Trabajo Final de Máster

creación de variables sintéticas mediante evolución gramatical en problemas de clasificación

Roberto Canduela Luengo

Máster en Inteligencia Artificial Avanzada

Profesores: Enrique Carmona / José Luis Aznarte

Septiembre 2017

© Copyright de Roberto Canduela Luengo, 2017.

Todos los derechos reservados.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

El presente estudio aborda el requerimiento de un trabajo final del Máster en Inteligencia Artificial Avanzada, impartido por la UNED.

El objetivo perseguido con la investigación realizada, expuesta en las siguientes páginas, ha sido abordar la creación de variables "sintéticas" en problemas de clasificación. Entendiendo una variable "sintética" como la que se forma a partir de las existentes en un conjunto de datos, mediante la combinación de estas y distintos operadores matemáticos.

El objetivo de la creación de estas nuevas variables es el de ampliar el espacio de características del problema inicial, bajo la idea de que un aumento en la expresividad de las variables puede redundar en un mejor aprendizaje del clasificador, y por tanto en unos mejores resultados. Consideramos que el trabajo viene a profundizar en algunos planteamientos existentes en la bibliografía sobre creación de variables, así como responde a los resultados obtenidos en la práctica a la hora de modelar problemas mediante clasificadores, donde se ha podido comprobar que la creación de nuevas variables puede mejorar los resultados.

El objetivo planteado se ha abordado desde la perspectiva de los algoritmos evolutivos, en concreto desde el punto de vista de la evolución gramatical, campo en el que casi no existe bibliografía sobre su uso en el problema planteado.

El trabajo se ha dividido en dos partes:

- La primera parte es una introducción al estado del arte en la que se exponen los trabajos que se han desarrollado en el ámbito de la creación y selección de variables en general, y más concretamente centrados en el entorno de los algoritmos evolutivos y en particular en la evolución gramatical. Realizaremos un repaso a las diferentes técnicas de selección y creación

de variables, comprobando que el número de trabajos es claramente asimétrico, contando con multitud de investigaciones en el ámbito de la selección de variables y con relativamente pocas en el campo de la creación de nuevas variables. Abordaremos estos dos puntos, aunque nuestro estudio se centra solo en la creación de variables, dado que en ocasiones van emparejados, puesto que la creación de variables necesita de una selección previa de variables del conjunto de datos.

- La segunda parte aborda los experimentos realizados para intentar conseguir una metodología lo más general posible, que permita abordar el problema de creación de variable "sintéticas" en cualquier conjunto de datos y para cualquier clasificador.

Los trabajos realizados en torno a la construcción de variables mediante evolución gramatical presentan posibilidades de desarrollo en los puntos:

- Identificación de una metodología clara y transparente y de un sistema de medición de resultados que aporte validez estadística.
- Prueba con diferentes tipos de clasificadores.
- Prueba de operadores.
- Comparación de los resultados usando las nuevas variables creadas frente al conjunto de variables iniciales.
- Manejo de variables continuas y discretas.
- Prueba de funcionamiento de la metodología en un caso de negocio real.

Hay que destacar que en todo momento se ha buscado demostrar la validez estadística de los resultados obtenidos, punto que no se encuentra en algunas de las referencias bibliográficas consultadas.

El elevado coste computacional de los experimentos realizados ha impedido, en algunas ocasiones, lograr una mayor profundidad en el análisis, no obstante consideramos que se ha realizado un esfuerzo considerable que permite dar una perspectiva general de las diferentes metodologías propuestas bajo diferentes condiciones.

La creación de variables "sintéticas" ofrece, bajo el enfoque de la evolución gramatical, un interesante campo de estudio que consideramos puede aportar resultados de interés a la hora de abordar problemas de clasificación.

Capítulo 2

Trabajos relacionados

Aunque la variedad de temas que abarca el presente proyecto es amplia podemos centrar principalmente los puntos de mayor relevancia en dos:

- Selección de variables
- Construcción de nuevas variables

En las siguientes secciones haremos un repaso a algunos de los estudios existentes sobre los dos puntos mencionados. Cabe destacar que nos centraremos en la mejora de los resultados en modelos de clasificación supervisada, mediante la creación de nuevas variables sintéticas.

En la clasificación supervisada se busca un modelo que sea capaz de maximizar el acierto a la hora de asignar en l categorías las observaciones de una muestra de test, fijadas estas en la fase de entrenamiento por una variable conocida a priori que emplearemos en el aprendizaje sobre la muestra de entrenamiento. Haremos referencia por defecto a variables dependientes discretas, en cualquier otro caso, como por ejemplo en problemas de regresión, se explicitará.

Como notación vamos a emplear para los datos de entrada $[x_{ij}, y_k]$, con m casos de $i = 1..m$ y n variables de $j = 1..n$, x_{ij} es el valor del i-ésimo caso para la j-ésima variable que pertenece a la clase y_k, las clases podrán tomar los valores $k = 1..l$.

Emplearemos la notación F_0 para referirnos al conjunto inicial de las n variables y F_t a los diferentes subconjuntos de variables que puedan ser seleccionados.

2.1. Selección de variables

La selección de variables es una parte fundamental dentro de las técnicas de manipulación que ayudan a mejorar los resultados del modelado, así como un posible paso previo a la creación de nuevas variables.

Partimos de la base de que todas las variables que describen un problema no son igualmente relevantes, contando con variables irrelevantes y redundantes en los datos, que no solo no aportan información para la mejora de la capacidad predictiva del algoritmo de clasificación, sino que pueden introducir ruido, empeorando su rendimiento.

El objetivo deseable será seleccionar las variables más "relevantes", eliminando aquellas con alta correlación o mucho ruido y que no aportan una mejora significativa al usarse en el proceso de modelado. Entendemos el término "variable" como una propiedad individual del proceso que estamos observando [16]. De la observación de un fenómeno pueden surgir múltiples variables que se emplearán en problemas de clasificación, tanto supervisada como no supervisada.

En las últimas décadas hemos pasado de disponer de unas decenas de variables, para el modelado de problemas, a las miles o decenas de miles que podemos tener actualmente, por ejemplo en el modelado de textos [17] o el de genes. Esto hace que cobre una especial relevancia la selección previa de variables, atendiendo a los posibles beneficios en las fases de modelado [23] tendremos:

1. **Mejora del rendimiento**, al reducir el ruido y eliminar variables irrelevantes, disminuyendo el riesgo de sobreajuste.

2. **Reducción de la dimensión** del problema y por tanto del tiempo de ejecución del entrenamiento y la extrapolación, junto con una menor ocupación de los datos. Por extrapolación hacemos referencia a la clasificación de nuevas observaciones no empleadas en la fase de aprendizaje del modelo.

3. **Ayuda a un mejor entendimiento** del problema y una mejor visualización al reducir la dimensión.

Se ha demostrado que un excesivo número de variables con alta correlación entre ellas y ruido puede disminuir la capacidad predictiva de los algoritmos de clasificación. En el caso de los modelos estadísticos clásicos la condición de independencia es restrictiva para muchos de ellos, por ejemplo en la regresión lineal. Para resolver este problema se ha recurrido históricamente a técnicas como el PCA [4] (Análisis de Componentes Principales) o al Análisis Factorial. Estas técnicas generan un conjunto de variables incorreladas además de una posible reducción en la dimensión del problema a costa de

una pérdida de información. Las trataremos con más detalle en el apartado de transformación de variables.

El problema de selección de variables requerirá del uso de una medida de aporte o calidad de cada una de las variables, o de un subconjunto de ellas, sobre la variable objetivo y_k. Este problema es extremadamente costoso en términos de computación, dado que disponemos de (2^n) posibles subgrupos definibles a partir del conjunto inicial de n variables, por tanto deberemos emplear procesos sub-óptimos para abordar el problema, de forma que sea asumible computacionalmente.

Una clasificación básica de los métodos de selección de variables se puede encontrar en [16] y en [45]:

- **Filter:** Los métodos de filtrado realizan una ordenación de la aportación, de mayor a menor, de cada variable individual sobre la variable objetivo y_k.
- **Wrapper:** La discriminación se realiza en función del rendimiento del clasificador sobre y_k para cada subconjunto F_t de F_0.
- **Embedded:** En este caso la selección de variables es parte del algoritmo de clasificación, como por ejemplo en el caso de los árboles de decisión.
- **Hybrid:** Combina la filosofía de los métodos de Filtering y Wrapper, empleando propiedades de la distribución de los datos.

A continuación describiremos brevemente cada uno de los métodos mencionados.

2.1.1. Filter Method

Este métodos es el más extendido, debido a la simplicidad de implementación y a los buenos resultados que ofrece.

Se denomina método de filtrado dado que después de realizar la ordenación se efectúa una eliminación de variables en base a un criterio de corte, asumiendo que una única variable, por si misma, contiene información relevante para la clasificación [39].

Estos métodos son independientes del clasificador que se vaya a usar, lo que les dota de una gran capacidad para adaptarse a la mayor parte de los problemas. Pueden emplear diferentes medidas como el coeficiente de correlación de Pearson, criterios de entropía, chi cuadrado... la elección de la medida para la realización de la ordenación puede estar condicionada por el tipo de variables: continuas, discretas, categóricas.

Este método, no obstante, presenta el problema, para la mayoría de las métricas existentes, de no tener en cuenta posibles interacciones entre variables.

Métricas y puntos de corte

Hemos mencionado los principales métodos empleados en la selección de variables, la mayoría de ellos necesitan decidir de forma previa a su utilización [3] la métrica de aportación que se va a emplear y el método de corte.

Los diferentes métodos existentes pueden necesitar de la transformación de las variables para poder ejecutarse, algunos solo trabajan con variables continuas y otros solo lo hacen con discretas. Esto hace necesaria una adecuación de los datos de entrada previa a la ejecución de la métrica de evaluación de las diferentes variables o conjuntos de variables.

Métrica de aportación

A continuación presentaremos algunos de los criterios más habituales para la medición de la relevancia de variables de forma individual [3], se puede ver un cuadro muy exhaustivo en [45]

Hay que tener en cuenta que las métricas de aportación pueden ser sensibles al dominio de las variables, puede ser necesario realizar una normalización antes de su uso.

1. **Mutual Information:** También conocido como ganancia de información, mide la cantidad de información que una variable aporta sobre una clase. Se define como la entropía de la clase y la entropía de la clase condicionada al conocimiento de la variable evaluada [4].

$$I(x, y) = H(x) + H(x \mid y)$$

   donde $H(x)$ es la entropía marginal y $H(x \mid y)$ es la entropía condicional

2. **Gain Ratio:** Se define como el ratio entre la ganancia de información y la entropía de la variable. Este medida fue la empleada en el desarrollo del algoritmo C4.5 [41].

$$\text{Gain Ratio} = \frac{I(y, x)}{H(y)}$$

3. **Gini Index:** Se puede entender como la probabilidad de que dos registros seleccionados aleatoriamente sean de distinta clase. Se emplea en los árboles tipo CART [28].

$$\text{Gini Index} = \sum_{i,j \in C; i \neq j} p(i \mid y) p(j \mid y)$$

Método de corte

El método de corte y la elección del valor concreto a partir del cual aceptamos o rechazamos variables del dominio es fundamental, en [3] podemos ver un estudio completo con múltiples conjuntos de datos evaluados, distintas métricas de aportación, y diferentes métodos de corte y valores empleados. Los autores llegan a la conclusión de que no es fácil realizar una generalización sobre cual es el mejor método y sus parámetros, dado que cada conjunto presenta una combinación óptima que no tienen porque ser la mejor para el resto.

Podemos enumerar los diferentes métodos de corte analizados en la bibliografía para problemas de selección de variables con evaluaciones una a una, sin tener en cuenta interacciones.

- **Número fijo:** Seleccionamos un número fijo de variables, atendiendo solo a la ordenación de las mismas en función de la métrica de aportación. Este sistema tiene la pega de que es difícil generalizar un número de variables a priori, pudiendo en algunos casos seleccionar variables con baja aportación y en otros casos dejando fuera a variables con aportaciones significativas.
- **Fracción:** Seleccionamos un porcentaje de variables en función de su ordenación por la métrica de aportación. Presenta una pega similar al "Fixed number", dado que a priori es complicado definir el parámetro para una generalidad de problemas.
- **Umbral:** Selecciona las variables que están por encima de un determinado umbral de la métrica de aportación. Presenta el mismo problema que los dos anteriores aunque en menor medida, dado que aquí la experiencia sí nos puede servir para identificar en cada métrica umbrales de aportación que sean significativos.
- **Diferencia:** Empieza a seleccionar las variables en función de su ordenación, hasta que la diferencia de la evaluación está por encima de un umbral.
- **Pendiente:** Selecciona de la ordenación las mejores variables hasta que la pendiente con la siguiente variable esté por encima de un umbral.

2.1.2. Wrapper Method

La discriminación se realiza en función del rendimiento del clasificador, evaluando todas las posibles combinaciones de variables sobre los datos de validación, habiendo divido los datos de en-

trenamiento en entrenamiento y validación. Se emplea una medida del acierto para seleccionar el mejor conjunto de variables. La evaluación exhaustiva de todos los posibles subconjuntos es computacionalmente inviable con una dimensionalidad no muy grande, por ello se han diseñado heurísticas que permiten reducir el espacio de búsqueda. Al depender de un algoritmo de clasificación para obtener una medida de la calidad del conjunto F_t, el valor puede variar dependiendo del algoritmo empleado. Hay que tener en cuenta que el uso del clasificador, en la selección del conjunto F_t, tiene la ventaja de encontrar el espacio de variables óptimo para el clasificador que emplearemos en la fase de aprendizaje, maximizando el rendimiento del mismo.

Dentro de está técnica se pueden emplear diferentes algoritmos de búsqueda:

- **Búsqueda completa**: búsqueda sobre todo el conjunto de variables.
- **Búsqueda secuencial**: búsqueda secuencial.
- **Búsqueda aleatoria**: búsqueda aleatorizada sobre el conjunto de variables.

2.1.3. Embedded Method

En este caso la selección de variables forma parte del algoritmo de clasificación, que combina la selección de variables y el proceso de aprendizaje. Ejemplos de ello son los árboles de decisión o los bosques de árboles. Otro ejemplo son las SVM [33], donde podemos configurar el algoritmo para minimizar el número de variables empleadas junto con el error [23]. De esta forma es el propio algoritmo el que, mediante una métrica, como por ejemplo el Gain Ratio del C4.5 [41] o el Índice Gini [28], determina qué variables formarán parte de la fase de aprendizaje.

2.1.4. Hybrid Method

Combinan los métodos de Filtering, para podar el espacio de variables de forma rápida y eficiente, y las estrategias de búsqueda local de Wrapper para encontrar el subconjunto óptimo F_t.

2.1.5. Conclusiones

Hay una gran variedad de métodos de selección de variables en los que intervienen diferentes métricas de aportación y sistemas para fijar las variables o conjuntos de variables seleccionados. Hay un equilibrio, como es habitual, entre los resultados obtenidos y el coste computacional, los métodos de filtrado ofrecen buenos rendimientos con bajo coste, trabajando únicamente con aportaciones individuales de las variables. Los métodos más sofisticados que tienen en cuanta interacciones entre

variables presentan, no obstante, tiempos de ejecución elevados y se usan con menos frecuencia en la práctica que los métodos de filtrado.

Los experimentos realizados en métodos de filtrado con distintas métricas [3] no parecen encontrar una generalización que permita identificar un sistema óptimo sobre una amplia variedad de ejemplos. Se deduce, por tanto, que la elección de la métrica y los parámetros de corte están muy relacionados con los datos de entrada.

2.2. Creación de variables

En todo proceso de clasificación supervisada partimos de un espacio de variables que contiene toda la información conocida sobre el dominio, y una variable objetivo que clasificará los diferentes registros en categorías. La materia prima con la que trabajará el modelo de clasificación serán las variables de entrada y por tanto condicionarán la calidad del modelo obtenido. Por tanto una buena representación del dominio del problema será de vital importancia.

En esta sección abordaremos el problema de como emplear operadores para transforma una o varias variables iniciales con el objetivo de generar variables derivadas con mejores propiedades, que posibiliten transformar el espacio de representación en uno nuevo donde las regularidades de los datos sean mejor detectadas [15]. Partiríamos por tanto de un conjunto inicial de variables F_0 sobre el que emplearemos alguna de las técnicas de selección de variables vistas en el punto anterior, una vez seleccionada la variable o variables buscaríamos uno o varios operadores que aplicaríamos sobre ellas. Los métodos de construcción pueden buscar diferentes propósitos, pudiendo buscarse solo alguno de ellos [37]:

- **Mejora de la capacidad predictiva del modelo de aprendizaje:** en estos casos es habitual que el conjunto de variable finales sea superior al inicial
- **Reducción de la dimensión del espacio de variables:** en estos casos el conjunto de variables finales debe ser menor que el inicial.
- **Mejora en la interpretabilidad de las variables.**

De los tres puntos anteriores los dos primeros son los que centran principalmente la atención de los investigadores. La mejora de la adecuación del modelo resultante a los datos de partida suele repercutir en la calidad de la clasificación. La reducción de la dimensión del espacio de variables cobra mucha importancia en problemas en los que es necesario procesar una gran cantidad de variables

como por ejemplo en visión artificial o procesamiento de textos. En concreto en el caso de visión artificial puede ser más crucial si es necesario un procesamiento en tiempo real, donde el coste computacional puede hacer que no sea factible el sistema planteado.

Deberemos tener en cuenta además que la generación de nuevas variables puede agravar problemas ya existentes en las variables originales o hacer que aparezcan otro nuevos. Los problemas habituales de un conjunto de variables son:

- **Irrelevancia de variables:** es posible que dentro de los datos que usemos para la modelización del problema haya variables que no aportan información, penalizan el tiempo de ejecución, y además introducen ruido, lo que puede perjudicar el aprendizaje del clasificador.
- **Interacción de variables:** La falta de dependencia de las variables puede ser un grave problema dependiendo del clasificador empleado, introduciendo ruido. Hay estudios que demuestran que modelar con este tipo de variables degrada la capacidad predictiva del modelo por ejemplo en árboles de decisión [18] como el C4.5 [41].

En la figura 2.1 mostramos un resumen básico del proceso de construcción de variables. Partiendo de un conjunto inicial F_0 empleamos un método de selección para encontrar el conjunto F_i, de una o más variables, sobre el que realizaremos las transformaciones. Realizamos la transformación a través de los operadores prefijados $\phi(F_i)$ obteniendo un conjunto de variables de salida F_t sobre el que analizaremos su utilidad, midiendo la aportación sobre el clasificador. El proceso se puede repetir de forma iterativa mientras existan diferentes conjuntos que seleccionar en F_0 o mientras estos conjuntos cumplan con el criterio de selección de variables fijado.

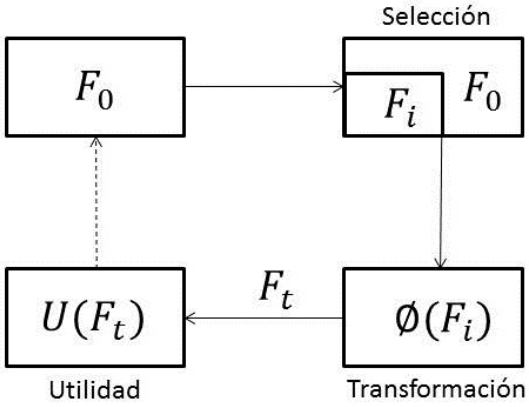

Figura 2.1: Flujo construcción de variables

Los métodos de construcción de variables se pueden dividir en [15]:

- **Impulsados por hipótesis (Hypothesis Driven):** Se construye una hipótesis y a partir de ella se generan las nuevas variables. Se añaden las variables construidas al conjunto original y se procede a revisar la hipótesis para construir una nueva. El proceso se repite de forma iterativa y los resultados tienen una gran dependencia de las hipótesis formuladas.

- **Impulsados por los datos (Data driven):** Trabaja sobre los datos para construir nuevas variables sin realizar ninguna hipótesis. Una muestra de ello es el algoritmo GALA [26] que solo emplea operadores AND y OR sobre variables booleanas y que podría considerarse un precursor del uso de algoritmos evolutivos aunque no los emplee, como sí hace evolución gramatical [25].

También hay que tener en cuenta, en el proceso de creación de variables, el criterio de evaluación de las variables generadas. En este sentido podemos identificar dos métodos distintos [15]:

- **Enfoque de entrelazado (Interleaving Approach):** En esta metodología se emplea el algoritmo de clasificación para evaluar la calidad de las nuevas variables generadas. Por tanto los resultados podrían no ser generalizables a otros algoritmos de aprendizaje.

- **Enfoque de preprocesado (Preprocessing Approach):** En esta metodología se evalúa la calidad de los atributos directamente sobre los datos, sin emplear un algoritmo de clasificación en el proceso.

Los operadores empleados en $\phi(F_i)$ dependerán del tipo de la variables con las que trabajemos, podemos hacer una división en:

- **Variables lógicas:** operadores del tipo conjunción, disyunción, negación.

