Metaheurísticas

P1 - Búsqueda por trayectorias para el problema de la selección de características

Tercero Grado Ing. Informática

Francisco Carrillo Pérez
Grupo 1 jueves a las 17.30

Contents

1 Abstract

El problema de la selección de características consiste en minimizar el número de características necesarias para obtener un buen porcentaje de aciertos en la predicción de distintas clases en problemas de aprendizaje automático. En nuestro caso hemos utilizado como clasificador el 3-NN que se basa en la clase de los tres vecinos más cercanos.

2 Definición matemática

$$tasa\_clase(3-NN(s))=100*\frac{n^{O}deinstanciasbienclasificadasdeT}{n^{O}totaldeinstanciasdeT}$$

Donde:

- $s = \{0, 1\}, 1 \leq i \leq n$, es decir, un vector binario de tamaño n que define si una característica ha sido seleccionada o no.
- 3-NN es el clasificador que utilizaremos, que obtiene tres vecinos utilizando las características seleccionadas.
- Nos devuelve el porcentaje de aciertos que ha obtenido nuestro clasificador con esas características.

3 Intención de la selección de características

Con la selección de características nuestro objetivo es reducir el número de características necesarias para ser usadas en un problema de clasificación. Esto nos permite quitar características redundantes o que no aporten ningún tipo de valor, reduciendo así el tiempo de ejecución para la clasificación.

4 Aplicación de los algoritmos empleados para el problema

4.1 Descripción del esquema de representación de soluciones

Las soluciones se presentan como un vector binario de N posiciones, siendo este el número de atributos de la base de datos correspondiente. Si el valor de la posición está a 0, significa que esta característica no es considerada, mientras que si se encuentra a 1 esa característica se cuenta en consideración.

4.2 Generación de un vecino

El pseudocódigo de la generación de un vecino nuevo es la siguiente:

```
subproceso funcion generarVecinoRandom
para i = 0; i < num_atributos-1; i++
        num-aleatorio = generamos un número aleatorio
        si num-aleatorio >= 0.5
             cambiamos 0 por 1 o 1 por 0
```

4.3 Cambio de una posición del vector de atributos

El pseudocódigo de la función flip, usada para cambiar el valor de un elemento del vector de atributos, sería la siguiente:

```
subproceso funcion flip
si el valor de la posición es 0
    valor de la posición = 1
en caso contrario
    valor de la posición = 0
```

5 Algoritmos implementados

5.1 Greedy SFS

El pseudocódigo de este algoritmo es el siguiente:

```
subproceso funcion greedy
mientras que mejoremos el exito o hayamos recorrido la lista de atributos
     para cada atributo, probamos uno a uno cuál produce mayor éxito
           si el max. éxito es menor que el éxito producido
              actualizamos el éxito al máximo que hemos encontrado
     Si el mejor éxito anterior es mejor que el mejor éxito conseguido
           ponemos la característica a 1 y actualizamos el mejor éxito
     En caso contrario acabamos
     Si hemos recorrido todos los atributos acabamos
Devolvemos la mejor solución obtenida
```

5.2 Búsqueda Local Primer Mejor

El pseudocódigo de este algoritmo es el siguiente:

```
subproceso funcion primerMejor
Generamos solución actual aleatoria
Obtenemos el éxito actual
Mientras que encontremos mejoras y no se hayan evaluado 15000 soluciones distintas
    Mientras que la solución nueva sea mejor que la actual o se haya generado
    todo el vecindario de la solución actual
           Obtenemos una solución nueva
           Aumentamos  el número de soluciones
           Obtenemos el éxito de esa solución
           Si el éxito obtenido es mejor que el actual o hemos generado todo el
           vecindario de la solución actual
                Salimos del bucle
    Si el éxito nuevo es mejor que el éxito actual
           Actualizamos el éxito y la solución
    Si el éxito nuevo es menor o igual que el éxito actual o se han evaluado
    15000 soluciones distintas
           Termina
Devolvemos la mejor solución obtenida
```

5.3 Enfriamiento Simulado

La inicialización de la temperatura inicial se ha realizado con la fórmula:

$$T_0 = \frac{\mu * C(S_0)}{-ln(\phi)}$$

Siendo $\mu = \phi = 0.3$.

