97.3145
15	10	0.5	0.5	77.775	97.537
15	10	0.7	0.5	78.05	87.164
15	10	0.3	0.7	77.6	90.7595
15	10	0.5	0.7	76.675	74.4325

Tabla 45. Conjunto de Datos: Waveform, Algoritmo: DE.

Experimento 2

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	95	5	0.000172
2	96.6667	3.33333	0.000446
3	95	5	0.000206
4	96.6667	3.33333	0.000172
5	98.3333	1.66667	0.000172
promedio	96.3333	3.66667	0.0002336
solucion: [1	1	1	1]

Tabla 46. Conjunto de Datos: Iris, Algoritmo: Sin pesos.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	95	5	0.001762
2	96.6667	3.33333	0.00176
3	93.3333	6.66667	0.001762
4	93.3333	6.66667	0.001747
5	93.3333	6.66667	0.001726
promedio	94.3333	5.66667	0.0017514
solucion: [0.109198	0.306227	1	0.327801]

Tabla 47. Conjunto de Datos: Iris, Algoritmo: Relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	96.6667	3.33333	0.007206
2	100	0	0.010492
3	98.3333	1.66667	0.003729
4	100	0	0.014075
5	98.3333	1.66667	0.007231
promedio	98.6667	1.33333	0.0085466
solucion: [0.578127	0.432077	0.421856	0.718341]

Tabla 48. Conjunto de Datos: Iris, Algoritmo: LS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	96.6667	3.33333	0.00718
2	98.3333	1.66667	0.007069
3	98.3333	1.66667	0.007158
4	98.3333	1.66667	0.007195
5	98.3333	1.66667	0.007166
promedio	98	2	0.0071536
solucion: [0.253759	0.302026	0.715629	0.463248]

Tabla 49. Conjunto de Datos: Iris, Algoritmo: LS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	98.3333	1.66667	0.022168
2	98.3333	1.66667	0.028861
3	98.3333	1.66667	0.035686
4	100	0	0.046238
5	98.3333	1.66667	0.039298
promedio	98.6667	1.33333	0.0344502
solucion: [0.666929	0.549467	0.676036	0.462461]

Tabla 50. Conjunto de Datos: Iris, Algoritmo: ILS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	98.3333	1.66667	0.036196
2	98.3333	1.66667	0.028689
3	98.3333	1.66667	0.035973
4	100	0	0.081547
5	100	0	0.046591
promedio	99	1	0.0457992
solucion: [0.258726	0.43725	0.674014	0.486377]

Tabla 51. Conjunto de Datos: Iris, Algoritmo: ILS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	98.3333	1.66667	0.106426
2	98.3333	1.66667	0.015299
3	98.3333	1.66667	0.019395
4	100	0	0.082493
5	100	0	0.124456
promedio	99	1	0.0696138
solucion: [0.42332	0.676786	0.625059	0.793053]

Tabla 52. Conjunto de Datos: Iris, Algoritmo: SA aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	98.3333	1.66667	0.056204
2	100	0	0.057864
3	98.3333	1.66667	0.080948
4	98.3333	1.66667	0.020825
5	100	0	0.065885
promedio	99	1	0.0563452
solucion: [0.171484	0.210903	0.54067	0.852287]

Tabla 53. Conjunto de Datos: Iris, Algoritmo: SA relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	98.3333	1.66667	0.339999
2	100	0	0.336462
3	98.3333	1.66667	0.330542
4	100	0	0.341162
5	98.3333	1.66667	0.343559
promedio	99	1	0.338345
solucion: [0.85759	0.592393	0.56226	0.517499]

Tabla 54. Conjunto de Datos: Iris, Algoritmo: SS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	98.3333	1.66667	0.334586
2	100	0	0.345401
3	98.3333	1.66667	0.333558
4	100	0	0.335266
5	98.3333	1.66667	0.343238
promedio	99	1	0.33841
solucion: [0.762056	0.671682	0.570012	0.507079]

Tabla 55. Conjunto de Datos: Iris, Algoritmo: SS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	96.6667	3.33333	0.00444
2	98.3333	1.66667	0.004386
3	98.3333	1.66667	0.004435
4	96.6667	3.33333	0.004465
5	96.6667	3.33333	0.004453
promedio	97.3333	2.66667	0.0044358
solucion: [0.488014	0.532067	0.281278	0.661454]

Tabla 56. Conjunto de Datos: Iris, Algoritmo: DE.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	77.1084	22.8916	0.001399
2	81.9277	18.0723	0.001391
3	75.9036	24.0964	0.001394
4	81.9277	18.0723	0.001448
5	87.9518	12.0482	0.001391
promedio	80.9639	19.0361	0.0014046