- **Variables nominales:** M de N, producto cartesiano.

- **Variables numéricas:** min, max, media...

La elección de operadores es fundamental, una mala selección puede generar variables irrelevantes degradando los resultados del modelo.

Se han propuesto diferentes métodos que repasaremos a continuación, centrándonos en los de reducción de la dimensión del espacio de variables y métodos para la mejora de la predicción. Fundamentalmente buscarán [37]:

- Generar nuevas variables que mejoren el rendimiento.

- Los métodos deberán ser computacionalmente eficientes.

- Deberán ser generalizables a diferentes clasificadores.
- Deberán permitir añadir conocimiento al dominio de manera sencilla.

2.2.1. Transformación de variables

El objetivo de estos métodos es conseguir un nuevo conjunto de variables, partiendo de las variables iniciales, tal que $|F_i| < |F_0|$. Ya hemos mencionado algunas de las aplicaciones de estos métodos en visión artificial o en procesamiento de textos. Hay que destacar que en este caso se tratará de algoritmos no supervisados, dado que no disponemos de una variable que contenga a priori las categorías.

Agrupación

Los algoritmos de agrupación o cluster habitualmente empleados son los jerárquicos [29] y el de K-medias [29], aunque su aplicación habitual es en la agrupación de registros también se pueden emplear a la hora de agrupar variables.

En los algoritmos jerárquicos partimos de todas las variables y vamos agrupándolas hasta conseguir llegar a un solo grupo que contendrá F_0, tendremos que decidir en qué punto deseamos cortar la agrupación determinando el número de variables nuevas que vamos a generar. Un ejemplo del uso de cluster jerárquico, en la reducción de dimensiones, lo podemos encontrar en [6] aplicado a la reducción de variables en análisis espectral.

A diferencia de los clusters jerárquicos el algoritmo de k-medias necesita especificar a priori el número de grupos que se deséan generar.

PCA

El PCA [29] (Principal Component Analysis) se basa en realizar una proyección, mediante transformaciones lineales, sobre unos nuevos ejes de coordenadas (componentes principales), buscando la proyección que mejor represente los datos en términos de mínimos cuadrados. La elección de los ejes se realiza seleccionando aquellos que explican un mayor parte de la variabilidad de los datos. Hay que destacar que los ejes son ortogonales entre sí y por tanto independientes. La independencia de las componentes principales generadas es de gran ayuda a la hora de emplear algoritmos como la regresión lineal que requieren la independencia de las variables de entrada. Hay diferentes técnicas para la selección de las componentes principales que finalmente serán empleadas para representar al conjunto inicial de variables, como:

- **Porcentaje explicado:** Consiste en fijar, a priori, un porcentaje de varianza explicada por las componentes. De estas forma se seleccionarán tantas componentes como sean necesarias para llegar a dicho porcentaje. La selección se hará por orden de aportación de cada componente.
- **Criterio de Kaisser:** se seleccionan únicamente las componentes principales con valores mayores o iguales a 1.
- **Criterio Gráfico:** visualmente se evalúan las pendientes en el gráfico de aportaciones de todas las componentes, ordenadas de mayor a menor aportación, de esta forma se seleccionan las componentes hasta que se observa que al pasar de una a otra no hay una pendiente significativa.

Como resumen indicaremos que si tenemos $[X_1, ..., X_p]$ variables las componentes principales serán variables compuestas e incorreladas. Idealmente unas pocas explicarían una gran parte de la variabilidad de los datos. Definiendo las componentes como:

$$Y_1 = Xt_1, Y_2 = Xt_2, ..., Y_p = Xt_p$$

Donde $T = [t_1, t_2, ..., t_p]$ es la matriz $p \times p$, cuyas columnas definen las componentes principales. Entonces tenemos la transformación por componentes principales de la forma:

$$Y = XT$$

Donde $t_1, t_2, ..., t_p$ son los p vectores propios normalizados de la matriz de covarianzas S [29].

SVD

El SVD [29] (Singular Value Descomposition) se emplea en la reducción de datos mediante el uso de "factores" que explican las correlaciones entra las variables. Partiendo de las variables iniciales y mediante combinaciones lineales y un error se generan los diferentes factores. Los campos en los que ha tenido una mayor aplicación han sido las ciencias sociales y márketing.

Si consideramos $X_1, ..., X_p$ variables, y un modelo de un solo factor que aglutina la variabilidad de todas las variables y p factores $U_1, ..., U_p$ uno por cada variable, tendríamos:

$$X_i = \alpha_i F + d_i U_i \quad i = 1, .., p$$

Por tanto cada variable X_i depende del factor común F y de un factor único U_i.

En el caso de tener m factores tendríamos:

$$X_1 = a_{11}F_1 + ... + a_{1m}F_m + d_1U_1$$

$$X_2 = a_{21}F_1 + ... + a_{2m}F_m + d_2U_2$$

...

$$X_p = a_{p1}F_1 + ... + a_{pm}F_m + d_pU_p$$

Para una mejor interpretabilidad de los factores se suele recurrir a rotaciones como las ortogonales y oblicuas.

2.2.2. Construcción de variables

Estos métodos tienen como objetivo generar nuevas variables que permiten obtener un mayor conocimiento del dominio, lo que ayudará a mejorar el rendimiento de los clasificadores. Partiremos de un conjunto de variables inicial F_0 y llegaremos a un conjunto F_n con $|F_n| > |F_0|$. Ya se ha mencionado la importancia que tiene en este tipo de métodos la selección de variables inicial y la de los operadores que se emplearan en la transformación, si no se hace esta selección de manera correcta se generan variables que incorporaran ruido a los datos con la consiguiente pérdida de capacidad predictiva.

Hay una gran variedad de técnicas para la selección de variables (sección 2.1) y para la combinación de los distintos operadores definidos con las variables seleccionadas, a continuación mencionamos algunos de ellos.

Métodos directos

Vamos a comentar alguno de estos métodos para el tratamiento de variables continuas, son muy usados en estadística y consisten en la aplicación de transformaciones individuales a las variables para conseguir una mejora en la predicción. Son muy habituales el uso de los operadores $\{\sqrt{x_i}, log(x_i), \frac{1}{x_i}, ...\}$ Estas transformaciones se basan principalmente en el análisis visual de las variables y sobre todo se han aplicado a problemas de regresión, su utilidad en problemas de clasificación es menor.

Podemos destacar también los métodos de Box-Cox [35] para la transformación de variables. Esta familia de transformaciones busca corregir la no linealidad en la relación de dos variables. La

transformación potencial es una función continua que varia en función de la potencia de λ, se pueden expresar como:

$$Y_i^\lambda = \begin{cases} K_1(Y_i^\lambda - 1) & \text{si} \quad \lambda \neq 1 \\ K_2 Ln(Y_i^\lambda) & \text{si} \quad \lambda = 0 \end{cases}$$

donde

$$K_2 = \left(\prod_{i=1}^{n} Y_i \right)^{\frac{1}{n}}$$

y

$$K_1 = \frac{1}{\lambda K_2^{\lambda - 1}}$$

Árboles de decisión

Hay diversos algoritmos para la creación de variables basados en el uso de árboles de decisión [37] . La estructura del árbol, con su capacidad de introducir variables de forma sistemática, evaluándolas a través de una función de utilidad como el Gain Ratio [41], y de combinarlas mediante operadores se adapta particularmente bien al problema de generar nuevas variables.

Una de los primeros algoritmos que aparecieron fue el FRINGE de Pagallo en 1989 [19]. En cada iteración se construyen nuevas variables combinando dos variables iniciales, empleando únicamente operadores NOT y AND. En una línea similar podemos citar también los algoritmos CITRE y DC FRINGE [37] que aparecieron poco después del FRINGE.

Más recientemente encontramos el FICUS de Markovitch y Rosenstein de 2002 [31]. Proporciona un conjunto de funciones constructoras (operadores) junto con las características. El algoritmos enriquece el espacio de características agregando las nuevas variables generadas. Las funciones constructoras que proponen pueden ser uno a varios de los operadores comunes o pueden ser proporcionados por un experto en el dominio.

Por último mencionaremos los algoritmos WoE (Weight of Evidence) [2], basados en árboles de decisión su objetivo es construir nuevas variables a partir de la "tramificación" de variables individuales. Se introduce una sola variable independiente y la variable dependiente en el árbol, de forma que este parta de manera óptima la variable independiente. Dependiendo de qué algoritmo de árbol empleemos la métrica a través de la que se genera el corte variará, por ejemplo en el caso de CART [28] se emplea el Índice de Gini y en el C4.5 [41] el Gain Ratio.

En general los árboles de decisión solo evalúan, a la hora de incorporar variables, su aportación individual, lo que limita su capacidad para tener en cuenta posibles interacciones entre variables y el posible beneficio obtenido en la construcción de variables al contar con ello.

ILP

Los métodos ILP (Inductive Logic Programming) [37] se usan para desarrollar descripciones de conocimiento, permitiendo incorporar conocimiento experto en el proceso de creación de variables. Estos métodos se emplearon por primera vez en el algoritmos LINUS desarrollado por Lavrac en 1991 [32]. Se suele aplicar una aproximación en dos pasos:

- **Construcción de variables:** se emplea el ILP para generar un nuevo conjunto de características.
- **Selección de variables:** se selecciona un subconjunto de variables basado en su utilidad en el proceso de predicción.

Procesos basados en anotaciones

Los métodos basados en anotaciones permiten obtener información del dominio a través de los datos de entrenamiento. En 2009 Roth y Small propusieron un nuevo protocolo interactivo para la construcción del espacio de variables, IFSC [44]. En vez de predefinir un espacio de características grandes mediante el uso de operadores su enfoque permite un espacio dinámico de características basado en la interacción entre la máquina de aprendizaje y un experto en el dominio. En la fase de entrenamiento se le presentan los casos al experto que genera anotaciones en forma de etiquetas para mejorar el conocimiento.

Computación Evolutiva

En el presente trabajo nos basaremos en las técnicas evolutivas para la creación de nuevas variables, por tanto mostramos las principales tendencias en este campo en un sección específica.

2.2.3. Sumarización de variables

Por sumarización o agregación de datos entendemos el proceso por el cual pasamos de tener m registros para una variable a generar n con $n < m$.

Un ejemplo claro de este proceso lo tenemos cuando recogemos información de una variable en diferentes momentos temporales, para la variables estudiada contaremos con m observaciones

recogidas en $\{t_1, t_2, ..., t_m\}$, el proceso de sumarización consiste en emplear una función sobre las observaciones para reducir la dimensión intentando perder la menor información posible. Ejemplos clásicos de funciones de agregación son: sumatorio, media, mediana, moda...

La aplicación de técnicas de sumarización automáticas es un campo muy poco explorado, podemos citar el trabajo de Alfred Rayner [38] en 2008, en el que presenta su algoritmo DARA. Mediante una matriz de pesado de frecuencias y la aplicación de técnicas de clusterización genera nuevas variables que agregan información de las variables iniciales que presentan múltiples registros. Esta aproximación al problema de sumarización aplica técnicas de clustering para la agregación de datos en tablas sin variable objetivo con alta cardinalidad [38]. También presenta una aproximación para la construcción de nuevas variables de agregación mediante algoritmos genéticos, de nuevo para conjuntos de datos sin variable objetivo.

2.3. Computación Evolutiva en la construcción de nuevas variables

La Computación Evolutiva engloba a una familia de metaheurísticas que se caracterizan por emplear mecanismos de genética evolutiva para encontrar soluciones óptimas o pseudo óptimas a una gran variedad de problemas. La aplicación de este tipo de mecanismos evolutivos cómo herramienta de optimización fue introducida por Holland [21] en los años setenta del siglo pasado. El esquema general [14] de un proceso de optimización evolutiva es:

- Partimos de una población que contiene n individuos, generada de forma aleatoria o a través de heurísticas.
- Cada individuo es una posible solución del problema.
- Determinamos una función de evaluación que mide el nivel de adaptación de cada uno de los individuos al problema, esto es, cómo de buena es la solución generada por el individuo.
- Se seleccionan individuos de la población inicial que serán los progenitores de la siguiente generación.
- Se genera la siguiente generación a través de operadores de selección y cruce.
- Se evoluciona la nueva generación a través de operadores de mutación.
- Se seleccionan los individuos que van a forma parte de la nueva generación.

Este proceso se repite de forma iterativa hasta que se cumple alguna de las condiciones de parada: óptimo alcanzado por debajo de un determinado valor, número de iteraciones igual a un valor prefijado... Los algoritmos genéticos tienen una gran capacidad para acercarse a valores óptimos de forma rápida, lo que les confiere un rango de aplicabilidad muy grande. Al trabajar con múltiples factores probabilísticos, como son la probabilidad de selección, cruce, mutación... los algoritmos puede presentar notables variaciones entre diferentes ejecuciones lo que hace necesaria la repetición del experimento un número de veces suficiente como para dotar a los datos obtenidos de validez estadística. El ecosistema de algoritmos surgidos bajo el paradigma de la Computación Evolutiva es muy amplio, algunos ejemplos son:

- **Algoritmos genéticos [21].**
- **Estrategias evolutivas [48].**
- **Programación genética [40].**
- **Algoritmos meméticos [14].**
- **Evolución gramatical [8].**

Algunos de estos algoritmos se han usado para resolver el problema de la construcción de nuevas variables, a partir de un conjunto inicial, con el objetivo de mejorar el rendimiento del clasificador. La estructura básica de estos algoritmos, al contar con parte probabilística, se adapta muy bien al problema de generar nuevas variables, dado que no sería abordable desde un punto de vista exhaustivo al tratarse de un problema NP-Hard. La capacidad de los algoritmos evolutivos para combinar soluciones y evolucionar los hace candidatos ideales para afrontar este complicado problema. A continuación nos centraremos en las principales familias de algoritmos evolutivos empleadas en la resolución del problema de creación de nuevas variables, comentando los resultados obtenidos. No obstante destacaremos que aunque se han empleado estos algoritmos en múltiples ocasiones el problema de construcción de variables no ha sido abordado con la prodigalidad que, por ejemplo, ha tenido el de selección de variables, por tanto es un campo en el que pensamos que aún hay camino por recorrer y posible margen de mejora.

2.3.1. Programación genética

La Programación Genética es un tipo especial de Computación Evolutiva ideada por Koza [40] en la década de los noventa del siglo pasado, que se adapta al método general expuesto, con la

particularidad de que los individuos están representados por árboles. Está singularidad hace que se ajuste particularmente bien al problema de generación de variables, dado que permite combinar en una estructura de árbol variables y operadores. Sobre la estructura de árbol se aplican los operadores ya mencionados, cruce y mutación, de forma similar que en el resto de algoritmos de Computación Evolutiva.

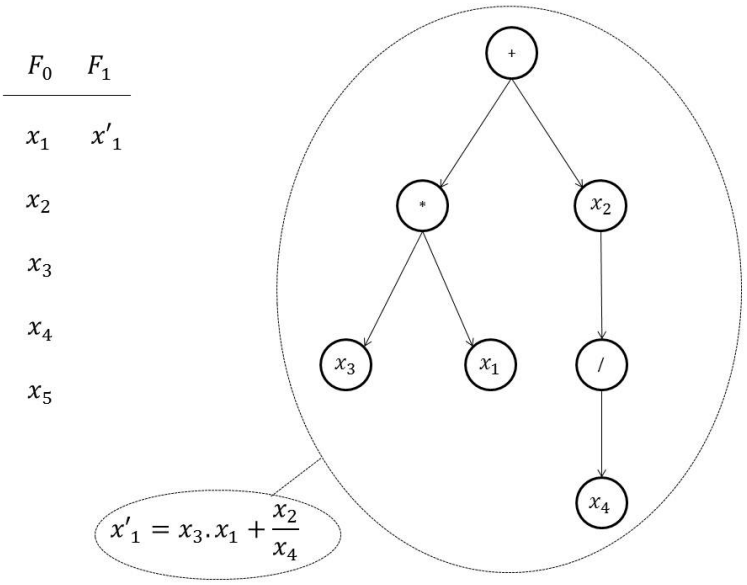

Figura 2.2: Programación Genética árbol de construcción de variables

En la figura 2.2 se muestra un ejemplo de como, a través de la Programación Genética, se pueden genera nuevas variables. Partimos de un conjunto inicial de variables $F_0 = \{x_1, x_2, x_3, x_4, x_5\}$ y construimos la nueva variable x'_1 a través de una combinación de variables iniciales y operadores. En este ejemplo hemos empleado los operadores básicos $\{+, -, *, /\}$.

Uno de los primeros trabajos sobre la creación de nuevas variables que emplean Programación Genética es el de Krawiec [27] de 2002. Este estudio presenta la particularidad de que compara un método general con un método extendido. El método extendido intenta evitar la destrucción de soluciones, por la propia dinámica del crecimiento del árbol en Programación Genética. Para paliar los efectos nocivos trabaja con un genotipo extendido por el que cada individuo se divide en dos partes disjuntas, compuestas por:

- ***Fevol:*** las variables que se encuentran en este subconjunto están sujetas a un proceso evolutivo.

- ***Fhid:*** a este conjunto pasan las mejores variables encontradas en Fevol para preservarlas de la posible degeneración, se compara sistemáticamente este conjunto contra Fevol para determinar que variables deben formarlo.

El objetivo que se persigue con este método es preservar las mejores soluciones evitando su deterioro al sufrir evoluciones reiteradas, de esta forma aislamos las mejores soluciones y evolucionamos el resto. En caso de encontrar mejores soluciones que las almacenadas se procede a hacer un reemplazo en el conjunto "Fhid" y se continúa con el proceso evolutivo.

Krawiec [27] emplea los operadores $\{+, -, *, \%, log, <, >, =, IF\}$ tanto, siendo $\%$ la división protegida.

En el artículo de Otero, Silva, Freitas y Nievola de 2003 [15] se trabaja en la línea de controlar el tamaño máximo del árbol para evitar problemas de sobreajuste y el efecto "bloat"que aparece en la Programación Genética (tendencia de los árboles a crecer de forma incontrolada). Sin embargo no limitan la profundidad del árbol sino que se centran en determinar el número máximo de nodos, lo que permite al árbol crecer en vertical y en horizontal.

El algoritmo de Otero, Silva, Freitas y Nievola de 2003 [15] permite crear nuevos atributos continuos y booleanos dependiendo de los operadores empleados. Destacar que como medida de la calidad de las variables generadas emplean el Gain Ratio [41].

En la línea del trabajo de Krawiec [27] el texto de Smith y Bull [30] de 2005 presenta algunas evoluciones destacables:

- No está limitado el numero de nuevas variables a cuatro como en el planteamiento de Krawiec. Se puede fijar el número de variables mediante un parámetro.
- El control del tamaño del árbol se realiza mediante una ecuación que asigna probabilidades de crecimiento a cada una de las hojas en función de la profundidad que tiene dentro del árbol. Premia con probabilidades de selección mayores a las hojas con menor profundidad lo que produce en media una limitación de la profundidad de los árboles.

2.3.2. Evolución gramatical

Se habla por primera vez de evolución gramatical en el texto de Ryan, Collins y O'Neill [8] en 1998. Conceptualmente este método es heredero de la Programación Genética de Koza [40], pero la dota de una mayor versatilidad y sencillez de implementación, prescindiendo de la estructura de árboles que habitualmente genera complejidad.

El sistema de Ryan, Collins y O'Neil se basa en la creación de una "gramática" que deberá ser respetada por todos los individuos durante el proceso evolutivo. Las principales mejoras que aporta este método frente a la programación genética son:

- **Resolver problema de clausura:** La clausura implica que una función u operador debería ser capaz de aceptar como entrada cualquier salida producida por cualquier función u operador del conjunto de no terminales. Esta propiedad, en la práctica puede ser difícil de satisfacer [36].
- **Sesgar el proceso evolutivo:** las gramáticas posibilitan la imposición a los individuos de determinadas características de obligado cumplimiento.
- **Mejora de la eficiencia:** al reducir el espacio de búsqueda, el rango de búsqueda está más acotado y es menos costos el rastreo de soluciones.

El uso de este tipo de técnicas para la resolución del problema de creación de nuevas variables es muy limitado. Solo se han encontrado referencias en los trabajos de Gavrilis y otros, en concreto en su propuesta general "Selecting and constructing feature using grammatical evolution" del 2008 [11], así como en la aplicación práctica de la propuesta en la detección problemas cardiacos en el feto [13] y [12].

```
S::=<expr> (0)
<expr> ::= ( <expr> <op> <expr> ) (0)
            | <func> ( <expr> ) (1)
            |<terminal> (2)
 <op> ::= + (0)
            | - (1)
            | * (2)
            | / (3)
<func> ::= sin (0)
            |cos (1)
            |exp (2)
            |log (3)
<terminal>::=<xlist> (0)
            |<digitlist>.<digitlist> (1)
<xlist>::=x1 (0)
            | x2 (1)
            .....
            |xN (N-1)
<digitlist>::=<digit> (0)
            | <digit><digitlist> (1)
<digit> ::= 0 (0) | 1 (1) | 2 (2) .... | 9 (9)
```

Figura 2.3: Ejemplo evolución gramatical

En la figura 2.3 mostramos la gramática diseñada por Gavrilis, Tsolulos y Dermatas [11] para la creación de nuevas variables. Su propuesta consiste en un nuevo método de selección y construcción

de variables, con el objetivo de mejorar el acierto del clasificador, empleando Gramáticas Evolutivas. Creando una superposición no lineal sobre las variables originales. Las principales ventajas del esquema propuesto, según los autores son:

- Se emplean gramáticas para representar el espacio de búsqueda, lo que permite su manipulación de forma sencilla.
- El método escala con facilidad, pudiendo incluir funciones complejas.
- Los resultados son comprensibles fácilmente, gracias a la legibilidad de la gramática.