La actualización de la temperatura se ha realizado siguiendo el esquema de Cauchy modificado que nos indica que:

$$T_{k+1} = \frac{T_k}{1 + \beta * T_k}$$

, siendo β:

$$\beta = \frac{T_0 - T_f}{M * T_0 * T_f}$$

El pseudocódigo de este algoritmo es el siguiente:

```
subproceso funcionn anneal
Generamos solución actual aleatoria
Obtenemos éxito actual
```

```
Inicializamos la temperatura incial y la mínima
Si la temperatura inicial es menor que la mínima
    Reducimos al temperatura mínimca
Mientras que acepte soluciones nuevas y el número de evaluaciones no sea 150000
    Mientras que generemos mejores soluciones, aceptemos por el criterio de
    metrópolis o no se haya generado todo el vecindario de la solución actual
        Obtenemos una solución jueva
        Aumentamos el número de soluciones
        Obtenemos el éxito de esa solución
        Si el éxito es mejor que el actual
           Salimos
        en caso contrario si aceptamos una solución por el criterio de
        metrópolis
           Reducimos la temperatura y salimos
        Si hemos generado todo el vecindario o el número de evaluaciones es
        15000
           Salimos y reducimos la temperatura
    Si el éxito nuevo es mejor que el global
        Actualizamos el éxito global y la solución global
    Si hemos aceptado la solución por el criterio de metrópolis  o el éxito
    nuevo es mejor que el actual
        Actualizamos la solucion y el éxito
    Si no hemos aceptado una solución nueva o hemos realizado 15000
    evaluaciones
        Salimos
Devolvemos la mejor solución global
```

6 Implementación

La implementación se ha realizado en C++, tomando como referencia algunas cosas del código proporcionado, sobre todo en el Simulated Annealling.
En la carpeta de software se proporciona un Makefile que compila el main. En el archivo src/main.cpp se puede comentar y descomentar dependiendo del algoritmo y la base de datos que queramos utilizar.
En la función **introducirDatos** se define que datos se van a coger con el último elemento:

- Si ponemos un 0, cogeremos la base de datos **arrythmia**.
- Si ponemos un 1, cogeremos la base de datos **movement-libras**.
- Por último, si ponemos un 2, cogeremos la base de datos **wdbc**.

Lo mismo ocurre con la función **dividirDatos**, el último elemento define de la forma en que dividimos los datos:

- SI ponemos un 0, dividimos los datos en dos conjuntos justo por la mitad.
- Si ponemos un 1, dividimos los datos introduciendo los atributos pares en un conjunto y los impares en otro.
- Si ponemos un 2, dividimos los conjuntos dividiendo los elementos intercaladamente de dos en dos.
- Si ponemos un 3, dividimos los conjuntos dividiendo los atributos intercaladamente de tres en tres.
- Por último, si ponemos un 4, dividimos los conjuntos dividiendo los atributos intercaladamente de cinco en cinco.

7 Resultados

La semilla utilizada para todos los experimentos ha sido **111** Los resultados obtenidos son los representados en las siguientes tablas:

	Wdbc			Movement	Libras		Arrhy	thmia	
	%_Clas	%_red	T	%_Clas	%_red	T	%_Clas	%_red	T
Partición 1-1	88	0	**0.33s**	1	0	0,25s	67	0	0.54s
Partición 1-2	88	0	0.33s	1	0	0.25s	67	0	0.54s
Partición 2-1	93	0	0.34s	68	0	0.25s	63	0	0.56s
Partición 2-2	93	0	0.34s	68	0	0.25s	63	0	0.56s
Partición 3-1	92	0	0.33s	68	0	0.25s	65	0	0.57s
Partición 3-2	92	0	0.33s	68	0	0.25s	65	0	0.57s
Partición 4-1	92	0	0.37s	76	0	0.24s	66	0	0.56s
Partición 4-2	92	0	0.37s	76	0	0.24s	66	0	0.56s
Partición 5-1	94	0	0.34s	67	0	0.24s	65	0	0.56s
Partición 5-2	94	0	0.34s	67	0	0.24s	65	0	0.56s
MEDIA	91.8	0	0.346	56	0	0.246	65.2	0	0.558