Tabla 57. Conjunto de Datos: Sonar, Algoritmo: Sin pesos.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	78.3133	21.6867	0.009954
2	78.3133	21.6867	0.009855
3	81.9277	18.0723	0.009792
4	77.1084	22.8916	0.009948
5	83.1325	16.8675	0.009889
promedio	79.759	20.241	0.0098876

Tabla 58. Conjunto de Datos: Sonar, Algoritmo: Relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	81.9277	18.0723	0.623256
2	87.9518	12.0482	0.623969
3	87.9518	12.0482	1.03882
4	86.747	13.253	0.417011
5	90.3614	9.63855	0.831244
promedio	86.988	13.012	0.706859

Tabla 59. Conjunto de Datos: Sonar, Algoritmo: LS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	84.3373	15.6627	1.03625
2	86.747	13.253	0.830281
3	84.3373	15.6627	0.624586
4	91.5663	8.43373	1.24647
5	90.3614	9.63855	0.831118
promedio	87.4699	12.5301	0.91374

Tabla 60. Conjunto de Datos: Sonar, Algoritmo: LS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	84.3373	15.6627	1.87742
2	92.7711	7.22892	5.60734
3	93.9759	6.0241	7.68041
4	90.3614	9.63855	5.80896
5	95.1807	4.81928	4.77883
promedio	91.3253	8.6747	5.15059

Tabla 61. Conjunto de Datos: Sonar, Algoritmo: ILS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	87.9518	12.0482	5.39431
2	90.3614	9.63855	4.776
3	91.5663	8.43373	3.9551
4	93.9759	6.0241	7.68466
5	95.1807	4.81928	8.09692
promedio	91.8072	8.19277	5.9814

Tabla 62. Conjunto de Datos: Sonar, Algoritmo: ILS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	91.5663	8.43373	5.54387
2	95.1807	4.81928	5.06552
3	95.1807	4.81928	5.63179
4	93.9759	6.0241	5.16609
5	98.7952	1.20482	5.47462
promedio	94.9398	5.06024	5.37638

Tabla 63. Conjunto de Datos: Sonar, Algoritmo: SA aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	90.3614	9.63855	5.31322
2	95.1807	4.81928	5.35089
3	92.7711	7.22892	6.02661
4	95.1807	4.81928	5.10518
5	97.5904	2.40964	5.24588
promedio	94.2169	5.78313	5.40835

Tabla 64. Conjunto de Datos: Sonar, Algoritmo: SA relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	89.1566	10.8434	3.55466
2	92.7711	7.22892	3.61533
3	91.5663	8.43373	3.59538
4	92.7711	7.22892	3.6235
5	92.7711	7.22892	3.53251
promedio	91.8072	8.19277	3.58428

Tabla 65. Conjunto de Datos: Sonar, Algoritmo: SS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	89.1566	10.8434	3.54879
2	87.9518	12.0482	3.58581
3	89.1566	10.8434	3.55036
4	93.9759	6.0241	3.6166
5	95.1807	4.81928	3.52895
promedio	91.0843	8.91566	3.5661

Tabla 66. Conjunto de Datos: Sonar, Algoritmo: SS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	84.3373	15.6627	0.539306
2	90.3614	9.63855	0.484297
3	86.747	13.253	0.263911
4	89.1566	10.8434	0.40158
5	92.7711	7.22892	0.346895
promedio	88.6747	11.3253	0.407198

Tabla 67. Conjunto de Datos: Sonar, Algoritmo: DE.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	81.4978	18.5022	0.006557
2	74.8899	25.1101	0.006608
3	81.0573	18.9427	0.006601
4	80.6167	19.3833	0.006574
5	79.2952	20.7048	0.006572
promedio	79.4714	20.5286	0.0065824

Tabla 68. Conjunto de Datos: WDBC, Algoritmo: Sin pesos.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	87.6652	12.3348	0.044485
2	86.3436	13.6564	0.044211
3	83.2599	16.7401	0.044527
4	89.4273	10.5727	0.044642
5	82.3789	17.6211	0.044419
promedio	85.815	14.185	0.0444568

Tabla 69. Conjunto de Datos: WDBC, Algoritmo: Relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	89.4273	10.5727	0.602471
2	93.8326	6.1674	1.78402
3	87.6652	12.3348	0.604509
4	92.9515	7.04846	1.19341
5	94.7137	5.28634	2.3709
promedio	91.7181	8.28194	1.31106

Tabla 70. Conjunto de Datos: WDBC, Algoritmo: LS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	90.3084	9.69163	0.405704
2	88.9868	11.0132	0.603737
3	84.141	15.859	0.408664
4	93.3921	6.60793	1.1935
5	92.0705	7.92952	1.38871
promedio	89.7797	10.2203	0.800063