Capítulo 3

Diseño experimental

El objetivo de esta sección es realizar una presentación de la metodología de creación de nuevas variables, partiendo del conjunto inicial de datos, a través del uso de evolución gramatical. Así como del diseño experimental planteado que permita validar los resultados.

Como se ha visto en los capítulos anteriores hay cierto número de trabajos sobre la creación de variables y su importancia a la hora de mejorar el rendimiento de los modelos de aprendizaje supervisado. Más reducidos son los que se centran en metodologías basadas en programación genética, y aun menos los que emplean una aproximación a la creación de variables a través de la evolución gramatical. Como excepción podemos citar los estudios de Gavrilis [11], [13], [12], que abordan parte de la metodología que se desarrollará en este trabajo.

Hay que destacar que en el principal artículo de Gavrilis, Tsoulos y Dermatas de 2008 [11], se realiza una aproximación combinada a la creación de variables, dado que también implica selección. La metodología que proponen los autores combina la creación de variables mediante evolución gramatical y la selección de las tres más prometedoras, que serán las empleadas en los modelos de clasificación y regresión.

La metodología propuesta se centrará en ampliar el espacio de variables, pasando del espacio inicial F_0 a uno ampliado F_A que contará con las variables iniciales y las derivadas F_D más prometedoras, dejando la función de selección en manos del clasificador seleccionado a priori.

$$F_0 = \{X_1, X_2, ..., X_n\}$$

$$F_D = \{X'_1, X'_1, ..., X'_{n'}\}$$

$$F_A = F_0 \cup F_D$$

En una fase inicial del proyecto se creó un simulador básico con el software R [50], a través del editor RStudio [51], basándonos en la implementación de evolución gramatical que ofrece la librería "gramEvol" [34]. El resultado no fue el deseado puesto que el algoritmo genético, que evolucionaba la población, no convergía de forma adecuada. Por ello nos hemos visto en la necesidad de implementar nuestro propio algoritmo evolutivo, en concreto un algoritmo genético, en el software R [50]. Destacar que hemos empleado las funciones de creación de la gramática y de traducción de la misma implementadas en la librería "gramEvol", dado que el rendimiento que presentan es bueno y ha facilitado la labor de desarrollo. Por traducción nos referimos al proceso que permite pasar de un individuo codificado con una cadena numérica a un individuo representado por n' expresiones matemáticas evaluable, que darán lugar a las n' nuevas variables.

3.1. Arquitectura del experimento

La arquitectura general de la metodología propuesta, que se aplicará a los diferentes experimentos, está reflejada en la figura 3.1.

Destacamos que se ha empleado como función de evaluación la medida F1-Score, por sus buenas prestaciones en conjuntos balanceados y no balanceados. En el Apéndice E hacemos una descripción detallada de la métrica y sus propiedades.

El clasificador que se va a emplear deberá ser seleccionado a priori por el usuario. En nuestro caso se ha usado el mismo clasificador en las tres partes del proceso en las que se requiere su uso, en el bucle de evolución de la población, en la evaluación del conjunto de variables ampliado F_A y en la evaluación del conjunto de variables inicial F_0.

En las ejecuciones iniciales del estudio nos hemos encontrado problemas con el sobreajuste del modelo. Este hecho nos ha llevado a introducir una muestra de validación que permita al sistema independizar la fase de evolución de la población de la fase de evaluación final del conjunto ampliado F_A, formado por el conjunto inicial de variables F_0 y las n' nuevas variables resultantes del proceso evolutivo.

Los pasos que sigue la metodología propuesta son los siguientes:

1. Partiendo del conjunto de datos se construyen las muestras de entrenamiento, validación y test. La división en estas muestras se realiza seleccionando un 80 % de los registros para entrenamiento y un 20 % para test. Del conjunto de entrenamiento se reserva un 30 % de los

registros para la muestra de validación. La selección se hace mediante un muestreo aleatorio simple.

2. Se inicializa el algoritmo evolutivo con un número predefinido de individuos en la población y de variables nuevas que generará cada individuo n'.

3. Se evoluciona la población durante un número prefijado de iteraciones, definido inicialmente en los parámetros de configuración del experimento. Dentro del bucle de evolución de la población se realiza, para cada individuo:

   - Se traducen las n' expresiones del individuo a n' funciones matemáticas mediante la gramática, creando el nuevo espacio ampliado de variables F_A para la muestra de entrenamiento.
   - Se entrena el clasificador seleccionado con el conjunto ampliado de variables F_A sobre la muestra de entrenamiento.
   - Se traducen las n' expresiones del individuo a n' funciones matemáticas, creando el nuevo espacio ampliado de variables F_A para la muestra de validación.
   - Se mide el resultado del clasificador calculando el F1-Score del modelo sobre la muestra de validación.
   - Se construye la nueva población, en función del F1-Score y procesos de selección, cruce y mutación.

4. Al finalizar el número de iteraciones fijado a priori el algoritmo evolutivo devuelve el mejor individuo, compuesto por las n' nuevas variables.

5. Las n' nuevas variables se extrapolan sobre el conjunto de entrenamiento. Por extrapolar nos referimos al proceso de aplicar las nuevas variables sobre un conjunto de datos obteniendo los correspondientes valores numéricos que formarán las nuevas variables.

6. Se entrena el clasificador seleccionado a priori sobre los datos de entrenamiento ampliados F_A.

7. Las n' nuevas variables se extrapolan sobre el conjunto de test.

8. Medimos el resultado obtenido por el clasificador en la muestra de test, mediante el F1-Score, sobre el conjunto de variables ampliado F_A.

9. Se entrena el clasificador sobre la muestra de entrenamiento para el conjunto inicial de variables F_0 y se mide el F1-Score obtenido sobre la muestra de test.

10. Se comparan los resultados obtenidos, F1-Score, sobre la muestra de test para los modelos entrenados sobre el conjunto ampliado de variables F_A y el conjunto inicial de variables F_0.

11. Se repite todo el proceso tantas veces como repeticiones se hayan fijado. El número de repeticiones es importante puesto que afecta a la fiabilidad del test estadístico de comparación de medias, ver Apéndice D.

3.2. Configuración de los experimentos

Dentro de cada uno de los experimentos se definirán dos bloques básicos de parámetros. El primero de ellos hará referencia a la configuración de la gramática y sus operadores, el segundo hará referencia a los parámetros de configuración del algoritmo genético, a saber:

- **Variables creadas:** Número de nuevas variables que se van a crear por individuo.
- **Repeticiones:** Número de ejecuciones independientes que se aplica la metodología sobre cada conjunto de datos, realizando en cada ocasión un muestreo aleatorio simple. Esto nos permitirá disponer de ejecuciones independientes sobre las que aplicaremos un test estadístico para la igualdad de medias, determinando si los valores obtenidos son o no significativos, ver Apéndice D.
- **Iteraciones:** Número de veces que se evoluciona la población mediante el algoritmo genético. Es un valor relevante, dado que un número de iteraciones bajo puede producir una no convergencia del algoritmo y un número muy elevado implica un coste computacional inasumible.
- **Tamaño población:** nº de individuos que tiene la población a evolucionar.
- **Longitud gen:** nº de posiciones que toman valor entre 1 y 255 dentro de cada individuo.
- **Wrapping:** parámetro que indica el número de veces que se puede leer el gen a la hora de decodificar la gramática, en caso de que no se pueda finalizar el proceso de construcción de la variable en una primera lectura.
- **Tipo selección:** metodología empleada para seleccionar los mejores padres para la reproducción y los mejores individuos que pasan a la siguiente generación.
- ***k*1:** número de individuos que participan en la selección por torneo de la ascendencia.
- ***k*2:** número de individuos que participan en la selección por torneo de la nueva población.

- **Elitismo:** número de los mejores individuos que pasan directamente a la próxima generación.
- **Selección de supervivientes:** conjunto de la población sobre el que se realizará la selección de los individuos que pasan a la siguiente generación.
- **Probabilidad de cruce:** probabilidad de que dos padres seleccionados se crucen generando un hijo.
- **Probabilidad mutación:** probabilidad de que un individuo mute.

3.3. Conjuntos de datos

En la tabla 3.1 se muestra un resumen de las principales características de los conjuntos empleados, todos ellos obtenidos en el repositorio UCI [1], en la sección de conjuntos de datos para problemas de clasificación.

Conjuntos datos	Número registros	Número atributos	Número clases	Muestreado	Registros empleados
Glass	214	10	6	No	214
Ion	351	35	2	No	351
Wdbc	569	31	2	No	569
ILPD	583	11	2	No	583
Pima	768	9	2	No	768
Forest type	523	28	4	No	523
Qsar	1.055	42	2	No	1.055
Ecoli	336	8	8	No	336
Heart	270	14	2	No	270
Retinopathy	1.151	20	2	No	1.151
Statlog	2.310	20	7	Si	700
Phishing	11.055	31	2	Si	1.200
Spam	4.601	58	2	Si	1.200
Eeg eye	14.980	15	2	Si	800
Drive diag	58.509	49	11	Si	800

Tabla 3.1: Ficheros

En los experimentos, debido al elevado tiempo de ejecución, se realizarán aproximaciones a conjuntos reducidos de ficheros con el objetivo de disponer de una primera aproximación sobre el funcionamiento general. La selección de estos conjuntos se basa en su bajo número de registros y variables, lo que acelera las pruebas preliminares y en los resultados obtenidos en los experimentos previos. En caso de obtener buenos resultados con estos ficheros abordaremos la prueba completa

sobre el resto de ficheros, en caso de obtener resultados negativos concluiremos que el ajuste o modificación realizada sobre el proceso no ha dado resultado.

Indicamos que como criterio de selección de los ficheros para realizar los experimentos se han tenido en cuenta:

- **Tipo fichero:** dado que nos queremos centrar en problemas de clasificación, solo hemos seleccionado conjuntos de datos de este tipo.

- **Tamaño:** Los tiempos de ejecución de los experimentos son muy elevados dado que requieren múltiples ejecuciones e iteraciones para su validación estadística. El tamaño del conjunto de datos es muy importante dado que estamos empleando el clasificador para generar la función de evaluación (F1-Score). Por tanto en cada iteración del algoritmo se tendrá que ejecutar el clasificador. Esto hace que conjuntos de datos de gran tamaño ralenticen mucho el proceso. Por tanto hemos tomado, como criterio aproximado, que los conjuntos de datos no tengan más de 1.000 registros. En los casos en que el tamaño del conjunto de datos supera esa cantidad con creces se ha procedido a muestrear de forma aleatoria, obteniendo un conjunto de datos de menor tamaño que permite reducir los tiempos de ejecución. En la tabla 3.1 se pueden ver los conjuntos que han sido muestreado y con que número de observaciones. El método de muestreo ha sido un aleatorio simple, teniendo como criterio que todas las clases estuvieran suficientemente representadas como para hacer viable una división en tres conjuntos de datos, entrenamiento, validación y test, teniendo validez estadística. Para que las diferentes ejecuciones se realicen sobre las misma muestra siempre se ha introducido una semilla aleatoria antes del muestreo.

- **Balanceo:** Derivado de los dos puntos anteriores, se han intentado usar conjuntos de datos balanceados o al menos que no presentarán clases muy des-balanceadas. Al disponer de pocos registros nos podríamos encontrar con clases, en alguno de los tres subconjuntos de datos generados (entrenamiento, validación y test), que no tuviera suficientes observaciones de alguna de ellas como para ser representativas a nivel estadístico. Como regla general hemos intentado que cada clase tuviera al menos 150 observaciones.

- **Número de variables:** Al igual que el número de registros penaliza el tiempo de ejecución, el número de variables también lo hace, y por tanto se ha optado por conjuntos con el menor número de variables posible.

3.4. Diseño experimental y evaluación

El principal problema que ha surgido en la realización de los experimentos ha sido el coste computacional del proceso, con unos tiempos de ejecución muy elevados.

Para reducir en la medida de los posible los tiempo de cómputo, como hemos indicado en el punto anterior dedicado a los conjuntos de datos seleccionados, se ha optado por conjuntos lo más reducidos posible y en algunos casos por muestrearlos. Para reducir el número de ejecuciones se ha optado por realizar n ejecuciones independientes, cada una de ellas con todos los pasos del algoritmo y por tanto con una selección muestral diferente.

Hemos comprobado en la fase previa de diseño experimental que la selección de muestra es determinante a la hora de entrenar el sistema. Por tanto para realizar un experimento lo más riguroso posible se han planificado n ejecuciones con cada conjunto de datos, en cada uno de los experimentos se indicara dicho número, que será suficiente para garantizar la validez estadística de los resultados.

En el anexo hay un punto dedicado al contraste de medias pareadas que se empleará para medir la mejora o no obtenida entrenando el clasificador sobre el conjunto de variables ampliado F_A frente a entrenar el mismo clasificador con el conjunto de variables iniciales F_0 D. Como ya hemos expuesto la métrica de rendimiento será el F1-Score E,por tanto analizaremos los valores obtenidos sobre el conjunto de test con F_0 y los obtenidos sobre F_A.

3.5. Computación

La capacidad de computación ha resultado un punto decisivo a la hora de abordar el proyecto de investigación, dado que los tiempos de procesado son extremadamente elevados. Hay que tener en cuenta que los diferentes experimentos se analizan en 15 conjuntos de datos, como hemos visto en 3.3. Además cada conjunto se muestrea y evalúa un número n de veces para garantizar la validez estadística de los datos y, dentro de cada una de esas evaluaciones se realizan m ejecuciones del algoritmo evolutivo. Además hay que tener en cuenta que el uso de un clasificador como generador de la función de evaluación hace que las evaluaciones de la población sean lentas, lo que penaliza de forma considerable el tiempo global de cómputo.

Por centrar el problema de computación diremos que para la ejecución de los 15 ficheros 48 veces cada uno y con 500 iteraciones del algoritmo genético, usando la implementación de “rpart” [49] de

R [50] y el F1-Score resultante como función de evaluación, hemos tardado en torno a una semana. Esto ha obligado a emplear dos máquinas de forma simultanea para reducir los tiempos.

Indicamos, así mismo, que los algoritmos programados están totalmente paralelizados y permiten la ejecución de tantos hilos como cores tenga el procesador. Esta programación ha supeditado la elección del número de muestras aleatorias extraídas de los conjuntos. El objetivo ha sido que fueran múltiplo de los cores disponibles para reducir el tiempo de ejecución y evitar tener las máquinas trabajando por debajo del 100 %.

Para el desarrollo del presente estudio se han empleado dos infraestructuras:

- HP Z600 worksation: Este ordenador de sobremesa dispone de un procesador X5650 a 2.67GHz y 24 GB de RAM, con un Windows 10 como sistema operativo. Tiene dos placas en paralelo con 6 cores cada una, lo que nos brinda 12 cores físicos de procesado, que con hyper threading se transforman en 24 cores lógicos.
- Google Instance VM: Se han desplegado tres máquinas virtuales en Google Cloud con 16, 16 y 24 cores de procesamiento y 16Gb, 16Gb y 24Gb de RAM, con sistema operativo Debian GNU/Linux 8.7 (jessie).

3.6. Pruebas iniciales

Se han realizado una infinidad de pruebas iniciales sobre el problema planteado, intentado diferentes aproximaciones. Destaca como uno de los objetivos iniciales, usar una función de evaluación que fuera independiente del clasificador empleado para medir el rendimiento del modelo sobre el conjunto de variables ampliado F_A. Para ello se han realizado aproximaciones a través del Gain Ratio [41] y del índice de Gini [28], que como se ha visto en la sección 2.1 son dos métricas empleadas de forma habitual para seleccionar variables de interés. Si bien es cierto que estas dos medidas son empleadas por algoritmos de aprendizaje basados en árboles de decisión, es importante tener en cuenta que permiten la evaluación de la calidad de la variable de forma independiente del clasificador, y con unos tiempos de ejecución mucho más reducidos.

Los experimentos realizados en este sentido no han sido satisfactorios y han obligado a un cambio de planteamiento, pasando a usar el clasificador como generador de la función función de evaluación del algoritmo evolutivo.

Hay que destacar también, por el gran impacto que ha tenido en el tiempo de desarrollo, que el uso de la librería de R [50] "gramEvol" [34] no ha dado los resultados esperados. Los algoritmos

evolutivos que emplea (algoritmo genético o estrategia evolutiva) no han convergido con el paso de las iteraciones, comportándose de forma irregular. Por ello ha sido necesario implementar en R [50] un algoritmo genético que permitiera evolucionar la población. Destacamos que hemos mantenido las funciones de creación de la gramática y de traducción de los individuos a funciones matemáticas proporcionadas por el paquete "gramEvol". [34].

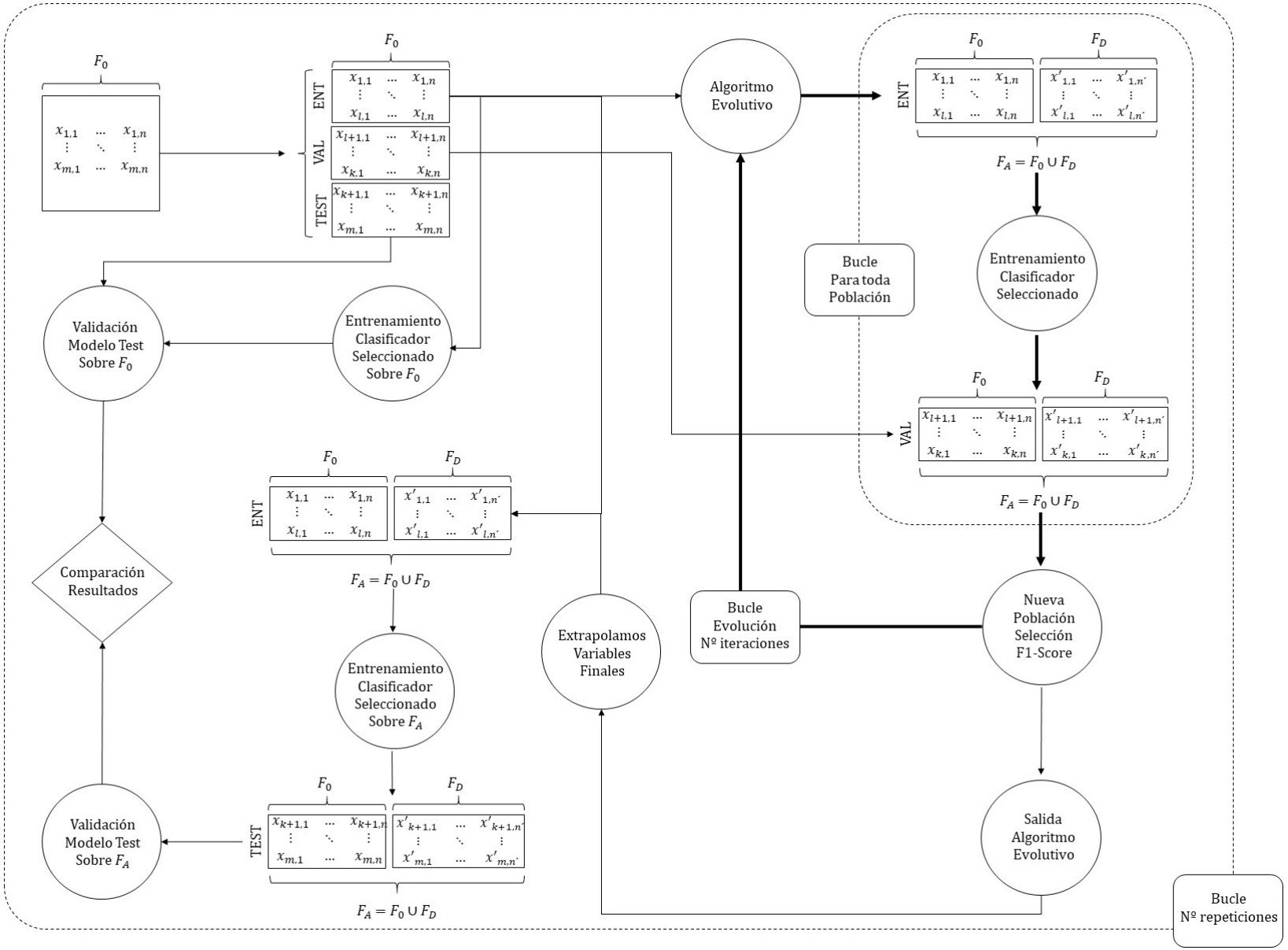

Figura 3.1: Esquema arquitectura experimentos

Capítulo 4

Resultados experimentales y discusión

En esta capítulo vamos experimentar con el planteamiento expuesto en la sección 3.1, resumido en la figura 3.1, empleando los clasificadores: árbol de decisión tipo CART [28], k-vecinos [24] y regresión logística [24].