Figure 1: Resultados obtenidos por el alg. 3NN en el problema de la SC

	Wdbc			Movement	Libras		Arrhy	thmia	
	%_Clas	%_red	T	%_Clas	%_red	T	%_Clas	%_red	T
Partición 1-1	89.47	10	**3m46.510s**	2.22	0	23m7.391s	77.7202	1.79	503m56.865s
Partición 1-2	89.47	10	3m46.050s	2.22	0	23m8.391s			
Partición 2-1	92.98	6.7	3m48.704s	83.88	11.1	21m56.819s	76.1658	1.79	494m52.487s
Partición 2-2	92.98	6.7	3m48.227s	83.88	11.1	21m45.819s			
Partición 3-1	92.98	10	3m50.012s	77.77	11.1	21m54.176s	76.6839	1.79	507m4.567s
Partición 3-2	92.98	10	3m49.567s	77.77	11.1	21m53.180s			
Partición 4-1	92.98	10	3m49.821s	80.55	22.2	21m58.300s	75.1295	1.79	510m4.567s
Partición 4-2	92.98	10	3m49.360s	80.55	22.2	21m59.251s			
Partición 5-1	92.98	10	3m50.931s	76.11	8.8	22m19.147s	62.1762	0.36	506m3.45s
Partición 5-2	92.98	10	3m50.931s	76.11	8.8	22m20.186s			
MEDIA	92.278	7.34	3m49.009s	64.106	19.64	21m6s	73.5751	1.5	504m4.703s

Figure 2: Resultados obtenidos por el alg. Greedy en el problema de la SC

	Wdbc			Movement	Libras		Arrhy	thmia	
	%_Clas	%_red	T	%_Clas	%_red	T	%_Clas	%_red	T
Partición 1-1	89.47	15.56	**0m9.553s**	0.56	54.55	0m0.617s	63.73	44.96	0m4.602s
Partición 1-2	89.47	15.56	0m9.523s	0.56	54.44	0m0.634s	63.73	44.96	0m4.580s
Partición 2-1	92.28	50	0m1.551s	76.66	54.44	0m0.641s	62.69	50.71	0m1.728s
Partición 2-2	92.28	50	0m1.527s	76.66	54.44	0m0.617s	62.69	50.71	0m1.653s
Partición 3-1	92.98	50	0m1.847s	69.44	54.44	0m0.630s	62.69	51.43	0m2.739s
Partición 3-2	92.98	50	0m1.817s	69.44	54.44	0m0.652s	62.69	51.43	0m2.720s
Partición 4-1	92.98	50	0m9.468s	77.22	54.44	0m0.646s	62.69	47.48	0m4.627s
Partición 4-2	92.98	50	0m9.468s	77.22	54.44	0m0.646s	62.69	47.48	0m4.654s
Partición 5-1	93.33	50	0m1.558s	69.44	53.33	0m2.246s	64.76	51.79	0m5.640s
Partición 5-2	93.33	50	0m1.533s	69.44	53.33	0m2.230s	64.76	51.79	0m5.610s
MEDIA	92.16	43.11	0m4.78s	58.06	54.229	0m0.955s	63.312	49.274	0m3.855s

Figure 3: Resultados obtenidos por el alg. BL en el problema de la SC

	Wdbc			Movement	Libras		Arrhy	thmia	
	%_Clas	%_red	T	%_Clas	%_red	T	%_Clas	%_red	T
Partición 1-1	89.47	50	**0m10.004s**	0.56	55.56	0m18.898s	71.50	48.9	6m0.326s
Partición 1-2	89.47	50	0m10.000s	0.56	55.56	0m18.893s	71.50	48.9	6m0.326s
Partición 2-1	93.68	60	0m12.402s	81.66	54.44	0m56.567s	67.35	49.28	5m19.053s
Partición 2-2	93.68	60	0m12.397s	81.66	54.44	0m56.540s	67.35	49.28	5m18.897s
Partición 3-1	93.68	53.33	0m15.735s	71.66	54.44	0m37.021s	64.76	45.68	3m10.195s
Partición 3-2	93.68	53.33	0m15.735s	71.66	54.44	0m37.021s	64.76	45.68	3m10.067s
Partición 4-1	92.98	50	0m9.519s	78.33	55.55	0m39.862s	72.02	45.68	5m17.894s
Partición 4-2	92.98	50	0m9.513s	78.33	55.55	0m39.830s	72.02	45.68	5m17.740s
Partición 5-1	94.73	53.33	0m15.938s	71.11	55.55	0m45.300s	67.35	49.28	5m33.198s
Partición 5-2	94.73	53.33	0m15.933s	71.11	55.55	0m45.267s	67.35	49.28	5m33.198s
MEDIA	92.9	53.332	0m12.71s	60.66	55.108	0m38.51s	68.59	47.76	4m16.08