Tabla 71. Conjunto de Datos: WDBC, Algoritmo: LS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	90.3084	9.69163	4.97282
2	92.511	7.48899	5.77275
3	93.3921	6.60793	8.95168
4	94.2731	5.72687	9.9416
5	94.2731	5.72687	12.8751
promedio	92.9515	7.04846	8.50279

Tabla 72. Conjunto de Datos: WDBC, Algoritmo: ILS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	95.1542	4.84581	7.7663
2	93.8326	6.1674	4.42216
3	90.7489	9.2511	5.62027
4	93.3921	6.60793	3.07075
5	96.0352	3.96476	12.9236
promedio	93.8326	6.1674	6.76061

Tabla 73. Conjunto de Datos: WDBC, Algoritmo: ILS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	92.0705	7.92952	4.52551
2	93.8326	6.1674	4.83991
3	92.511	7.48899	5.22042
4	94.2731	5.72687	5.7093
5	92.9515	7.04846	5.59812
promedio	93.1278	6.87225	5.17865

Tabla 74. Conjunto de Datos: WDBC, Algoritmo: SA aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	94.2731	5.72687	4.7021
2	93.8326	6.1674	4.11005
3	92.511	7.48899	6.21225
4	94.7137	5.28634	5.23623
5	96.9163	3.0837	4.86425
promedio	94.4493	5.55066	5.02498

Tabla 75. Conjunto de Datos: WDBC, Algoritmo: SA relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	93.3921	6.60793	17.0508
2	93.3921	6.60793	17.071
3	89.8678	10.1322	16.9464
4	92.0705	7.92952	17.2118
5	92.511	7.48899	17.1141
promedio	92.2467	7.7533	17.0788

Tabla 76. Conjunto de Datos: WDBC, Algoritmo: SS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	92.9515	7.04846	17.029
2	90.3084	9.69163	17.0275
3	89.8678	10.1322	17.1414
4	92.9515	7.04846	16.8977
5	92.9515	7.04846	17.066
promedio	91.8062	8.19383	17.0323

Tabla 77. Conjunto de Datos: WDBC, Algoritmo: SS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	91.63	8.37004	4.65517
2	90.7489	9.2511	5.32155
3	89.4273	10.5727	4.14282
4	86.3436	13.6564	1.91062
5	93.8326	6.1674	5.1788
promedio	90.3965	9.60352	4.24179

Tabla 78. Conjunto de Datos: WDBC, Algoritmo: DE.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	80.4348	19.5652	0.712625
2	79.837	20.163	0.719387
3	79.5109	20.4891	0.70446
4	80.5435	19.4565	0.714578
5	80.1087	19.8913	0.707981
promedio	80.087	19.913	0.711806

Tabla 79. Conjunto de Datos: Spambase, Algoritmo: Sin pesos.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	74.3478	25.6522	4.93467
2	74.0761	25.9239	4.73183
3	72.2826	27.7174	4.99992
4	79.2391	20.7609	4.89711
5	71.7391	28.2609	4.70345
promedio	74.337	25.663	4.8534

Tabla 80. Conjunto de Datos: Spambase, Algoritmo: Relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	90.7609	9.23913	140.254
2	80.4891	19.5109	33.0097
3	82.663	17.337	86.9464
4	93.2065	6.79348	191.334
5	91.087	8.91304	137.596
promedio	87.6413	12.3587	117.828

Tabla 81. Conjunto de Datos: Spambase, Algoritmo: LS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	82.3913	17.6087	108.385
2	87.8804	12.1196	156.496
3	79.7826	20.2174	214.823
4	90.2717	9.72826	187.683
5	74.8913	25.1087	53.8632
promedio	83.0435	16.9565	144.25

Tabla 82. Conjunto de Datos: Spambase, Algoritmo: LS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	89.6739	10.3261	117.134
2	89.8913	10.1087	184.471
3	87.8261	12.1739	76.36
4	80.0543	19.9457	31.6223
5	84.4022	15.5978	97.8148
promedio	86.3696	13.6304	101.48

Tabla 83. Conjunto de Datos: Spambase, Algoritmo: ILS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	83.6957	16.3043	149.919
2	75.3261	24.6739	45.6347
3	75.2174	24.7826	39.7025
4	87.0652	12.9348	64.5066
5	76.5217	23.4783	214.913
promedio	79.5652	20.4348	102.935

Tabla 84. Conjunto de Datos: Spambase, Algoritmo: ILS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	92.2826	7.71739	293.697
2	82.0109	17.9891	163.863
3	90.8696	9.13043	318.516
4	89.9457	10.0543	191.347
5	91.1957	8.80435	200.73
promedio	89.2609	10.7391	233.631