El capítulo se ha dividido en secciones que se corresponden con experimentos, en cada uno de ellos se probarán diferentes configuraciones o metodologías para intentar avanzar hacia una solución satisfactoria del problema. Dentro de cada una de ellas se expondrá el planteamiento concreto que se ha empleado así como los resultados obtenidos, con las correspondientes conclusiones. En algunas de las secciones se realizará comparativas con secciones previas, con el objetivo de extraer conclusiones sobre qué enfoque o configuración ofrece mejores resultados.

En cuanto a los clasificadores empleados, paquetes empleados de R [50] y configuraciones de los mismos se pueden consultar los apéndices. En concreto:

- **Árbol de decisión:** consultar Apéndice A.
- **Regresión logística:** consultar Apéndice B.
- ***k*-vecinos:** consultar Apéndice C.

Hay que destacar que el árbol de decisión es el clasificador que, a priori, mejor se ajusta al planteamiento realizado, dado que presenta una gran capacidad para trabajar con variables correlacionadas, variables con valores perdidos, valores infinitos... Estas características hacen que la

generación de nuevas variables pueda ser menos restrictiva que con otros tipos de algoritmos de clasificación. Por ello el estudio principal se centra en la capacidad, de la metodología propuesta, para trabajar con árboles de decisión tipo CART, intentando extender los resultados que se obtengan a los clasificadores regresión logística y k-vecinos.

En los próximos experimentos probaremos:

- Distintas configuración del algoritmos evolutivo y de los clasificadores.
- Diferentes métodos de cruce y mutación.
- Diferentes tipos de operadores matemáticos en la gramática.
- Penalización de la función de evaluación en función de la correlación de las nuevas variables generadas con las iniciales.
- Análisis de la convergencia del algoritmo evolutivo en función de las iteraciones.
- Comportamiento de la metodología propuesta con diferentes clasificadores.

4.1. Experimento 1

Este experimento es el primero que hemos realizado con el planteamiento final del estudio, expuesto en el apartado 3.1. Se han realizado una serie de pruebas previas para comprobar la utilidad de diferentes planteamientos, la programación realizada y el rendimiento del algoritmo evolutivo, las conclusiones obtenidas se han tomado como punto de partida para los experimentos expuestos a continuación.

Sin pretensión de exhaustividad el objetivo de este experimento es realizar un acercamiento inicial, que nos ofrezca unos primeros resultados que permitan validar la arquitectura propuesta.

4.1.1. Planteamiento

Se ha definido la gramática reflejada en la tabla 4.1, donde las funciones f_log y $func\_sigm$ corresponden a:

$$f\_log(x) = \begin{cases} log(x) & \forall x > 0 \\ 0 & \forall x \leq 0 \end{cases}$$

$$func\_sig(x) = \frac{1}{1 + e^{-x}}$$

Gramática
$< expr > ::= < expr >< op >< expr > \mid < func > (< expr >) \mid < var >$
$< func > ::= sin \mid cos \mid f\_log \mid func\_sigm$
$< op > ::= + \mid - \mid * \mid /$
$< var >$ ::= variables independientes del conjunto de datos

Tabla 4.1: Gramática Experimento 1

Los parámetros de configuración del algoritmo están reflejados en tabla 4.2. En la sección 3.2 se explica el significado de cada uno de ellos.

Como clasificador se ha empleado un árbol de decisión tipo CART, ver Apéndice A.

Se fija el número de variables derivadas que se van a crear en cada ejecución del proceso en cinco. En las primeras pruebas se probó con diez pero con el uso del clasificador como función de evaluación y el consecuente aumento de tiempos de ejecución nos hemos visto obligados a reducir el número. Hay que tener en cuenta que la elección del número de variables derivadas que se van a

Parámetro	Valor
Variables creadas	5
Repeticiones	48
Iteraciones	500
Población	300
Longitud gen	30
Wrapping	1
Tipo selección	Tournament
$k1$	2
$k2$	2
Elitismo	0
Selección de supervivientes	(μ, λ)
Probabilidad cruce	1
Probabilidad mutación	0.5
Clasificador	CART

Tabla 4.2: Parametrización Experimento 1

generar se realiza a priori, sin disponer de datos comparativos que nos puedan servir para evaluar el número óptimo. No obstante es probable que dicho número varíe en función del conjunto de datos.

En este experimento no se ha penalizado la función de evaluación en caso de que alguna de las variables no pueda ser decodificada. Una variable es no decodificable cuando el proceso de "traducción" del vector numérico que la representa, al lenguaje matemático mediante operadores aritméticos, no se puede finalizar. Normalmente esto ocurre por falta de elementos en el gen que representa a la variable, lo que provoca que no se concluya la decodificación. En estos casos la variable se marcará como "NT".

Como estrategia de selección se ha empleado "tournament selection", tanto en la selección de los padres para combinar como en la de los individuos que progresan a la siguiente generación, dado que ha mostrado su fiabilidad frente a otros tipos de selección como "fitness proportional selection" o "ranking selection" [14].

Hemos optado por no emplear elitismo para analizar como progresa el algoritmo sin mantener a los mejores individuos generación tras generación, lo que nos puede dar una perspectiva de como se comporta el resto de la población. Como probabilidad de cruce hemos trabajado con 1, por tanto todos los individuos seleccionados como padres para el cruce producirán un hijo candidato a la siguiente generación.

La selección de supervivientes que pasarán a la siguiente generación se realiza sobre el conjunto (μ, λ) esto es, la unión de los padres y los hijos obtenidos en el proceso de combinación y mutación.

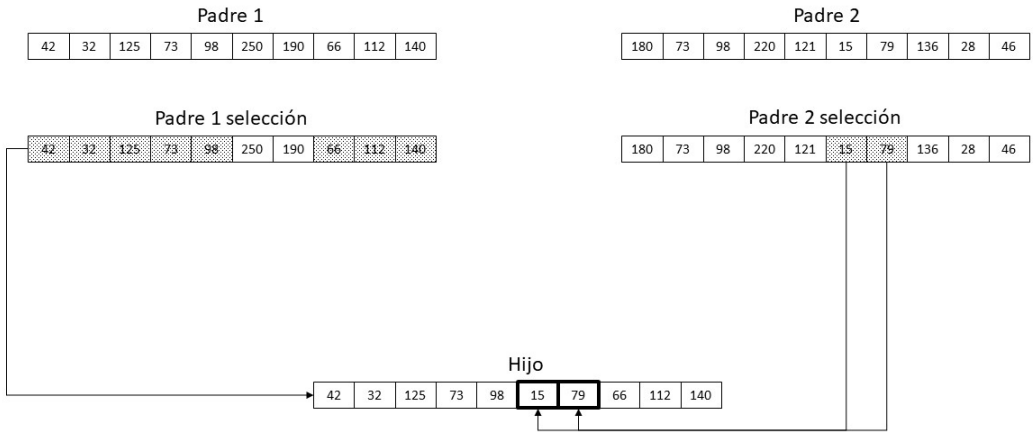

Figura 4.1: Esquema de cruce en el Experimento 1

La estrategia de cruce empleada consiste en seleccionar, para cada uno de los genes del individuo, tantos como variables nuevas vayamos a crear, un par de puntos de corte. Con dichos puntos de corte, sobre los dos individuos que se van a combinar, se realiza el intercambio de información. Por cada ejecución de la combinación se produce un hijo resultante. El esquema se puede ver en la figura 4.1.

En cuanto a la estrategia de mutación, partiendo del parámetro de probabilidad definido a priori que sirve para determinar si se va a producir o no la mutación, obtenemos un vector de probabilidades para cada individuo de la población, con tantos valores como variables tenga. Si la probabilidad obtenida es menor que la prefijada se produce la mutación de una de las posiciones del individuo, de forma aleatoria. En la figura 4.2 se puede ver un esquema del proceso.

4.1.2. Resultados

Los resultados obtenidos en los conjuntos de datos,, ya presentados en la tabla 3.1, se pueden ver en la tabla 4.3. Los criterios de comparación de medias y contraste se han reflejado en el Apéndice D.

La tabla de resultados 4.3 contiene la información:

- **Conjunto de datos:** conjunto de datos sobre el que se realiza el experimento.

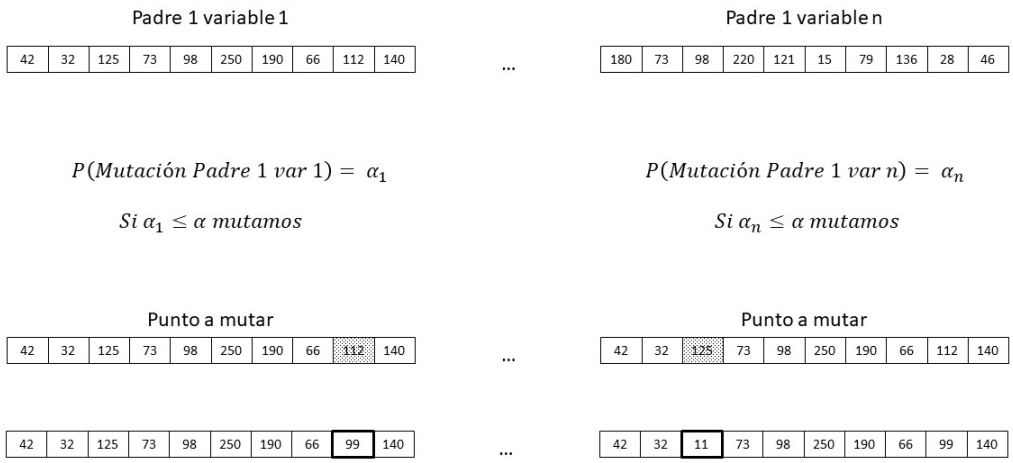

Figura 4.2: Esquema de mutación en el Experimento 1

- ***F*1(F_0):** F1-Score medio resultante de ejecutar el modelo de clasificación sobre el conjunto de variables inicial F_0, sobre cada una de las 48 muestras aleatorias.
- ***F*1(F_A):** F1-Score medio resultante de ejecutar el modelo de clasificación sobre el conjunto de variables ampliado F_A, sobre cada una de las 48 muestras aleatorias.
- **Diferencia:** media de las diferencias obtenidas en cada ejecución independiente sobre el resultado de $F1(F_A) - F1(F_0)$.
- ***p*-valor:** valor de la función de distribución de contraste en función del estadístico, ver Apéndice D.
- **Sobreajuste:** Porcentaje de ejecuciones independientes en las que $F1(F_0) \geq F1(F_A)$. Al emplear el F1-Score, obtenido sobre la muestra de validación como función de evaluación del algoritmo evolutivo, puede ocurrir que tras n iteraciones del algoritmo evolutivo las variables nuevas generadas se hayan ajustado "demasiado" a la muestra de entrenamiento y validación, perdiendo generalidad.
- **Mejora:** indica si con la metodología, bajo el nivel de significatividad $1-\alpha$, la diferencia entre el F1-Score obtenido con el conjunto de datos inicial F_0 y el obtenido con F_A es estadísticamente significativa.

Los resultados muestran que en 11 del 15 conjuntos de datos, con la metodología planteada, se consigue una diferencia significativa en el rendimiento del clasificador, mediada a través del F1-Score. Hay que destacar que aparecen con cierta frecuencia casos en los que se produce sobre-ajuste del clasificador, casos en los que el resultado obtenido con el conjunto de variables ampliado es peor que el obtenido con el conjunto de variables iniciales.

Conjunto de datos	$F1(F_0)$	$F1(F_A)$	Diferencia	p-valor	Sobreajuste	Mejora $\alpha = 0{,}05$
drive_diag	45.02	48.89	3.86	0.000351	29.17	1
ecoli	56.31	58.32	2.01	0.025369	29.17	1
eeg_eye	63.16	68.24	5.07	2.4e-05	14.58	1
forest_type	74.88	76.69	1.81	0.006282	29.17	1
glass	41.14	45.15	4.01	0.001378	25.00	1
heart	75.65	78.47	2.83	0.010727	29.17	1
ilpd	69.78	68.9	-0.88	0.87251	33.33	0
ion	88.11	88.36	0.25	0.385793	43.75	0
phishing	92.06	92.17	0.11	0.338426	31.25	0
pima	74.47	73.61	-0.86	0.948204	58.33	0
qsar	81.52	82.87	1.36	0.004216	33.33	1
retinopathy	62.77	71.21	8.44	0	4.17	1
spam	88.02	88.7	0.67	0.042207	37.5	1
statlog	74.68	78.49	3.8	0.000392	16.67	1
wdbc	91.74	94.34	2.6	3.1e-05	14.58	1
Total	71.95	74.92	2.34			73.33 %

Tabla 4.3: Resultados Experimento 1

Con una fiabilidad del 95 % (α=0.05) podemos ver, en la tabla de resultados 4.3, como la metodología propuesta mejora el resultado obtenido con F_0 en el 73.33 % de los conjuntos de datos analizados.

Para analizar la convergencia del algoritmo se han realizado los gráficos de evolución, que muestran el F1-Score medio obtenido sobre la muestra de validación en las 500 iteraciones. Se presenta un gráfico para cada conjunto de datos con la línea de la media del F1-Score en cada iteración y un intervalo de confianza al 95 %, definido por la zona gris.

Podemos ver como los gráficos de evolución muestran una mejora rápida en las primeras iteraciones, en algunos de los conjuntos de datos parece que la tendencia de crecimiento se proyecta más allá de las 500 iteraciones realizadas. Un ejemplo de este comportamiento lo tenemos en las figuras: 4.7, 4.8, 4.13 y 4.16.

Cabe destacar que los gráficos que corresponden a los conjuntos de datos que no mejoran el rendimiento con la aplicación de la metodología propuesta 4.10, 4.12, 4.11 y 4.15 presentan un

comportamiento similar al resto. Hay que tener en cuenta que los gráficos se han obtenido sobre la muestra de validación, por tanto se produce un claro sobreajuste.