Figure 4: Resultados obtenidos por el alg. Simulated Annealing en el problema de la SC

Table 1: Comparación media de todos los algoritmos

	Wdbc			Movement Libras			Arrythmia		
	%_Clas	%_red	T	%_Clas	%_red	T	%_Clas	%_red	T
3-NN	91.8	0	0.346s	56	0	0.246s	65.2	0	0.558s
SFS	92.278	7.34	3m49s	64.106	19.64	21m6s	73.57	1.5	504m7s
BL	92.16	43.11	4.78s	58.06	54.229	0.955s	63.312	49.274	3.855s
ES	92.9	53.332	12.71s	60.66	55.108	38.51s	68.59	47.46	4m16.08s

Lo primero que observamos son los resultados obtenidos por el **clasificador 3-NN** sin ninguna reducción de características. Los resultados son bastante buenos, sobre todo por la tasa que se

obtiene, y el tiempo de ejecución. El problema que se encuentra, y que se repite en el resto de algoritmos, es en la primera forma de separar los datos con la base de datos del **Movement Libras**. Si separamos los datos dividiendo por la mitad, en el archivo movement_libras se encuentran ordenados de forma que todas las muestras de una clase se encuentran juntas, seguidas de otra clase, todas juntas también. Por ello al entrenar, estamos entrenando con un conjunto de muestras que son de ciertas clases, pero el test no tiene muestras que sean de esas clases de las que hemos entrenado, si no de totalmente diferentes, por ello la poca tasa de acierto que se repite en todos los algoritmos.

El siguiente algoritmo que nos encontramos es el **Greedy**. Este obtiene tasas de reducción de características no muy altas. Además, los tiempos de ejecución con la base de datos de **Arrythmia** son desorbitadas, por ello solo se ha ejecutado de una forma la ejecución, ya que la media de tiempo de ejecución eran unas 9 horas. Podemos observar que tenemos el mismo problema detallado anteriormente en en la base de datos **movement_libras**. Con respecto a esta base de datos, mejora bastante lo obtenido con el clasificador 3-NN, sin reducir mucho las características.

Con el algoritmo **Búsqueda Local Primer Mejor**, obtenemos un mejora inmesa en los tiempos de ejecución con respecto al Greedy. Esto es algo destacable, porque por ejemplo, con la base de datos **arrythmia** pasamos de tiempos de ejecución de media de 9 horas, a tiempos de media de **0.3 segundos**. Además en las tres bases de datos reducimos de media hasta la mitad de las características, obteniendo tasas de éxito más bajas en algunos casos, pero vale la pena por la reducción de características y los tiempos de ejecución que hemos obtenido.

Por último, con el algoritmo de **Enfriamiento Simulado**, obtenemos las mismas mejoras que obtenía el algoritmo de Búsqueda Local, pero obteniendo aún mejores resultados que este último de media, excepto en los tiempos, pero el aumento no es significativo para la ganancia que obtenemos en los otros apartados.

Ahora voy a añadir una serie de gráficas para ver de forma visual las diferencias:

Los tiempos son en **segundos**.