Tabla 85. Conjunto de Datos: Spambase, Algoritmo: SA aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	91.3043	8.69565	250.512
2	87.1196	12.8804	294.885
3	91.6848	8.31522	351.957
4	92.7174	7.28261	282.244
5	80.4348	19.5652	211.338
promedio	88.6522	11.3478	278.187

Tabla 86. Conjunto de Datos: Spambase, Algoritmo: SA relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	85.7609	14.2391	101.454
2	83.4239	16.5761	101.813
3	84.2935	15.7065	96.7718
4	91.1957	8.80435	106.709
5	86.25	13.75	103.161
promedio	86.1848	13.8152	101.982

Tabla 87. Conjunto de Datos: Spambase, Algoritmo: SS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	89.0761	10.9239	95.5262
2	84.4565	15.5435	95.1953
3	83.2065	16.7935	101.227
4	87.337	12.663	98.5232
5	83.913	16.087	92.3228
promedio	85.5978	14.4022	96.5588

Tabla 88. Conjunto de Datos: Spambase, Algoritmo: SS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	90	10	111.036
2	91.4674	8.53261	108.701
3	89.5652	10.4348	111.614
4	86.6304	13.3696	111.419
5	87.337	12.663	109.538
promedio	89	11	110.461

Tabla 89. Conjunto de Datos: Spambase, Algoritmo: DE.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	75.3	24.7	0.619373
2	75.3	24.7	0.60366
3	76.95	23.05	0.618277
4	76.2	23.8	0.611247
5	77.15	22.85	0.631247
promedio	76.18	23.82	0.616761

Tabla 90. Conjunto de Datos: Waveform, Algoritmo: Sin pesos.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	79.05	20.95	4.59823
2	79.3	20.7	4.6118
3	80.05	19.95	4.61729
4	78.65	21.35	4.53
5	80	20	4.49625
promedio	79.41	20.59	4.57072

Tabla 91. Conjunto de Datos: Waveform, Algoritmo: Relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	76	24	49.9766
2	74.15	25.85	37.7179
3	76.1	23.9	25.6078
4	73.55	26.45	19.8571
5	72.65	27.35	32.003
promedio	74.49	25.51	33.0325

Tabla 92. Conjunto de Datos: Waveform, Algoritmo: LS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	80.9	19.1	70.1857
2	80.35	19.65	31.8059
3	80.75	19.25	13.4468
4	79.85	20.15	19.5674
5	80.75	19.25	32.3149
promedio	80.52	19.48	33.4641

Tabla 93. Conjunto de Datos: Waveform, Algoritmo: LS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	71.25	28.75	28.6148
2	73.2	26.8	28.5639
3	75.3	24.7	39.962
4	73.75	26.25	24.1764
5	72.9	27.1	20.3085
promedio	73.28	26.72	28.3251

Tabla 94. Conjunto de Datos: Waveform, Algoritmo: ILS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	80.35	19.65	27.808
2	80	20	48.8887
3	80.95	19.05	51.4484
4	80.2	19.8	49.5898
5	80.65	19.35	48.0654
promedio	80.43	19.57	45.16

Tabla 95. Conjunto de Datos: Waveform, Algoritmo: ILS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	74.7	25.3	8.58915
2	71.4	28.6	6.08966
3	71.55	28.45	6.10144
4	73.1	26.9	6.22676
5	74.8	25.2	8.39475
promedio	73.11	26.89	7.08035

Tabla 96. Conjunto de Datos: Waveform, Algoritmo: SA aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	80.45	19.55	9.97953
2	79.3	20.7	3.69251
3	80.4	19.6	37.078
4	78.8	21.2	16.0471
5	80	20	3.71935
promedio	79.79	20.21	14.1033

Tabla 97. Conjunto de Datos: Waveform, Algoritmo: SA relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	76	24	90.4146
2	77.15	22.85	94.071
3	76.95	23.05	90.8371
4	78.85	21.15	87.2551
5	78.15	21.85	90.7283
promedio	77.42	22.58	90.6612

Tabla 98. Conjunto de Datos: Waveform, Algoritmo: SS aleatorio.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	79.2	20.8	88.3571
2	79.35	20.65	91.3485
3	80.45	19.55	85.4704
4	79.95	20.05	86.2325
5	79.2	20.8	87.3997
promedio	79.63	20.37	87.7616

Tabla 99. Conjunto de Datos: Waveform, Algoritmo: SS relief.

Particion	Aciertos(%)	Error(%)	Tiempo(sg)
1	75.5	24.5	15.8365
2	76.05	23.95	15.8633
3	74.65	25.35	15.841
4	73.4	26.6	16.019
5	77.45	22.55	15.6708
promedio	75.41	24.59	15.8461

Tabla 100. Conjunto de Datos: Waveform, Algoritmo: DE.