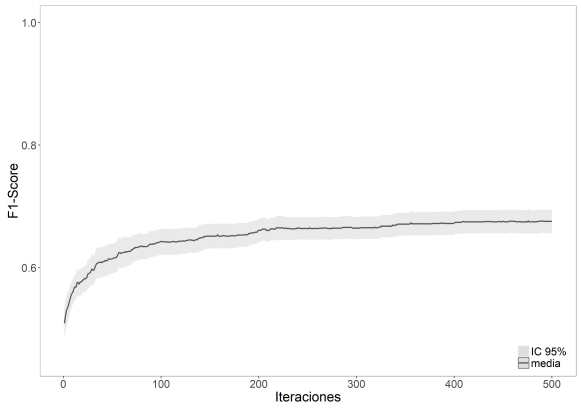

Figura 4.3: Exp1: train drive diag

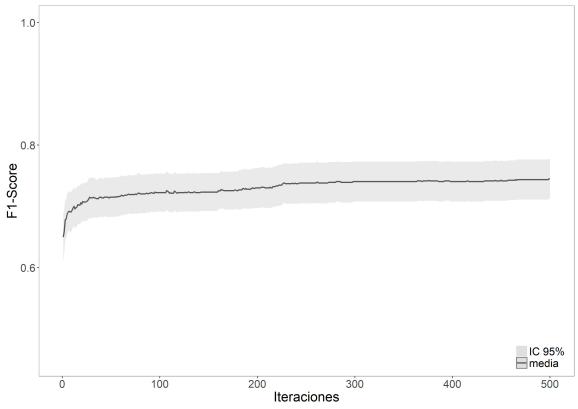

Figura 4.4: Exp1: train ecoli

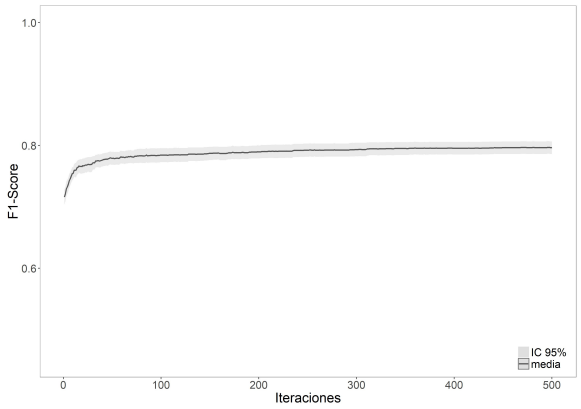

Figura 4.5: Exp1: train eeg eye

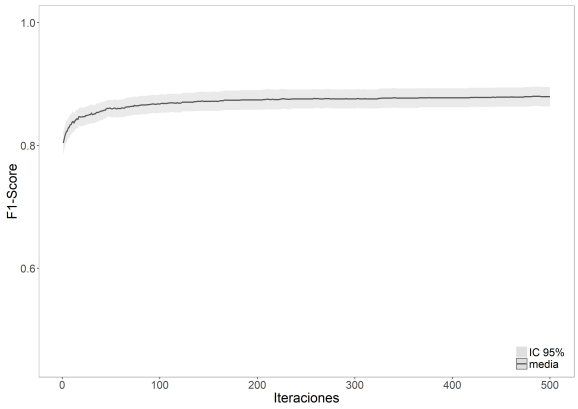

Figura 4.6: Exp1: train forest type

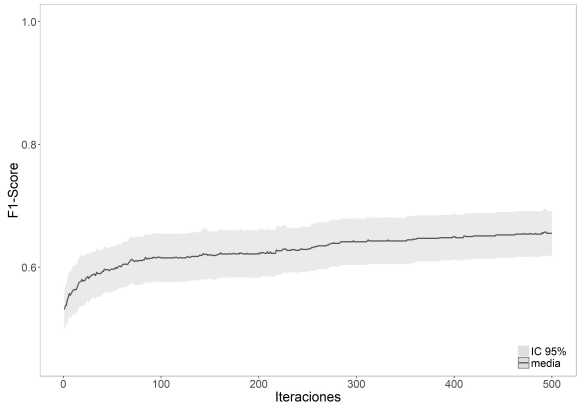

Figura 4.7: Exp1: train glass

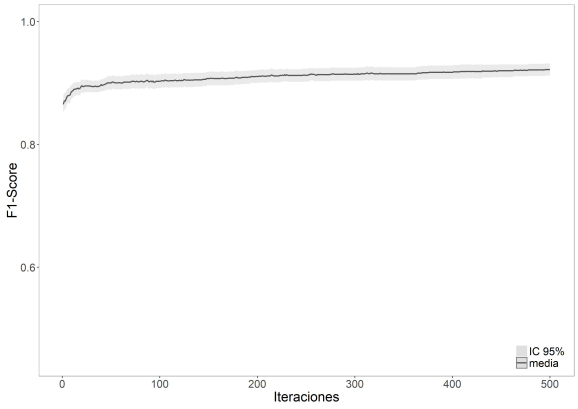

Figura 4.8: Exp1: train heart

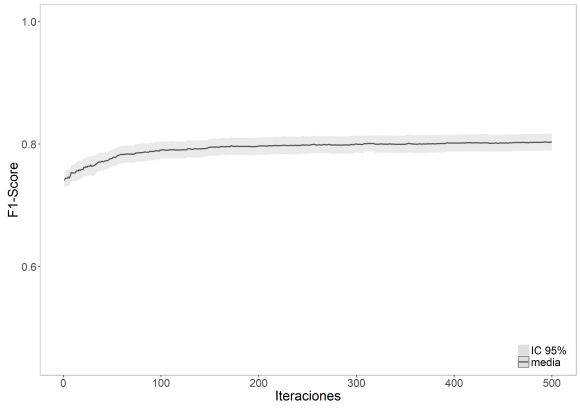

Figura 4.9: Exp1: train ilpd

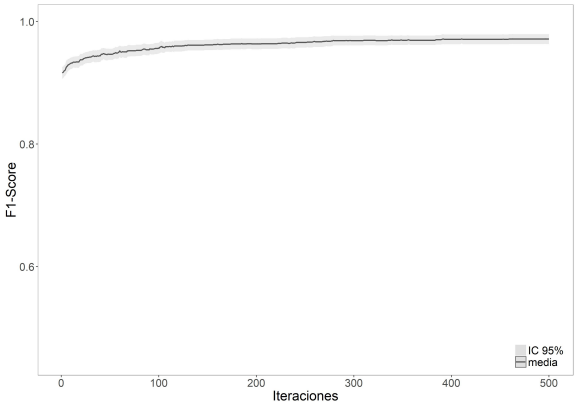

Figura 4.10: Exp1: train ion

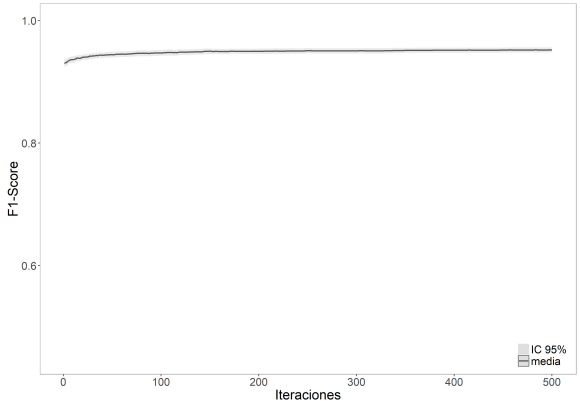

Figura 4.11: Exp1: train phishing

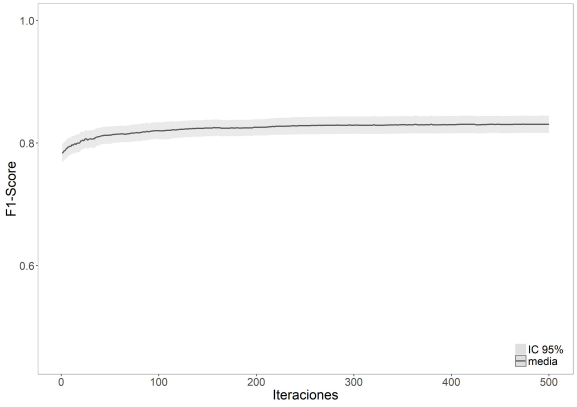

Figura 4.12: Exp1: train pima

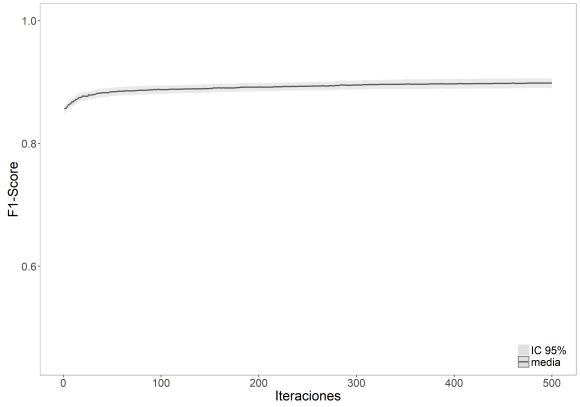

Figura 4.13: Exp1: train qsar

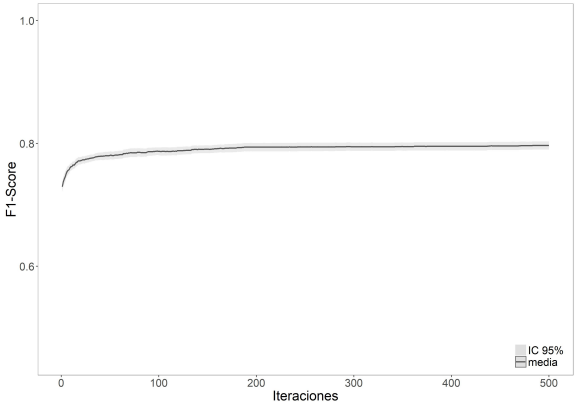

Figura 4.14: Exp1: train retinopathy

En las figuras 4.18 hasta 4.32 podemos ver un ejemplo de las variables creadas en cada conjunto de datos. Para seleccionar las variables mostradas hemos buscado la ejecución con cada conjunto que tuviera mayor diferencia entre el $F1(F_0)$ y el $F1(F_A)$. Como se puede ver, en algunos casos, aparecen variables a "NT", esto indica que la cadena numérica que codificable dicha variable dentro del individuo seleccionado no es decodificable completamente.

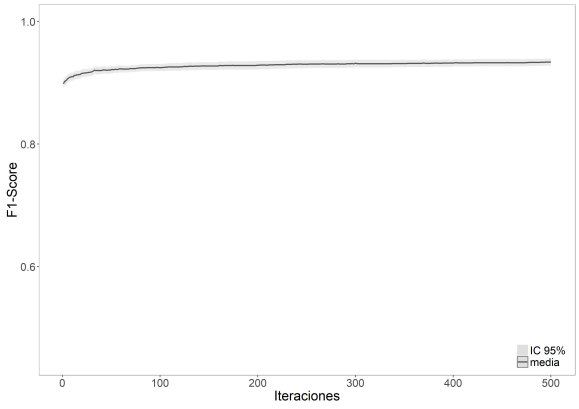

Figura 4.15: Exp1: train spam

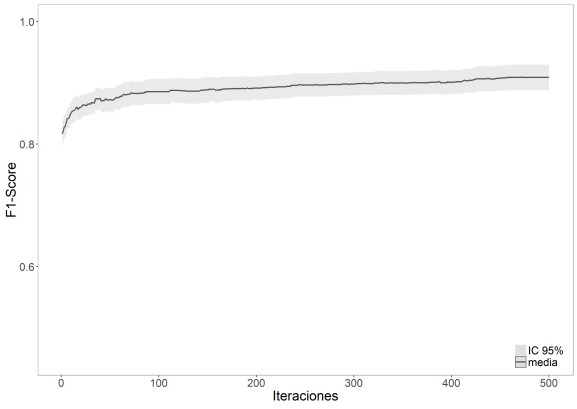

Figura 4.16: Exp1: train statlog

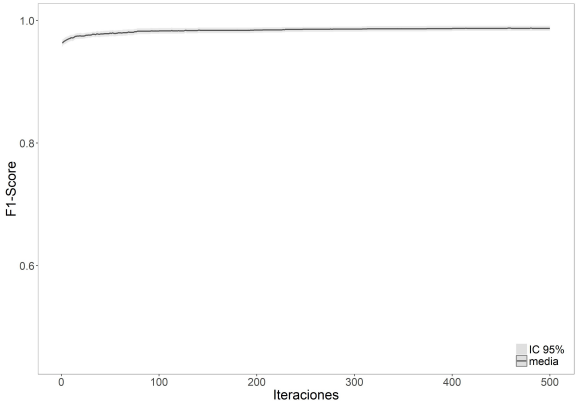

Figura 4.17: Exp1: train wdbc

En general las variables resultantes no muestran una excesiva complejidad, lo que a priori puede redundar en un menor sobre-ajuste, pero también podría indicar que con un mayor número de iteraciones del algoritmo evolutivo se podría llegar a obtener variables derivadas más complejas, que podrían ayudar a ajustar mejor el modelo.

4.1.3. Conclusiones

Más allá de las diferencias estadísticas podemos identificar 10 casos en los que la mejora del clasificador, aplicando la metodología propuesta, está por encima del 1.36 %. Esto supone que en el 66.66 % de los casos sería interesante aplicar esta metodología. En el caso del conjunto de datos "spam" aunque la mejora obtenida es estadísticamente significativa es solo del 0.67 % lo que podría hacer poco interesante aplicar este sistema debido a su coste computacional.

A la vista de los gráficos de evolución parece que alguno de los conjuntos de datos podrían seguir mejorando la función de evaluación con más ejecuciones.

De los gráficos también podemos concluir que el comportamiento, en los conjuntos de datos que presentan mejora con la aplicación de la metodología, es bueno y que por tanto se produce un sobreajuste al extrapolar sobre la muestra de test. Podría ser debido al tamaño reducido de los conjuntos de datos, apareciendo algún problema en el muestreo que haga que las muestras no sean representativas. Si bien el número de ejecuciones aleatorias es suficientemente alto como para corregir ese problema podría ser interesante abordar el problema con más ejecuciones o con una mejora en el muestreo que garantice la representatividad de las muestras de entrenamiento, validación y test.

$$f(x) = \frac{V12}{V8}$$

$$f(x) = \frac{\sin(V14)}{V10}$$

$$f(x) = \sin(f\_log(V18)) \cdot V25$$

$$f(x) = V7 \cdot V9 \cdot V11$$

$$f(x) = NT$$

Figura 4.18: Exp1: ecuaciones drive diag

$$f(x) = V3$$

$$f(x) = \cos(\cos(V7 + V8 + V3))$$

$$f(x) = func\_sigm(\cos(V2 + V3))$$

$$f(x) = \sin(V6 - \frac{func\_sigm(V4)}{V3} \cdot V2)$$

$$f(x) = f\_log(\sin(\sin(\frac{\frac{\cos(V2)}{\sin(V4)}}{V5})))$$

Figura 4.19: Exp1: ecuaciones ecoli

$$f(x) = f\_log(f\_log(P7 - FC5))$$

$$f(x) = O1 - T8$$

$$f(x) = f\_log(\frac{F8}{P7})$$

$$f(x) = \frac{F8 \cdot func\_sigm(F7)}{F4}$$

$$f(x) = \sin(T8 \cdot \cos(\cos(f\_log(P8) - f\_log(FC6)))) + f\_log(f\_log(O2))$$

Figura 4.20: Exp1: ecuaciones eeg eye

$$f(x) = b4 + func\_sigm(\frac{func\_sigm(pred\_minus\_obs\_H\_b2)}{\cos(pred\_minus\_obs\_H\_b1)}) - pred\_minus\_obs\_H\_b1$$

$$f(x) = NT$$

$$f(x) = pred\_minus\_obs\_H\_b2$$

$$f(x) = \frac{pred\_minus\_obs\_H\_b1}{b7}$$

$$f(x) = func\_sigm(f\_log(f\_log(b2)) + func\_sigm(\frac{b6}{b1}))$$

Figura 4.21: Exp1: ecuaciones forest type

$$f(x) = func\_sigm(V7 + V4)$$

$$f(x) = func\_sigm(\sin(V7))$$

$$f(x) = V4 - V8$$

$$f(x) = \frac{f\_log(\frac{V5}{V6} \cdot \cos(V7))}{V5}$$

$$f(x) = NT$$

Figura 4.22: Exp1: ecuaciones glass

$$f(x) = V13$$

$$f(x) = V3 + \frac{\cos(V5)}{\cos(V12)}$$

$$f(x) = V2 - f\_log(\cos(V12) + f\_log(V3))$$

$$f(x) = func\_sigm(V10 \cdot \sin(V13))$$

$$f(x) = f\_log(V11) \cdot V12$$

Figura 4.23: Exp1: ecuaciones heart

$$f(x) = f\_log(Sgot\_Aspartate) + DB\_Direct\_Bilirubin$$

$$f(x) = NT$$

$$f(x) = \frac{ALB\_Albumin.}{\sin(Sgpt\_Alamine - \frac{AG\_Ratio}{Age})}$$

$$f(x) = \frac{Sgot\_Aspartate}{TB\_Total\_Bilirubin}$$

$$f(x) = Age - TP\_Total\_Protiens$$

Figura 4.24: Exp1: ecuaciones ilpd

$$f(x) = V26 - \frac{V3}{func\_sigm(V2)}$$

$$f(x) = NT$$

$$f(x) = \cos(V4) + V31$$

$$f(x) = f\_log(V31 \cdot \cos(V6) \cdot V23 + V16 \cdot \cos(f\_log(V24)))$$

$$f(x) = V22$$

Figura 4.25: Exp1: ecuaciones ion

$$f(x) = Google\_Index$$

$$f(x) = HTTPS\_token \cdot URL\_Length$$

$$f(x) = \frac{RightClick}{func\_sigm(web\_traffic)}$$

$$f(x) = \sin(func\_sigm(Domain\_registeration\_length) - \cos(Request\_URL)) + Request\_URL$$

$$f(x) = HTTPS\_token - \frac{\frac{having\_Sub\_Domain}{func\_sigm(Page\_Rank)}}{having\_At\_Symbol}$$

Figura 4.26: Exp1: ecuaciones phishing

$$f(x) = V6$$

$$f(x) = func\_sigm(V2)$$

$$f(x) = f\_log(\sin(V6)) + V2$$

$$f(x) = NT$$

$$f(x) = \cos(\cos(f\_log(V1))) - V7$$

Figura 4.27: Exp1: ecuaciones pima

$$f(x) = \cos(V30)$$

$$f(x) = V13 - V5 - V4$$

$$f(x) = NT$$

$$f(x) = NT$$

$$f(x) = \frac{\cos(f\_log(V8) - f\_log(V18))}{f\_log(f\_log(V22))}$$

Figura 4.28: Exp1: ecuaciones qsar

$$f(x) = \sin(V15 - \sin(V12))$$

$$f(x) = V9 \cdot V1$$

$$f(x) = V18 + \sin(V12)$$

$$f(x) = V4 + V10 \cdot V3$$

$$f(x) = V4 - V2$$

Figura 4.29: Exp1: ecuaciones retinopathy

$$f(x) = NT$$

$$f(x) = \sin(V45) - \sin(\sin(V7))$$

$$f(x) = V45$$

$$f(x) = \text{func\_sigm}(V53 + V24)$$

$$f(x) = V15$$

Figura 4.30: Exp1: ecuaciones spam

$$f(x) = NT$$

$$f(x) = \cos(\sin(\sin(V18))) + \cos(V19)$$

$$f(x) = V1$$

$$f(x) = NT$$

$$f(x) = \cos(\cos(\text{func\_sigm}(\sin(V8) \cdot \sin(\sin(V14)))))$$

Figura 4.31: Exp1: ecuaciones statlog

$$f(x) = \frac{V25}{\text{f\_log}(V10)} \cdot V24$$

$$f(x) = V23$$

$$f(x) = NT$$

$$f(x) = NT$$

$$f(x) = V26 \cdot V32$$

Figura 4.32: Exp1: ecuaciones wdbc

4.2. Experimento 2

El planteamiento que hemos seguido en el desarrollo del estudio es ir añadiendo complejidad a los experimentos según avanzamos, para poder validar si las modificaciones incorporadas mejoran los resultados o no, y en su caso si las posibles penalizaciones en tiempo de ejecución o en complejidad es justifican con los resultados obtenidos.

4.2.1. Planteamiento

Este segundo experimento, mantiene la metodología general expuesta en el apartado 3.1. El objetivo ha sido probar nuevas estrategias de cruce y mutación, aplicando técnicas menos destructivas que las empleadas en el Experimento 1. También hemos incluido elitismo en el algoritmo evolutivo.

En cuanto a la estrategia de cruce pasamos a trabajar con variables completas dentro de cada individuo. Como primer paso seleccionamos un punto de corte, de forma aleatoria, que corresponde al inicio de una de las variables, si el individuo tiene n variables existen n posibles puntos de cruce. En el segundo padre seleccionamos el mismo punto de cruce que en el primero y desde dicho punto hasta el final del individuo realizamos el cruce de material genético. En la figura 4.33 se puede ver la metodología propuesta.

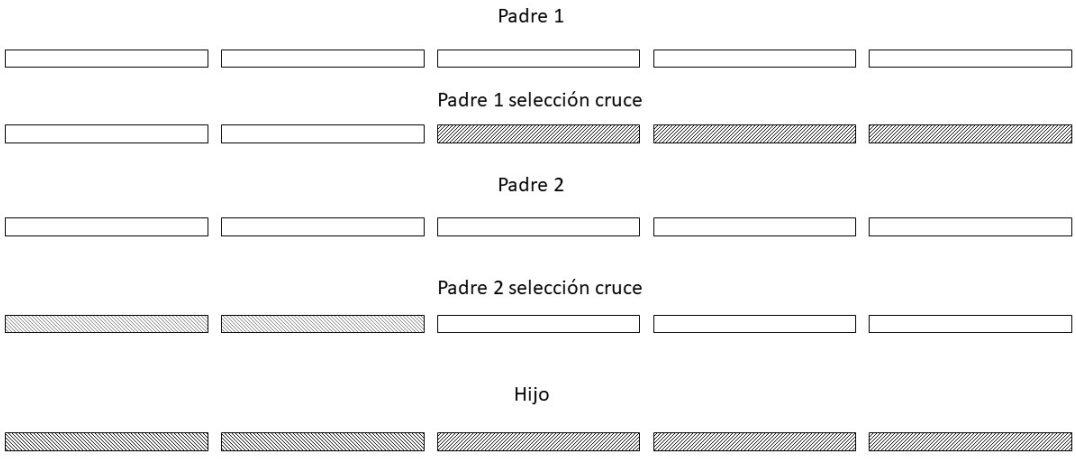

Figura 4.33: Esquema de cruce en el Experimento 2

Este sistema de cruce garantiza que el nuevo individuo es la unión de variables del primero y del segundo individuo, sin destruir material genético y posibles soluciones ventajosas. Dado que la unión de los individuos sigue manteniendo las mismas soluciones pero ahora distribuidas de otra forma entre los dos nuevos individuos.

Parámetro	Valor
Variables creadas	5
Repeticiones	48
Iteraciones	500
Población	300
Longitud gen	30
Wrapping	1
Tipo selección	Tournament
$k1$	2
$k2$	2
Elitismo	5
Selección de supervivientes	(μ, λ)
Probabilidad cruce	0.8
Probabilidad mutación	0.05
Clasificador	CART

Tabla 4.4: Parametrización Experimento 2

También se ha modificado la estrategia de mutación. Ahora la probabilidad fijada en la configuración sirve de umbral a la hora de decidir si se muta cada uno de los elementos que componen un gen (variable nueva generada), dentro de cada individuo tendremos tantos genes como variables nuevas estemos creando. Generamos un vector aleatorio de la longitud del gen, en aquellas posiciones en las que el valor aleatorio obtenido es inferior al umbral fijado procedemos a realizar una mutación aleatoria.

Hemos empleado la misma gramática que en el primer experimento, reflejada en la tabla 4.1. La configuración de parámetros se encuentra en la tabla 4.4. Como nuevo parámetro se ha introducido elitismo de cinco individuos, esto se traduce en que en cada iteración los cinco individuos con mejor función de evaluación en la población inicial se mantienen en la nueva población generada. Lo que en la práctica sirve para mantener las mejores soluciones en la diferentes iteraciones. La diferencia de comportamiento se puede ver en los gráficos de evolución comparativos del Experimento 1 vs Experimento 2.

4.2.2. Resultados

En la tabla 4.5 mostramos el resultado de ejecutar la configuración propuesta sobre los 15 conjuntos de datos. El significado de las columnas es el mismo que el expuesto en el Experimento 1, tabla 4.3.

Del total de conjuntos en diez de los casos conseguimos mejorar el rendimiento del clasificador aplicando la metodología propuesta, esto representa un 66.66 %. En los casos en que el test de medias propuesto, ver Apéndice D, indica que los valores son significativos encontramos diferencias en el rendimiento desde 1.9, del conjunto "wdbc", hasta el 8.25 del conjunto "retinophaty".

4.2.3. Comparativa Experimento 1 vs Experimento 2

En la tabla 4.6 mostramos la comparación de resultados entre el Experimento 1 y Experimento 2. Hay que indicar que con este nuevo planteamiento en el conjunto "spam" ya no se obtienen diferencias significativas, si bien las obtenidas en el Experimento 1, como ya indicamos eran muy bajas y con un p-valor asociado elevado. En este nuevo experimento la diferencia en $F1$ pasa a ser de 0.46, ligeramente inferior que en el Experimento 1.

Conjunto de datos	$F1(F_0)$	$F1(F_A)$	Diferencia	p-valor	Sobreajuste	Mejora $\alpha = 0{,}05$
drive_diag	46.44	51.8	5.35	5.4e-05	16.67	1
ecoli	57.1	61.37	4.26	0.000551	22.92	1
eeg_eye	64.25	66.39	2.14	0.002343	27.08	1
forest_type	75.56	78.06	2.5	0.000686	27.08	1
glass	38.4	40.61	2.22	0.054569	35.42	1
heart	75.49	77.66	2.17	0.037366	31.25	1
ilpd	69.68	68.56	-1.12	0.964487	37.5	0
ion	89.17	89.66	0.48	0.209617	35.42	0
phishing	91.59	91.48	-0.12	0.697989	45.83	0
pima	74.42	74.45	0.04	0.474245	41.67	0
qsar	81.73	83.74	2.01	0.000138	16.67	1
retinopathy	62.67	70.94	8.27	0	4.17	1
spam	88.14	88.6	0.46	0.09451	41.67	0
statlog	70.33	75.22	4.88	0.000386	20.83	1
wdbc	92.24	94.14	1.9	0.00397	29.17	1
Total						66.66 %

Tabla 4.5: Resultados Experimento 2

La comparación de resultados muestra valores muy similares en los dos Experimentos. Los aciertos con el conjunto inicial $F1(F_0)$ son prácticamente idénticos, como era previsible, esto indica que los muestreos están dando buen resultado y que las muestras de trabajo son representativas. En cuanto a los valores obtenidos son el $F1(F_A)$ los valores también son muy similares. Esto indica que no hay

Conjunto datos	E1 $F1(F_0)$	E2 $F1(F_0)$	E1 $F1(F_A)$	E2 $F1(F_A)$	E1 Diferencia	E2 Diferencia	E1 Sobreajuste	E2 Sobreajuste
drive_diag	46.44	45.02	51.8	48.89	5.35	3.86	16.67	29.17
ecoli	57.1	56.31	61.37	58.32	4.26	2.01	22.92	29.17
eeg_eye	64.25	63.16	66.39	68.24	2.14	5.07	27.08	14.58
forest_type	75.56	74.88	78.06	76.69	2.5	1.81	27.08	29.17
glass	38.4	41.14	40.61	45.15	2.22	4.01	35.42	25
heart	75.49	75.65	77.66	78.47	2.17	2.83	31.25	29.17
ilpd	69.68	69.78	68.56	68.9	-1.12	-0.88	37.5	33.33
ion	89.17	88.11	89.66	88.36	0.48	0.25	35.42	43.75
phishing	91.59	92.06	91.48	92.17	-0.12	0.11	45.83	31.25
pima	74.42	74.47	74.45	73.61	0.04	-0.86	41.67	58.33
qsar	81.73	81.52	83.74	82.87	2.01	1.36	16.67	33.33
retinopathy	62.67	62.77	70.94	71.21	8.27	8.44	4.17	4.17
spam	88.14	88.02	88.6	88.7	0.46	0.67	41.67	37.5
statlog	70.33	74.68	75.22	78.49	4.88	3.8	20.83	16.67
wdbc	92.24	91.74	94.14	94.34	1.9	2.6	29.17	14.58
Total	71.81	71.95	74.18	74.29			28.89	28.61

Tabla 4.6: Comparativa Experimento 1 vs Experimento 2

una mejora clara con la nueva metodología, si bien es ligeramente superior, 74.29 frente a 74.18. Las diferencias son difíciles de comparar dado que en unos conjuntos un experimento es mejor que otro y en otros casos ocurre lo contrario. En cuanto a los casos en los que aparece sobreajuste también son muy similares.