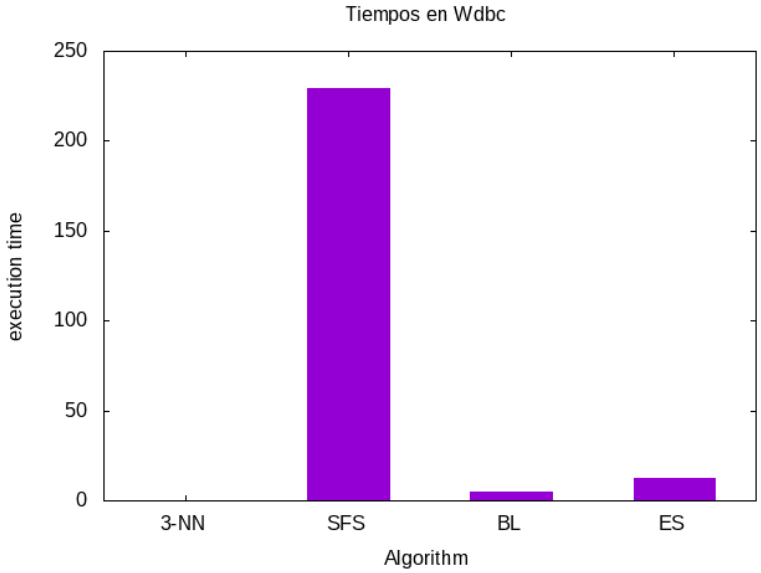

Figure 5: Diferencia de tiempos en Wdbc

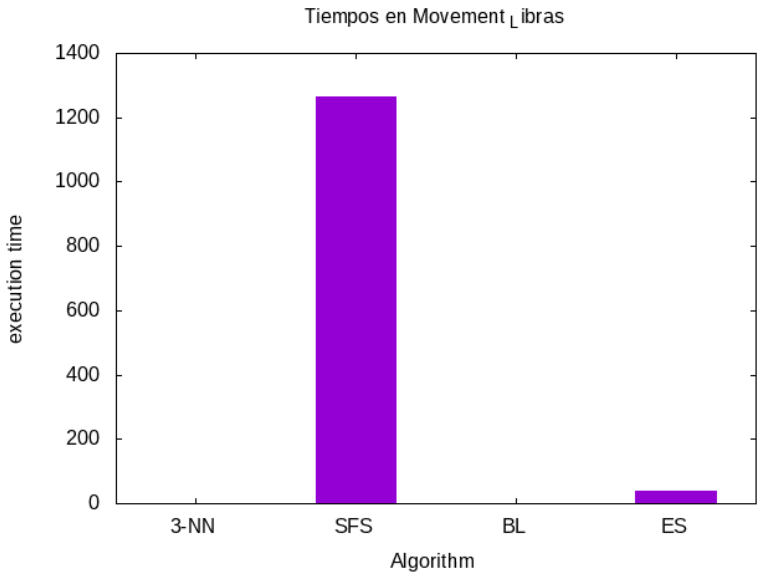

Figure 6: Diferencia de tiempos en Movement Libras

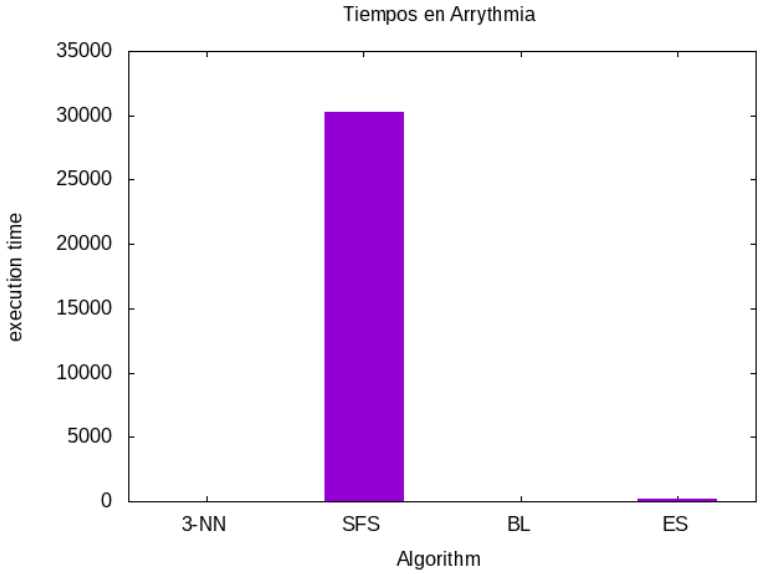

Figure 7: Diferencia de tiempos en Arrythmia

Podemos observar la gran diferencia entre el SFS y el resto de algoritmos.

Ahora vamos a comparar la tasa de reducción:

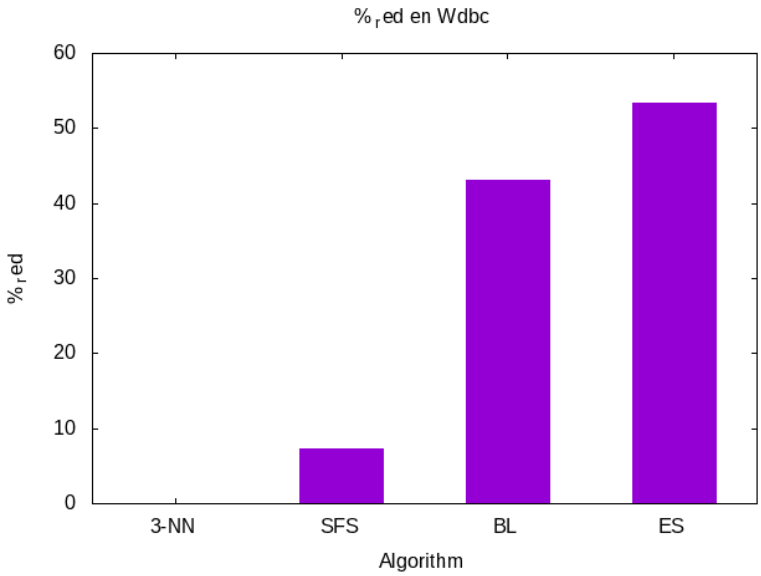

Figure 8: Diferencia de %_red en Wdbc

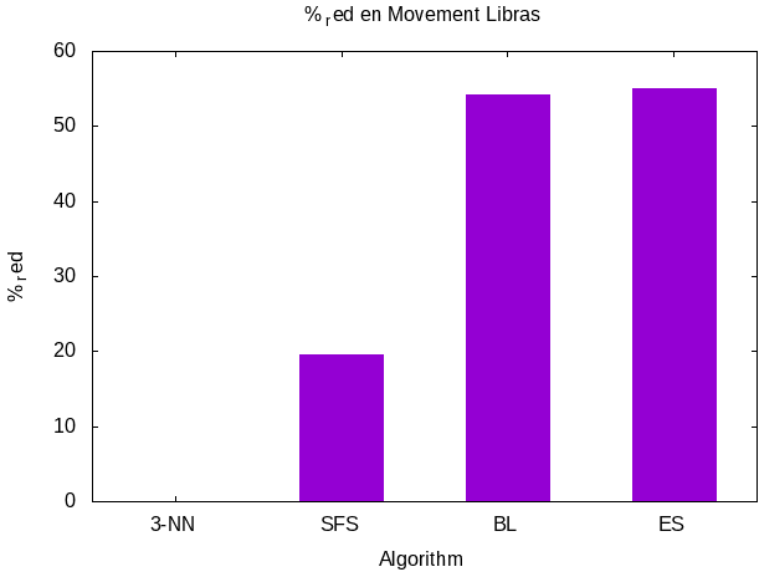

Figure 9: Diferencia de %_red en Movement Libras

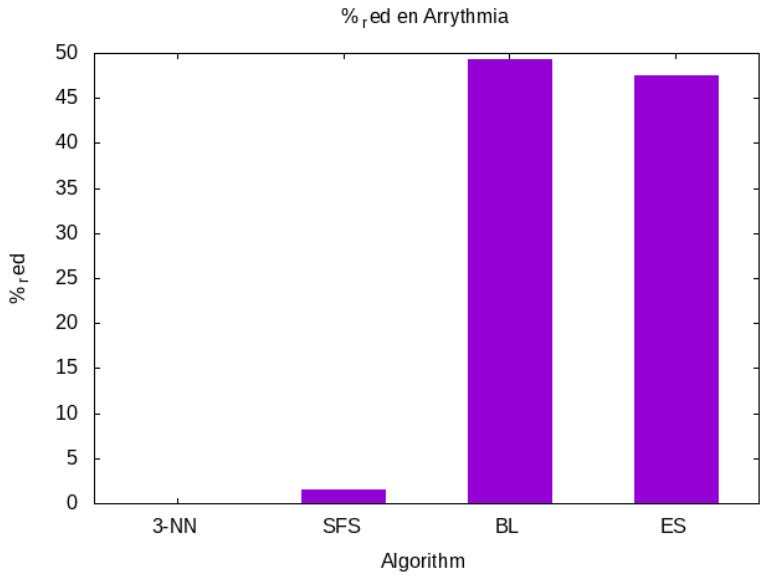

Figure 10: Diferencia de %_red en Arrythmia

De media el que obtiene mejor tasa de reducción es el algoritmo de ES.