En las figuras 4.34 hasta 4.48 mostramos las curvas de ajuste medias, para las 48 ejecuciones, sobre el conjunto de test. Comparamos las curvas obtenidas en el Experimento 1 vs las obtenidas en el Experimento 2. Es interesante indicar que en todos los casos, menos en el conjunto "glass", el planteamiento del Experimento 2 genera curvas con mejor rendimiento que las obtenidas con el Experimento 1. Esto indica que se ajusta mejor y más rápido a la muestra de test, aunque para las 500 iteraciones que estamos empleando no se traduzca en una mejora real sobre el conjunto de test. No obstante nos hace pensar que es mejor continuar trabajando con la línea del Experimento 2 para futuras evoluciones.

4.2.4. Conclusiones

Los resultados obtenidos son relativamente buenos, ver tabla 4.5, y hacen que la metodología sea interesante de aplicar, dado que en un 66.66 % de los casos obtenemos mejoras significativa a nivel estadístico y con diferencias que justificarían su aplicación. Casi no hay diferencias entre

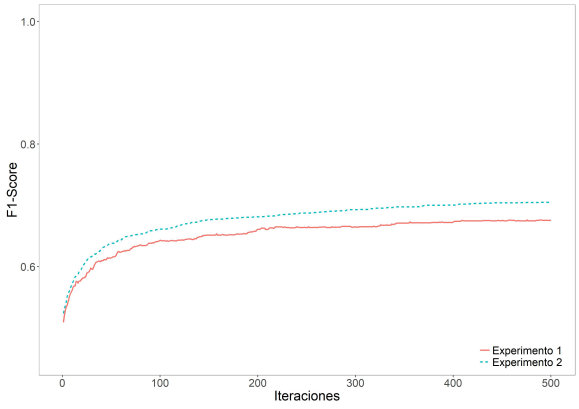

Figura 4.34: Exp1 vs Exp2: train drive diag

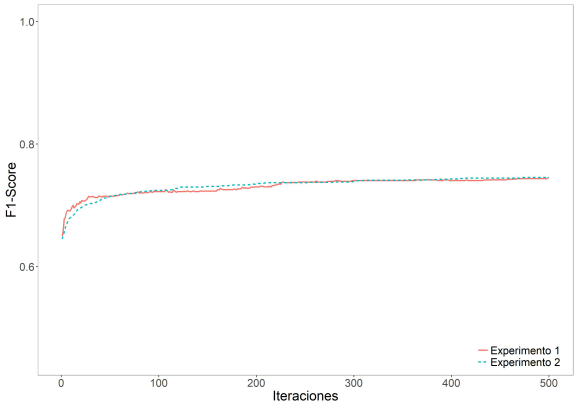

Figura 4.35: Exp1 vs Exp2: train ecoli

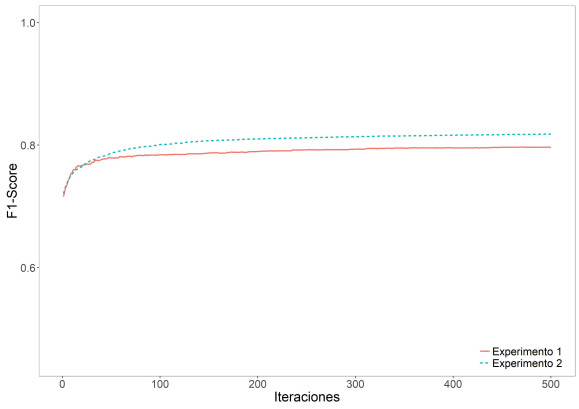

Figura 4.36: Exp1 vs Exp2: train eeg eye

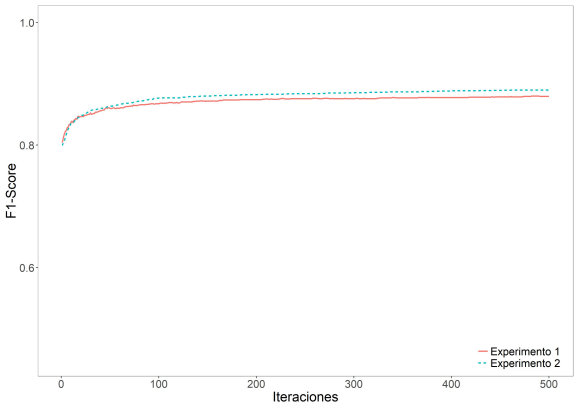

Figura 4.37: Exp1 vs Exp2: train forest type

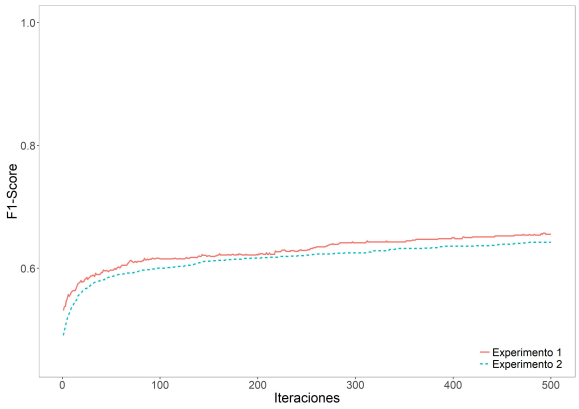

Figura 4.38: Exp1 vs Exp2: train glass

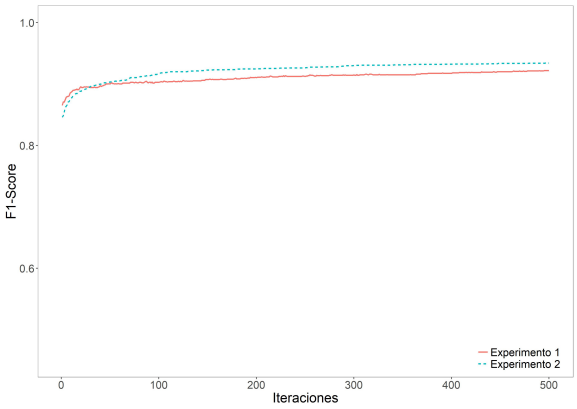

Figura 4.39: Exp1 vs Exp2: train heart

los resultados obtenidos con la configuración del Experimento 1 la del Experimento 2, ver tabla 4.6, no obstante los gráficos de rendimiento sobre la muestra de validación son mejores con el nuevo planteamiento. Además, desde el punto de vista conceptual es menos destructiva la técnica de combinación planteada en este experimento 4.33 frente a la propuesta en el Experimento 1 4.1.

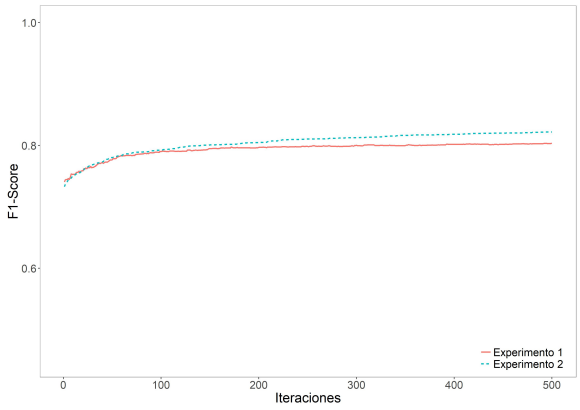

Figura 4.40: Exp1 vs Exp2: train ilpd

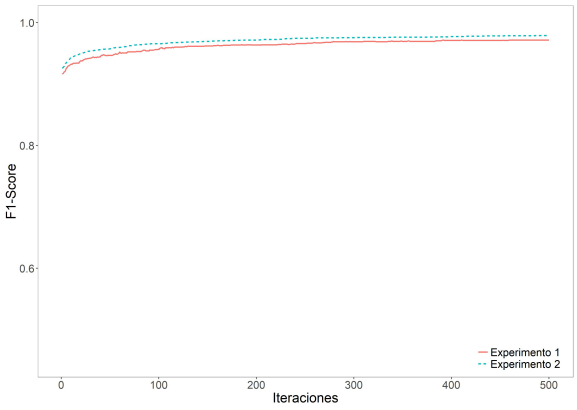

Figura 4.41: Exp1 vs Exp2: train ion

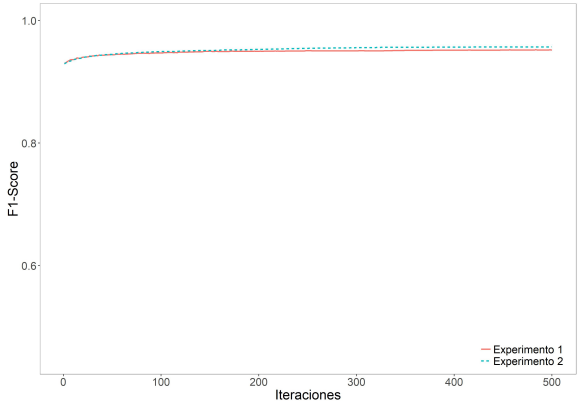

Figura 4.42: Exp1 vs Exp2: train phishing

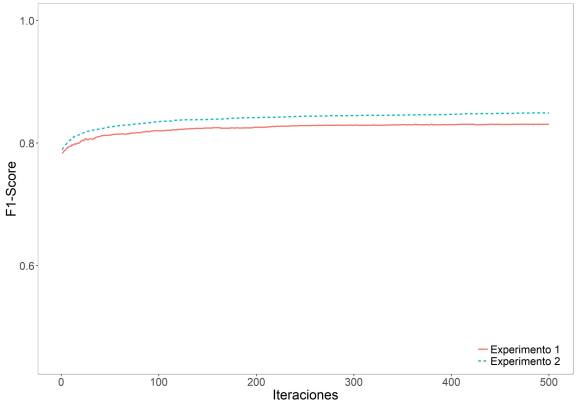

Figura 4.43: Exp1 vs Exp2: train pima

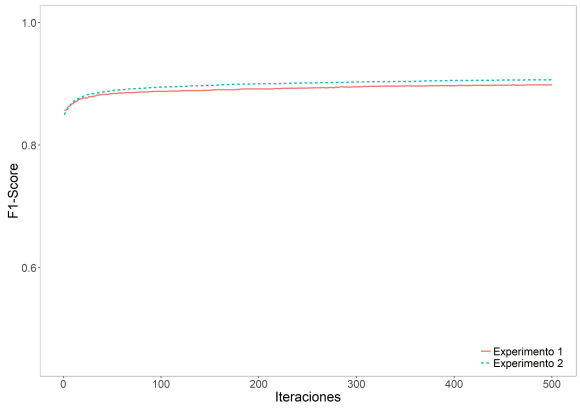

Figura 4.44: Exp1 vs Exp2: train qsar

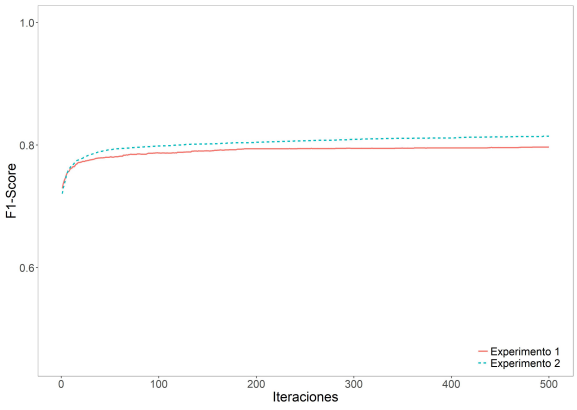

Figura 4.45: Exp1 vs Exp2: train retinopathy

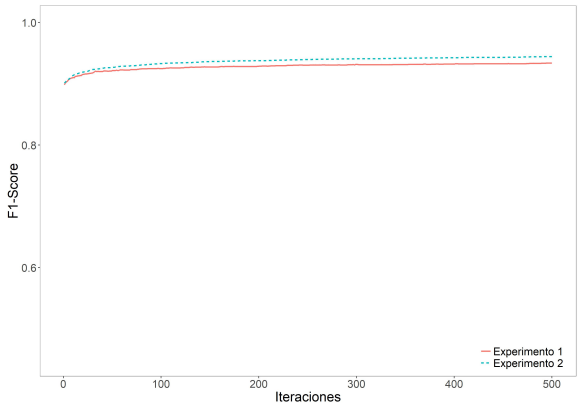

Figura 4.46: Exp1 vs Exp2: train spam

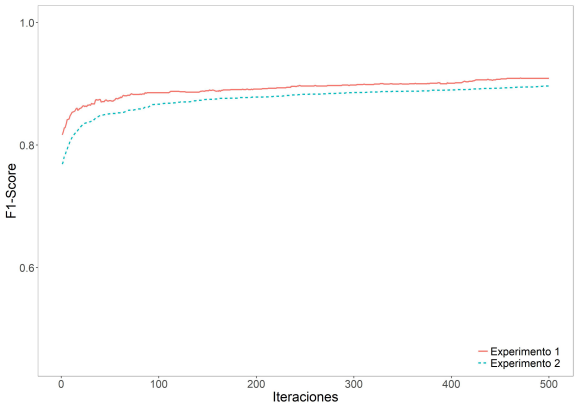

Figura 4.47: Exp1 vs Exp2: train statlog

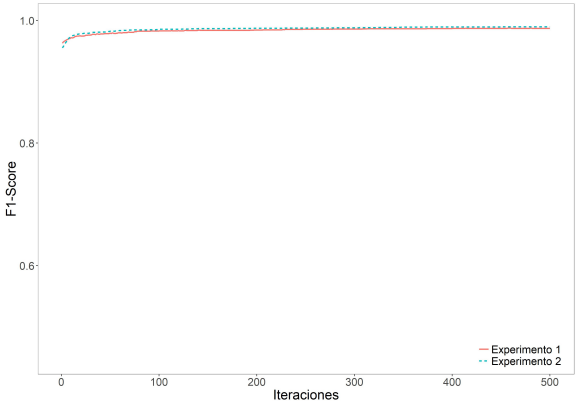

Figura 4.48: Exp1 vs Exp2: train wdbc

4.3. Experimento 3

Vamos a trabajar en torno a la idea de obtener variables nuevas, F_D, que sean lo más "diferentes" posible al conjunto inicial F_0. Hasta ahora, en los experimentos realizados, no se ha introducido ningún tipo de penalización, ni de sistema de corrección que evite que las variables obtenidas sean similares a las que hay en el conjunto inicial.

Tampoco hemos penalizado a los individuos que, dentro de su solución, tienen alguna variable no decodificable, esto es, que no puede ser traducida a una función matemática interpretable. Por tanto puede darse el caso de individuos que, teniendo un número menor de variables generadas que el fijado (cinco en nuestro caso) con buen rendimiento, salgan en las primeras posiciones de la ordenación por la función de evaluación, aunque cuenten con variables no decodificables.

Dada la particularidad de las variables obtenidas en los experimentos no es sencilla la comparación semántica de las expresiones obtenidas, ni tampoco la comparación de las fórmulas aplicadas sobre los datos originales. Se puede dar el caso, al no haber un proceso de consolidación de las expresiones, de que una variable se sume y reste a sí misma suficiente número de veces como para obtener, como resultado final, esa misma variable. También podría ocurrir que el resultado de aplicar una expresión sea casi igual que una de las variables de F_0 salvo una ligerísima variación a modo de constante, en estos casos la comparación directa de los resultados, registro a registro, no nos indicaría la similitud real de las variables.

4.3.1. Planteamiento

La forma más sencilla que hemos encontrado para controlar este tipo de comportamiento es analizar el vector de correlaciones generado entre la nueva variable creada y el conjunto de variables iniciales F_0. De esta forma podremos minimizar la aparición de variables del conjunto inicial F_0 en el conjunto de nuevas variables derivadas F_D.

Entendemos la correlación lineal de Pearson [20] en los términos:

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X.\sigma_Y}$$

Donde $Cov(X,Y)$ hace referencia a la covarianza entre las variables X e Y, medida como:

$$Cov(X,Y) = E[(X-\mu_X).(Y-\mu_Y)]$$

El valor que puede tomar ρ [20] está comprendido entre [-1,1], siendo el límite inferior el punto de máxima correlación lineal inversa y el superior de máxima correlación lineal directa.

El inconveniente que presenta este sistema es que solo estaremos teniendo en cuenta las relaciones lineales entre dos variables. En caso de existir similitudes no lineales esta metodología no será capaz de detectarlas. En cualquier caso parece un buen punto de partida para estudio del problema.

Conjunto datos	Media(Max)	Desv(Max)	Media(Min)	Desv(Min)
ion	0.9395	0.1118	0.0216	0.0917
pima	0.9577	0.0660	0.0373	0.0335
phishing	0.9335	0.1186	0.0364	0.1388
spam	0.9566	0.0878	0.0078	0.0380
glass	0.9524	0.1404	0.0446	0.0555
wdbc	0.9884	0.0228	0.0225	0.0266
retinopathy	0.8974	0.1439	0.0096	0.0135
eeg_eye	0.9151	0.1383	0.0789	0.1359
drive_diag	0.9858	0.0385	0.0044	0.0064
ilpd	0.8930	0.1930	0.0156	0.0181
qsar	0.9802	0.0407	0.0040	0.0048
statlog	0.9792	0.0421	0.0136	0.0249
ecoli	0.9767	0.0631	0.0374	0.0321
forest_type	0.9846	0.0363	0.0188	0.0203
heart	0.965	0.0672	0.0526	0.1351
Total	0.9248	0.0401	0.0270	0.0209

Tabla 4.7: Correlaciones Experimento 2

Comenzaremos estudiando los resultados del Experimento 2 en términos de correlación lineal. Las conclusiones se muestran en la tabla 4.7. Para obtener estos datos se ha procedido de la siguiente forma:

- Obtenemos el conjunto de variables finales de cada una de las 48 ejecuciones realizadas en el Experimento 2, para cada conjunto de datos.
- Para cada una de las cinco variables obtenidas en cada una de las 48 ejecuciones obtenemos la correlación contra las variables del conjunto inicial F_0, sin incluir la variable objetivo. Obtenemos el valor absoluto de las correlaciones dado que nos es indiferente que la relación sean directa o inversa, pero nos afecta a la hora de obtener el máximo o el mínimo.
- Del vector de correlaciones obtenido, para cada variable nueva contra el conjunto inicial nos quedamos con el máximo valor. El objetivo es detectar si alguna de las nuevas variables está muy correlacionada con cualquiera del conjunto inicial.

- Obtenemos, por tanto, un valor único máximo para cada variable nueva, generando un vector de correlaciones máximo de cinco valores, uno para cada variable nueva de F_D.
- De ese vector que indica la correlación de cada variable de la ejecución nos quedamos con el máximo, indicando la correlación máxima general de cada ejecución.
- De las 48 ejecuciones, para cada conjunto de datos, obtenemos el valor medio y la desviación.
- Obtenemos por tanto un valor medio de la correlación y la desviación típica para cada conjunto de datos, basado en los resultados de las 48 ejecuciones y las cinco variables nuevas creadas, comparadas con las variables de los datos iniciales F_0.
- Para el caso del mínimo realizamos los mismo pasos cambiando máximo por mínimo.
- Para obtener los valores se han eliminado los casos en los que, por tener la muestra desviaciones cero, el resultado de la correlación es infinito.
- En caso de que la solución de algún individuo no tenga las cinco variables fijadas por defecto el vector de correlación se obtendrá de las disponibles.

Los resultados, para el valor máximo, muestran medidas de correlación muy altas, con desviaciones pequeñas, lo que indica que la mayoría de las ejecuciones están cercanas a la media. Por tanto parte del esfuerzo computacional realizado se está perdiendo en variables que son muy similares, en términos lineales, a las que forman el conjunto inicial F_0. No obstante los valores mínimos muestran como la metodología propuesta si está siendo capaz de generar nuevas variables, relacionadas con la variable objetivo, pero con baja relación lineal con el conjunto de datos inicial. Al igual que en el caso del máximo las desviaciones son bajas, por tanto la mayoría de los resultados están cercanos a la media.

Parece que hemos encontrado un camino de mejora claro si consiguiéramos reducir los valores medios de la correlación máxima, intentando buscar variables nuevas menos correlacionadas con el conjunto inicial y por tanto que puedan aportar más al introducirlas en el clasificador.

Para abordar los cambios necesarios en el algoritmo evolutivo se han introducido una serie de modificaciones en la programación. Hay que destacar que la penalización como tal, basada en correlación, no nos ha parecido la mejor opción. Principalmente por el hecho de que la asignación de la penalización es complicada, ¿cuanto hay que penalizar una expresión por el hecho de tener alguna variable con alta correlación con F_0? Podría darse el caso de que una solución de mucha calidad

no fuera seleccionada porque la penalización hace que descienda en la ordenación por la función de evaluación.

Hemos realizado los cambios:

- Las variables generadas por los individuos, hasta ahora, se clasificaban como "T", variable traducidas completamente, y "NT" variable no traducida completamente y por tanto no válida. Hemos procedido a generar una nueva categoría "C" que indica que la variable obtenida tiene una correlación superior a un valor ϵ prefijado.
- Esta nueva categoría nos permite mejorar el sistema de cruce de dos individuos.
- El nuevo sistema de cruce ya no trabaja con un punto de corte a partir del cual realiza el intercambio genético, ver figura 4.33, ahora trabaja a nivel de intercambio de variables completas.
- A la hora de seleccionar las variables de un padre que van a ser reemplazas por las de otro lo haremos en función de un número aleatorio "n", que indica el número de variables a reemplazar. El reemplazamiento se hará sobre la lista de variables ordenadas, teniendo en cuenta que primero se pondrán las variables con "NT", después las variables con "C" y por último las que llevan la etiqueta "T". De esta forma priorizamos la aportación al hijo de las variables bien traducidas y con menor correlación frente al resto.

Como parámetros del modelo se han empleado los mismos que en el Experimento 2, ver tabla 4.4, más el parámetro $\epsilon = 0{,}8$ (marcamos como "C" las nuevas variables con correlación superior a 0.8, tanto directa como inversa. La programación es la misma que la del Experimento 2 a excepción de las modificaciones comentadas en la función de cruce y todo lo que se deriva de ellas.

4.3.2. Resultados

Como aproximación inicial al experimento hemos trabajado con una muestra reducida de los conjuntos de datos, ver sección 3.3. En la tabla 4.8 se pueden ver los resultados obtenidos con la nueva configuración, comparados con los obtenidos en el Experimento 1 y el Experimento 2.

El planteamiento no ha ofrecido buenos resultados dado que de los cuatro conjuntos de datos, solo en uno conseguimos diferencias significativas, frente a los tres que se obtenían con los dos experimentos anteriores. Los valores de sobreajuste están en torno a los obtenidos en los dos experimentos anteriores.

Conjunto de datos	Experimento	$F1(F_0)$	$F1(F_A)$	Diferencia	p-valor	Sobreajuste	Mejora $\alpha = 0{,}05$
ecoli	1	56.31	58.32	2.01	0.025369	29.17	1
ecoli	2	57.10	61.37	4.26	0.000551	22.92	1
ecoli	3	58.34	58.41	0.07	0.476314	45.83	0
glass	1	41.14	45.15	4.01	0.001378	25.00	1
glass	2	38.40	40.61	2.22	0.054569	35.42	1
glass	3	40.08	40.41	0.33	0.418435	29.17	0
heart	1	75.65	78.47	2.83	0.010727	29.17	1
heart	2	75.49	77.66	2.17	0.037366	31.25	1
heart	3	76.03	79.47	3.45	0.001705	22.92	1
ion	1	88.11	88.36	0.25	0.385793	43.75	0
ion	2	89.17	89.66	0.48	0.209617	35.42	0
ion	3	89.64	90.44	0.8	0.133345	31.25	0

Tabla 4.8: Comparativa resultados experimentos: 1,2 y 3

En la tabla 4.9 se muestran los valores de correlación entre el nuevo planteamiento y los obtenidos en el Experimento 2. Vemos como los valores de correlaciones medias máximas han descendido de forma considerable en todos los conjuntos de datos, mientras que los de correlaciones mínimas son similares. Las correlaciones máximas se sitúan por debajo del valor de $\epsilon = 0{,}8$, fijado por defecto.

Conjunto datos	Exp2 Media(Max)	Exp3 Media(Max)	Exp2 Media(Min)	Exp3 Media(Min)
ion	0.9395	0.6401	0.0216	0.0062
glass	0.9524	0.6762	0.0446	0.0455
ecoli	0.9767	0.7496	0.0374	0.0683
heart	0.9650	0.7411	0.0526	0.0328
Total	0.9584	0.7017	0.0391	0.0382

Tabla 4.9: Correlaciones Experimento 2 vs Experimento 3

En las figuras 4.49,4.50,4.51 y 4.52 se pueden ver los gráficos de evolución de las ejecuciones en los cuatro conjuntos de prueba sobre la muestra de validación. En tres de ellos (glass, heart, ion) las gráficas son casi idénticas a las obtenidas en el Experimento 2, no obstante, en el gráfico del conjunto de datos "ecoli" se observa una diferencia considerable, obtenido valores de ajuste superiores. Estos resultados podrían estar relacionados con el incremento del sobreajuste que se aprecia en dicho conjunto pasando de valores inferiores al 30 % a estar por encima del 45 %.

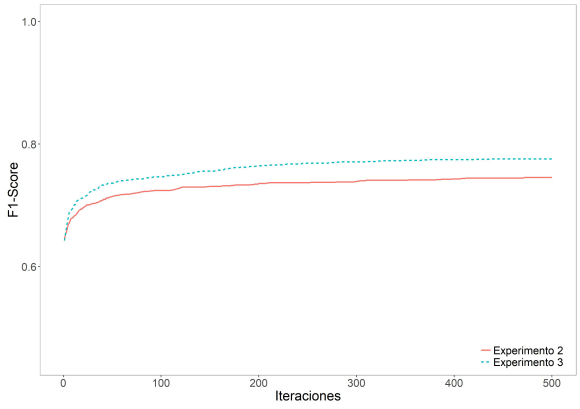

Figura 4.49: Exp2 vs Exp3: train ecoli

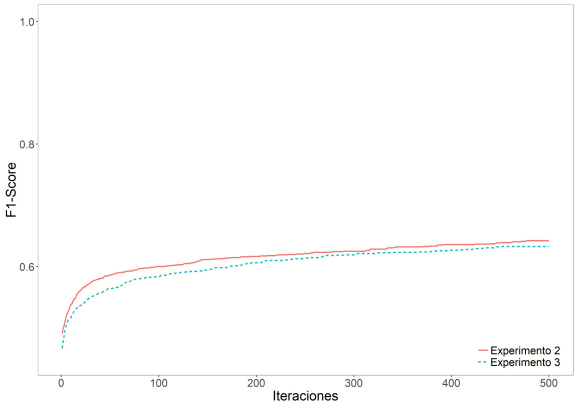

Figura 4.50: Exp2 vs Exp3: train glass

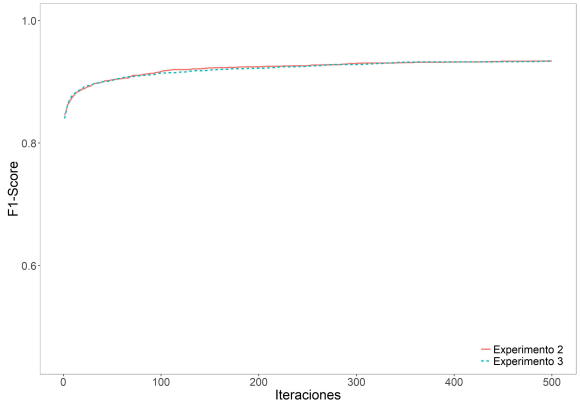

Figura 4.51: Exp2 vs Exp3: train heart

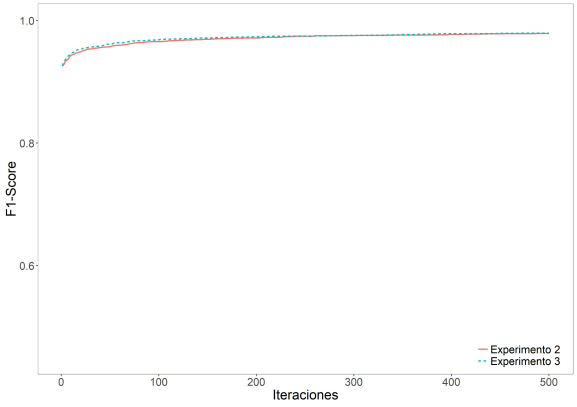

Figura 4.52: Exp2 vs Exp3: train ion

4.3.3. Conclusiones

La disminución media de la correlación máxima es del 25.66 % 4.9, valor significativo que indica que la metodología propuesta reduce la correlación entre las nuevas variables F_D y las variables iniciales F_0.

Si acudimos a los gráficos de evolución, tres de ellos son prácticamente iguales, pero el correspondiente al conjunto "ecoli" presenta diferencias importantes que podrían derivar en un mayor sobreajuste.

No obstante la pérdida de rendimiento hace necesaria una revisión y reflexión en torno a los cambios propuestos, dado que si bien han reducido la correlación esto no se ha traducido en una mejora del rendimiento del sistema.

4.4. Experimento 4

El objetivo de este experimento es añadir mayor capacidad expresiva a la gramática y analizar si esto redunda en una mejora en los resultados obtenidos, en concreto trabajaremos con potencias.

4.4.1. Planteamiento

Partimos de la gramática inicial, recogida en la tabla 4.1, añadiéndole las funciones de potencia y de raíz n-ésima. Para simplificar la gramática y el proceso de decodificación vamos a trabajar solo con con los número recogidos en $< n >$, en concreto {2,3,4,5,6,7,8,9}. Consideramos que, como punto de partida, la inclusión de las potencia y raíz de los números básicos, quitando el primer puesto que no aportaría nada, es suficiente para evaluar si hay una posible mejora en los resultados.

Gramática
$< expr > ::= < expr >< op >< expr > \mid < func > (< expr >) \mid < var >^{n} \mid abs(< var >)^{(1/n)} \mid < var >$
$< func > ::= sin \mid cos \mid f\_log \mid func\_sigm$
$< op > ::= + \mid - \mid * \mid /$
$< n > ::= 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
$< var >$ ::= variables independientes del conjunto de datos

Tabla 4.10: Gramática Experimento 4

Los parámetros de configuración son los mismos que los empleados en el Experimento 2, reflejados en la tabla 4.4.

Seguimos la metodología aproximativa, consistente en hacer una prueba inicial con un conjunto reducido de los 15 conjuntos de datos. En concreto emplearemos los ya utilizados en el Experimento 3 {ecoli, glass, heart, ion}.

4.4.2. Resultados

Los resultados obtenidos se muestran en la tabla 4.11. Se puede comprobar que son ligeramente peores que los obtenidos con el Experimento 1 y el Experimento 2. En concreto las pruebas con el conjunto "glass" resultan no significativas a un nivel $\alpha = 0{,}05$, si bien es cierto que las diferencia obtenida es de casi dos puntos porcentuales, lo que haría interesante el uso de la metodología propuesta. Además el p-valor obtenido es de 0.083 que sería significativo con un $\alpha = 0{,}1$ que no es demasiado alto y da un margen de confianza suficientemente.

En las figuras 4.53, 4.54, 4.55 y 4.56 podemos ver los gráficos de evolución de los cuatro conjuntos de datos, comparando los resultados obtenidos en el Experimento 2 y el Experimento 4.

Conjunto de datos	Experimento	$F1(F_0)$	$F1(F_A)$	Diferencia	p-valor	Sobreajuste	Mejora $\alpha = 0{,}05$
ecoli	1	56.31	58.32	2.01	0.025369	29.17	1
ecoli	2	57.10	61.37	4.26	0.000551	22.92	1
ecoli	4	58.29	60.96	2.67	0.005556	28.00	1
glass	1	41.14	45.15	4.01	0.001378	25.00	1
glass	2	38.40	40.61	2.22	0.054569	35.42	1
glass	4	39.10	41.00	1.90	0.083416	35.42	0
heart	1	75.65	78.47	2.83	0.010727	29.17	1
heart	2	75.49	77.66	2.17	0.037366	31.25	1
heart	4	73.95	76.14	2.19	0.015626	25.00	1
ion	1	88.11	88.36	0.25	0.385793	43.75	0
ion	2	89.17	89.66	0.48	0.209617	35.42	0
ion	4	89.12	89.42	0.30	0.339068	35.42	0

Tabla 4.11: Comparativa resultados experimentos: 1,2 y 4

Los gráficos son casi idénticos, a excepción del obtenido con el conjunto "ecoli", donde la curva del Experimento 4 es claramente mejor que la del 2. La mejora que se visualiza no repercute en un incremento de las diferencias que podría ser debido al incremento en el sobreajuste, que pasa del 22.92 % al 28 %.-

4.4.3. Conclusiones

A la vista de los resultados no podemos afirmar que las mejoras expresivas en la gramática redunden en un incremento del rendimiento. En concreto los resultados obtenidos en el conjunto "glass" son ligeramente peores que los obtenidos en los Experimentos 1 y 2, si bien la diferencia es pequeña.

Es difícil extraer conclusiones definitivas puesto que el ajuste depende mucho del conjunto de datos empleado.

Es interesante destacar que en los resultados reflejados en la tabla 4.11, no se aprecia, como cabría esperar, un incremento del sobreajuste producido por la modificación de la gramática.

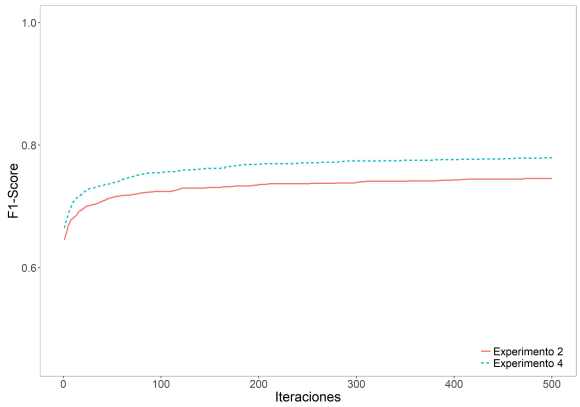

Figura 4.53: Exp2 vs Exp4: train ecoli

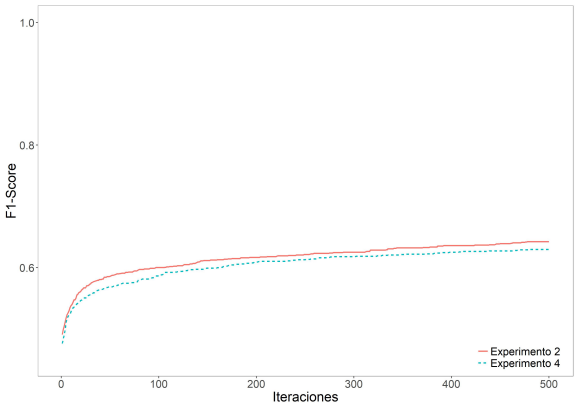

Figura 4.54: Exp2 vs Exp4: train glass

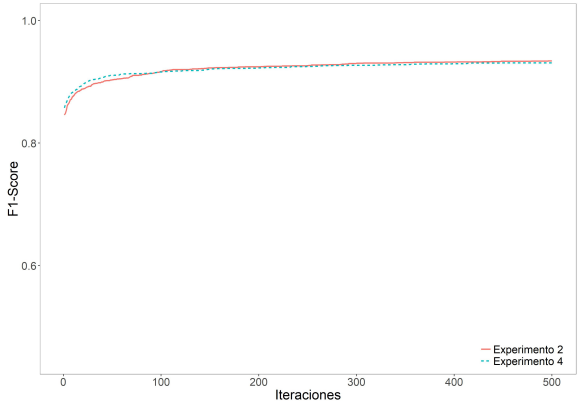

Figura 4.55: Exp2 vs Exp4: train heart

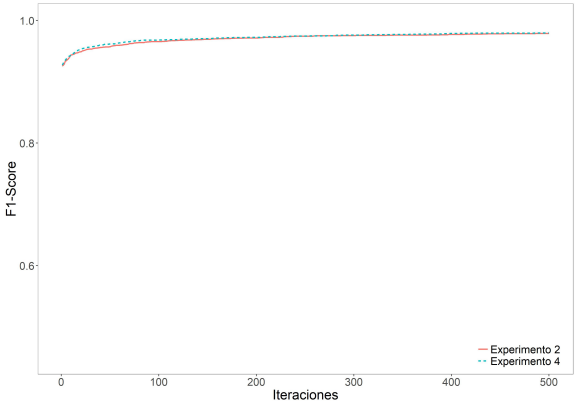

Figura 4.56: Exp2 vs Exp4: train ion

4.5. Experimento 5

Partimos de la programación del Experimento 2 y realizamos una serie de modificaciones:

- En cada cruce generamos dos descendientes a partir de los dos padres.
- Realizamos una sustitución directa de los hijos por los padres de la anterior generación (λ,μ).

El resto de la configuración del experimento se encuentra en la tabla 4.12. La gramática del experimento se encuentra reflejada en la tabla 4.1.

Parámetro	Valor
Variables creadas	5
Repeticiones	48
Iteraciones	500
Población	300
Longitud gen	30
Wrapping	1
Tipo selección	Tournament
$k1$	2
$k2$	2
Elitismo	5
Selección de supervivientes	(μ,λ)
Probabilidad cruce	0.8
Probabilidad mutación	0.05
Clasificador	CART

Tabla 4.12: Parametrización Experimento 5

4.5.1. Resultados

Ejecutamos el experimento con la muestra reducida. Los resultados obtenidos, que se muestran en la tabla 4.13, no son buenos dado que de los cuatro conjuntos solo en uno de ellos, "heart", se obtiene una mejora significativa. Por tanto los cambios no han tenido el efecto deseado.

Los gráficos de evolución comparada de la muestra de validación del Experimento 5 frente a los obtenidos en el Experimento 2 se pueden ver en las figuras 4.57, 4.58, 4.59 y 4.60.

4.5.2. Conclusiones

Los resultados no han sido buenos y no hemos conseguido mejorar ni alcanzar los resultados del Experimento 2. Las estrategias de cruce y de reemplazo no han tenido el efecto deseado.