Por último, vamos a comprar la tasa de clasificación:

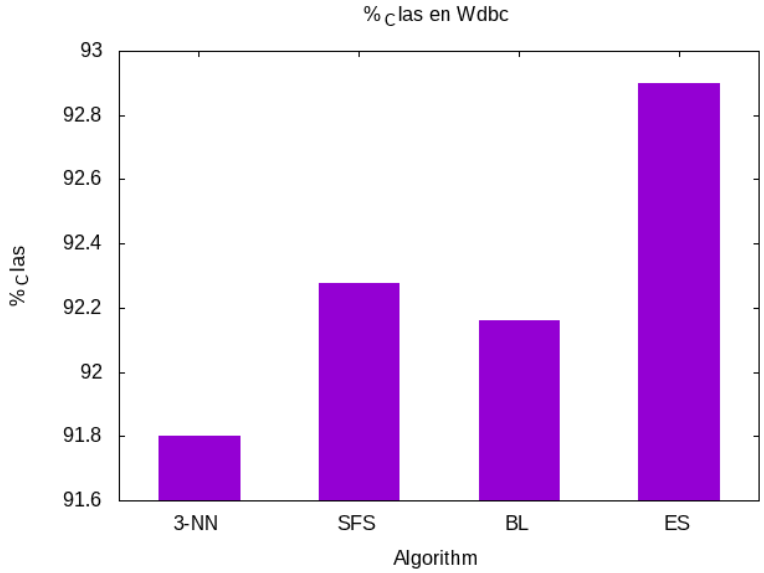

Figure 11: Diferencia de %_Clas en Wdbc

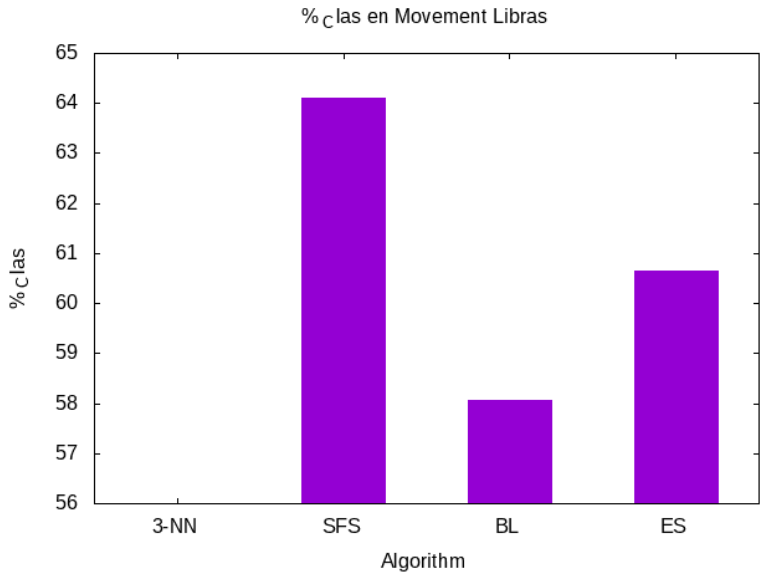

Figure 12: Diferencia de %_Clas en Movement Libras

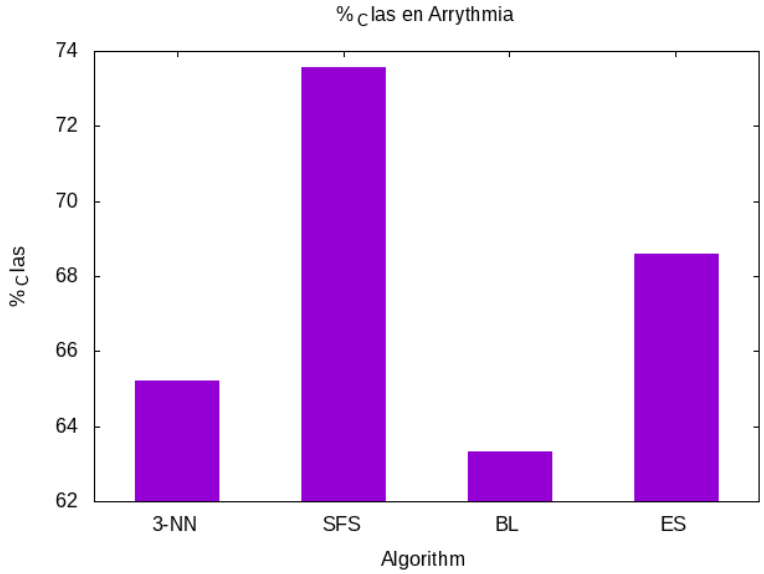

Figure 13: Diferencia de %_Clas en Arrythmia

8 Conclusiones Finales

Para concluir, querría añadir que en caso de tener que elegir uno de estos algoritmos para resolver un problema, utilizaría sin duda el Enfriamiento Simulado, ya que es con el que he conseguido resultados muy buenos tanto de clasificación como reducción de características(en el que es el mejor) en un tiempo muy reducido de ejecución.