Conjunto de datos	Experimento	$F1(F_0)$	$F1(F_A)$	Diferencia	p-valor	Sobreajuste	Mejora $\alpha = 0{,}05$
ecoli	1	56.31	58.32	2.01	0.025369	29.17	1
ecoli	2	57.10	61.37	4.26	0.000551	22.92	1
ecoli	4	58.29	60.96	2.67	0.005556	28.00	1
ecoli	5	57.76	59.39	1.63	0.094793	35.42	0
glass	1	41.14	45.15	4.01	0.001378	25.00	1
glass	2	38.40	40.61	2.22	0.054569	35.42	1
glass	4	39.10	41.00	1.90	0.083416	35.42	0
glass	5	41.14	42.58	1.45	0.129472	29.17	0
heart	1	75.65	78.47	2.83	0.010727	29.17	1
heart	2	75.49	77.66	2.17	0.037366	31.25	1
heart	4	73.95	76.14	2.19	0.015626	25.00	1
heart	5	75.17	78.22	3.05	0.003696	27.08	1
ion	1	88.11	88.36	0.25	0.385793	43.75	0
ion	2	89.17	89.66	0.48	0.209617	35.42	0
ion	4	89.12	89.42	0.30	0.339068	35.42	0
ion	5	89.27	89.5	0.23	0.339383	39.58	0

Tabla 4.13: Comparativa resultados experimentos: 1, 2, 4 y 5

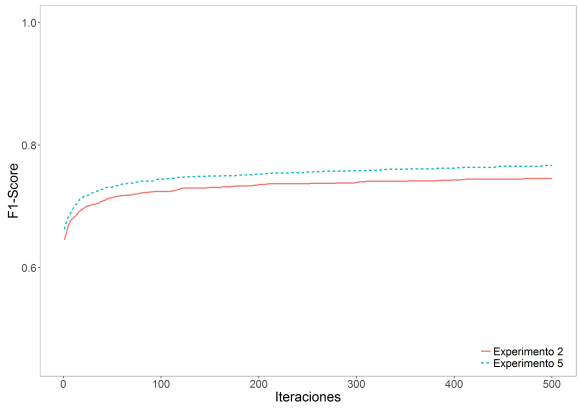

Figura 4.57: Exp2 vs Exp5: train ecoli

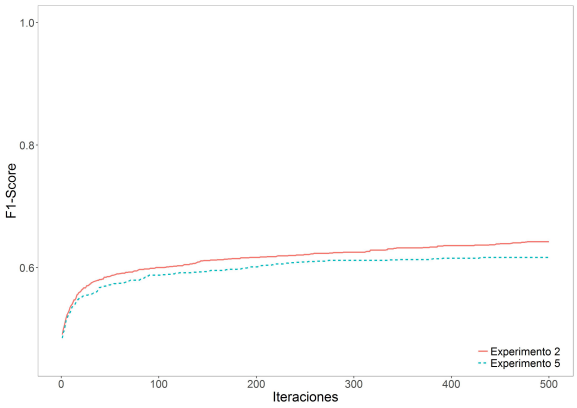

Figura 4.58: Exp2 vs Exp5: train glass

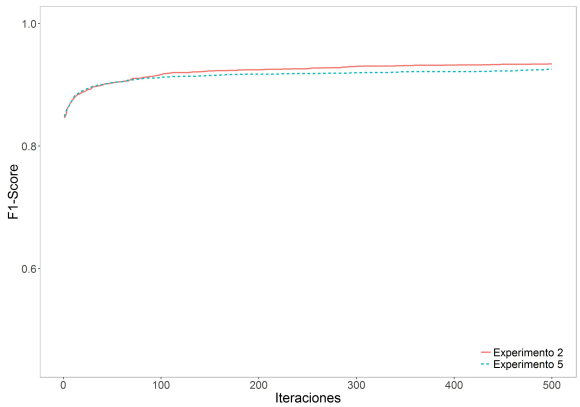

Figura 4.59: Exp2 vs Exp5: train heart

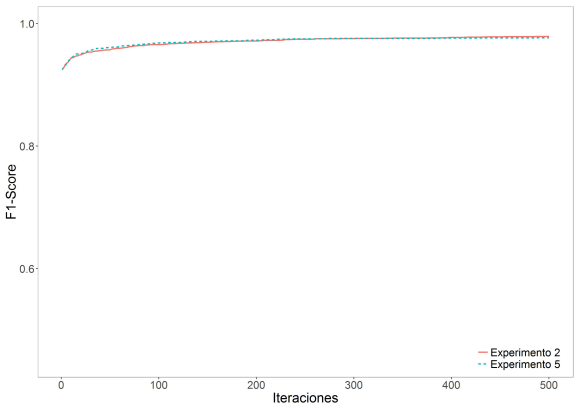

Figura 4.60: Exp2 vs Exp5: train ion

4.6. Experimento 6

Una vez que hemos determinado, con los experimentos anteriores, que el Experimento 2 es el que ofrece mejores resultados vamos a proceder a introducir unos ligeros cambios y a tratar de optimizar los parámetros para mejorar el rendimiento.

En este experimento vamos a mantener la gramática y la configuración empleada en el Experimento 2, ver la tabla 4.1 y tabla 4.4. Vamos a modificar la función de cruce de los individuos. Hasta este momento la función empleada era la descrita en la figura 4.33 que cruzaba dos padres para generar un único hijo. En este experimento el cruce de dos padres generara dos hijos como combinación de ambos. En la figura 4.61 se puede ver la estrategia de cruce aplicada.

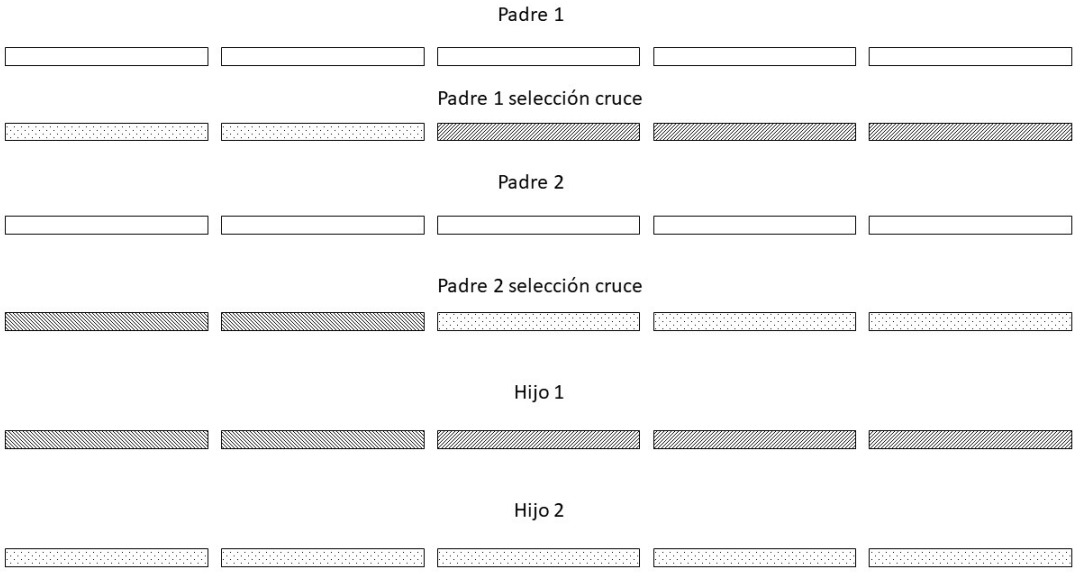

Figura 4.61: Esquema de cruce en el Experimento 6

4.6.1. Resultados

Si comparamos los resultados del Experimento 2 y el 6 son similares, con diferencias medias muy parecidas.

Aplicamos la configuración al resto de ficheros mostrando los resultados en la tabla 4.15. En la tabla 4.16 mostramos una comparación entre el Experimento 6.2 y los resultados obtenidos en el Experimento 2.

Conjunto de datos	Experimento	$F1(F_0)$	$F1(F_A)$	Diferencia	p-valor	Sobreajuste	Mejora $\alpha = 0{,}05$
ecoli	2	57.10	61.37	4.26	0.000551	22.92	1
ecoli	6	58.99	61.57	2.58	0.025421	33.33	1
glass	2	38.40	40.61	2.22	0.054569	35.42	1
glass	6	39.47	41.76	2.29	0.048748	33.33	1
heart	2	75.49	77.66	2.17	0.037366	31.25	1
heart	6	74.44	78.01	3.57	0.004061	31.25	1
ion	2	89.17	89.66	0.48	0.209617	35.42	0
ion	6	88.36	89.10	0.74	0.134726	31.25	0

Tabla 4.14: Comparativa resultados experimentos: 2 y 6

Conjunto de datos	$F1(F_0)$	$F1(F_A)$	Diferencia	p-valor	Sobreajuste	Mejora $\alpha = 0{,}05$
drive_diag	46.35	52.46	6.12	0.000013	10.42	1
ecoli	58.99	61.57	2.58	0.025421	33.33	1
eeg_eye	63.39	66.97	3.59	0.000110	22.92	1
forest_type	75.79	79.40	3.61	0.000302	20.83	1
glass	39.47	41.76	2.29	0.048748	33.33	1
heart	74.44	78.01	3.57	0.004061	31.25	1
ilpd	70.83	69.60	-1.23	0.983352	47.92	0
ion	88.36	89.10	0.74	0.134726	31.25	0
phishing	90.67	90.74	0.07	0.376066	33.33	0
pima	73.24	74.01	0.77	0.090024	31.25	0
qsar	81.41	82.58	1.17	0.012379	37.5	1
retinopathy	63.04	71.08	8.04	0	4.17	1
spam	88.11	88.69	0.58	0.038309	39.58	1
statlog	71.89	77.81	5.92	0.000049	12.5	1
wdbc	93.13	94.75	1.62	0.001853	25	1
Total	71.94	74.57	2.63			73.33 %

Tabla 4.15: Resultados Experimento 6.2

Los datos obtenidos para las diferencias son ligeramente mejores en el Experimento 6.2 que los obtenidos en el 2. En concreto tenemos una diferencia media de 2.63 frente a la que teníamos en el Experimento 2 que era de 2.34, también ha mejorado el sobreajuste medio pasando de 28.61 a 27.64. Con estos resultados incorporamos el nuevo sistema de cruce a la configuración básica.

4.6.2. Ajuste parámetros Experimento 6

Una vez que hemos identificado que los sistemas de cruce y mutación más eficientes son los empleados en el Experimento 6 vamos a trabajar en el ajuste fino de los parámetros del algoritmo genético, con el objetivo de mejorar los resultados obtenidos hasta el momento.

Vamos a trabajar con los parámetros:

Conjunto datos	E6.2 $F1(F_0)$	E2 $F1(F_0)$	E6.2 $F1(F_A)$	E2 $F1(F_A)$	E6.2 Diferencia	E2 Diferencia	E6.2 Sobreajuste	E2 Sobreajuste
drive_diag	46.35	45.02	52.46	48.89	6.12	3.86	10.42	29.17
ecoli	58.99	56.31	61.57	58.32	2.58	2.01	33.33	29.17
eeg_eye	63.39	63.16	66.97	68.24	3.59	5.07	22.92	14.58
forest_type	75.79	74.88	79.40	76.69	3.61	1.81	20.83	29.17
glass	39.47	41.14	41.76	45.15	2.29	4.01	33.33	25
heart	74.44	75.65	78.01	78.47	3.57	2.83	31.25	29.17
ilpd	70.83	69.78	69.60	68.90	-1.23	-0.88	47.92	33.33
ion	88.36	88.11	89.10	88.36	0.74	0.25	31.25	43.75
phishing	90.67	92.06	90.74	92.17	0.07	0.11	33.33	31.25
pima	73.24	74.47	74.01	73.61	0.77	-0.86	31.25	58.33
qsar	81.41	81.52	82.58	82.87	1.17	1.36	37.5	33.33
retinopathy	63.04	62.77	71.08	71.21	8.04	8.44	4.17	4.17
spam	88.11	88.02	88.69	88.70	0.58	0.67	39.58	37.5
statlog	71.89	74.68	77.81	78.49	5.92	3.80	12.50	16.67
wdbc	93.13	91.74	94.75	94.34	1.62	2.60	25.00	14.58
Total	71.94	71.95	74.57	74.29	2.63	2.34	27.64	28.61

Tabla 4.16: Comprativa Experimento 2 vs Experimento 6.2

- **Probabilidad de Cruce:** probabilidad de que dos individuos seleccionados se crucen para generar dos nuevos individuos.
- **Probabilidad de Mutación:** probabilidad de que un elemento del gen que forma el individuo se mute de forma aleatoria.
- ***k*1:** número de individuos que participan en la selección por torneo de la ascendencia.
- ***k*2:** número de individuos que participan en la selección por torneo de la nueva población.
- **Elitismo:** número de individuos que se mantienen de una generación a otra, en función de su función de evaluación.

En la tabla 4.17 se muestran los p-valores asociados al contraste de diferencia de medias de las configuraciones evaluadas. En gris más oscuro se muestran los p-valores por debajo de 0.05, confianza del 95 % en que la diferencia de las medias obtenidas es significativa. En gris más claro se muestran los p-valores en el rango (0.05,0.1], diferencia significativa con una confianza del 90 %.

Hemos trabajado únicamente con los cuatro conjuntos de datos en los que no se obtenía mejora con la configuración estándar del Experimento 6, en concreto: ilpd, ion, phishing y pima.

Comenzamos los experimentos de ajuste haciendo variaciones sobre los parámetros más relevantes, la probabilidad de cruce y la probabilidad de mutación, manteniendo los valores de $k1$, $k2$ y el elitismo. Los resultados muestran que los conjuntos ion y phishing pueden llegar a ofrecer resultados significativos, en cambio ilpd y pima no se muestran sensibles a las modificaciones probadas.

Parámetros					Ficheros			
Probabilidad Cruce	Probabilidad Mutación	$k1$	$k2$	Elitismo	ilpd	ion	phishing	pima
0.90	0.100	2	2	5	0.9961	0.0161	0.1195	0.4167
0.70	0.010	2	2	5	0.9551	0.0603	0.2940	0.6278
0.95	0.150	2	2	5	0.9824	0.7539	0.0334	0.8865
1.00	0.200	2	2	5	0.9919	0.7670	0.0532	0.1105
0.90	0.150	2	2	5	0.5750	0.0764	0.3277	0.2930
0.95	0.100	2	2	5	0.6174	0.1135	0.3500	0.0580
0.95	0.050	2	2	5	0.9993	0.1207	0.0400	0.2099
0.85	0.100	2	2	5	0.9967	0.0300	0.0705	0.5515
0.85	0.050	2	2	5		0.5107	0.1123	
0.90	0.075	2	2	5		0.3157	0.1711	
0.90	0.125	2	2	5		0.0023	0.2544	
0.90	0.100	2	2	1		0.0763	0.3664	
0.90	0.100	2	2	3		0.1413	0.0049	
0.90	0.100	2	2	4		0.1152	0.1456	
0.90	0.100	2	3	4		0.0692	0.0477	
0.90	0.100	2	3	5		0.0025	0.4265	
0.90	0.100	2	3	3		0.0041	0.0905	
0.90	0.100	2	4	4		0.0631	0.0024	
0.90	0.100	3	3	4		0.1682	0.1013	
0.90	0.100	2	4	3		0.0080	0.3809	
0.90	0.100	2	4	5		0.0222	0.1057	
0.85	0.150	3	4	4		0.0649	0.3457	
0.90	0.100	3	4	4		0.0552	0.0459	
0.85	0.150	2	4	4		0.1795	0.2175	
0.90	0.100	3	3	4		0.1666	0.2902	
0.90	0.100	3	4	4		0.6057	0.3634	

Tabla 4.17: Ajuste fino de parámetros Experimento 6

Una vez probadas las modificaciones sobre los dos parámetros de probabilidad y comprobando que los conjuntos más prometedores son ilpd e ion, decidimos continuar solo con estos dos ficheros para poder realizar un número mayor de pruebas y por tanto de combinación de nuevos parámetros de ajuste.

En el siguiente bloque solo trabajamos con los conjuntos de datos ion y phishing, probando con un conjunto más reducido de valores en concreto:

- **Probabilidad de cruce:** {0.9,0.85}.
- **Probabilidad de mutación:** {0.1,0.15}.
- ***k*1:** {2,3}.
- ***k*2:** {2,3,4}.
- **Elitismo:** {1,3,4,5}.

Los resultados más prometedores se obtienen con las configuraciones {0.9, 0.1, 2, 3, 4}, {0.9, 0.1, 2, 4, 4} y {0.9, 0.1, 3, 4, 4}.

4.6.3. Evaluación con ajuste fino

Parámetro	Valor
Variables creadas	5
Repeticiones	48
Iteraciones	1000
Población	300
Longitud gen	30
Wrapping	1
Tipo selección	Tournament
$k1$	3
$k2$	4
Elitismo	4
Selección de supervivientes	(μ,λ)
Probabilidad cruce	0.9
Probabilidad mutación	0.1
Clasificador	CART

Tabla 4.18: Ajuste fino Experimento 6

En la tabla 4.18 se muestran los parámetros empleados en la ejecución final del experimento para el árbol de decisión. Los parámetros se han ajustado con base en los resultados expuestos en la tabla 4.17. Además se ha pasado de las 500 ejecuciones que hemos venido realizando a 1.000, dado que en los gráficos de evolución de los algoritmos genéticos, para algunos ficheros, se apreciaba una tendencia creciente en los valores de la función de evaluación, lo que nos hace pensar que con más iteraciones podríamos mejorar los resultados. Obviamente el duplicar el número de iteraciones del experimento dobla el tiempo de ejecución, esto ha ocasionado que haya que trabajar con dos máquinas en cloud de Google en paralelo al ordenador personal ya mencionada anteriormente.

En la tabla 4.19 se muestran los resultados obtenidos a partir de la nueva configuración fina, incluyendo el aumento de iteraciones a 1.000. Con los ajustes hemos conseguido pasar de una mejora en el 73.33 % a obtener mejoría en el 86.66 % de los conjuntos de datos analizados. La diferencia media obtenida de 2.99 puntos porcentuales es mejor que la conseguida en el Experimento 6.2 estándar de 2.63 y la que habíamos obtenido en el Experimento 2 de 2.34 puntos porcentuales.

El sobreajuste también ha mejorado con la nueva configuración situándose en el 24.16 %, frente a los valores que teníamos en el Experimento 6.2 del 27.64 % y en el Experimento 2 del 28.61 %.

Conjunto de datos	$F1(F_0)$	$F1(F_A)$	Diferencia	p-valor	Sobreajuste	Mejora $\alpha = 0{,}05$
drive_diag	45.97	53.13	7.16	0.000016	20.83	1
ecoli	57.34	59.44	2.10	0.034274	27.08	1
eeg_eye	63.90	67.25	3.35	0.000237	27.08	1
forest_type	74.29	79.04	4.76	0.000023	10.42	1
glass	38.88	42.43	3.56	0.005944	27.08	1
heart	75.67	78.12	2.45	0.024376	35.42	1
ilpd	69.36	69.21	-0.15	0.628190	27.08	0
ion	89.16	90.53	1.37	0.025116	33.33	1
phishing	91.43	92.09	0.66	0.009774	33.33	1
pima	73.39	73.85	0.45	0.207754	33.33	0
qsar	81.07	83.37	2.30	0.000089	18.75	1
retinopathy	63.55	71.28	7.72	0.000000	2.08	1
spam	88.31	89.02	0.71	0.043526	35.42	1
statlog	71.51	77.40	5.89	0.000035	14.58	1
wdbc	91.73	94.26	2.52	0.000025	16.67	1
Total	71.70	74.69	2.99		24.16	86.66 %

Tabla 4.19: Resultados Experimento 6 ajuste fino

En la tabla 4.20 se muestran, para cada uno de los ficheros, el porcentaje de ejecuciones, de las 48 realizadas, en las que el mejor valor se obtiene en una iteración superior a las 500, una iteración superior 600... De esta forma podemos comprobar el aporte que ha supuesto el paso de 500 iteraciones a las 1.000 empleadas en este experimento. Casi un 40 % de las ejecuciones, en media, obtienen el mejor valor pasadas las 500 iteraciones.

En las figuras 4.62 hasta las 4.76 se muestran las gráficas de evolución del algoritmo genético sobre la muestra de validación para el Experimento 6 con los parámetros ajustados.

En las figuras 4.77 hasta 4.91 se muestra un ejemplo de las nuevas variables obtenidas en cada conjunto de datos.

Conjunto de datos	Max≥500	Max≥600	Max≥700	Max≥800	Max≥900
drive_diag	52.08 %	37.50 %	29.16 %	12.50 %	6.50 %
ecoli	25.00 %	18.75 %	12.50 %	10.41 %	2.83 %
eeg_eye	50.00 %	41.66 %	35.41 %	18.75 %	12.00 %
forest_type	33.33 %	22.91 %	12.50 %	6.25 %	2.83 %
glass	27.08 %	20.83 %	18.75 %	16.66 %	12.00 %
heart	35.41 %	29.16 %	22.91 %	14.58 %	6.50 %
ilpd	41.66 %	37.50 %	22.91 %	16.66 %	8.33 %
ion	18.75 %	12.50 %	8.33 %	8.33 %	6.50 %
phishing	47.91 %	31.25 %	27.08 %	22.91 %	10.16 %
pima	56.25 %	41.66 %	27.08 %	14.58 %	4.66 %
qsar	50.00 %	41.66 %	31.25 %	18.75 %	6.50 %
retinopathy	68.75 %	54.16 %	47.91 %	37.50 %	16.66 %
spam	47.91 %	43.75 %	29.16 %	16.66 %	8.33 %
statlog	27.08 %	22.91 %	16.66 %	12.50 %	2.83 %
wdbc	16.66 %	12.50 %	4.16 %	4.16 %	0 %
Total	39.86 %	31.25 %	23.05 %	15.41 %	6.94 %

Tabla 4.20: Análisis de la mejora con el paso de 500 iteraciones a 1.000 iteraciones

4.6.4. Conclusiones

Con los datos recogidos en la tabla resumen 4.19, después de realizar el ajuste fino de los parámetros del Experimento 6 podemos concluir que la metodología propuesta es interesante a la hora de mejorar el rendimiento del clasificador empleado, árbol de decisión CART. En concreto obtenemos una mejora significativa en el 86.66 % de los conjuntos de datos analizados, esto es, en 13 de los 15 ficheros. Además la diferencia en rendimiento, si solo tenemos en cuenta los casos en los que hay mejoría esta en 3.43 puntos porcentuales. Esta diferencia es más que significativa desde el punto de vista práctico y justificaría la aplicación de la metodología.

Hay que destacar, como muy positivo, que se ha encontrado una configuración "óptima" que permite un buen comportamiento de la metodología en la mayor parte de los conjuntos de datos. Este punto es de gran interés porque podría haberse dado el caso de que cada conjunto de datos hubiera requerido una configuración propia del algoritmo evolutivo, dificultando en gran medida nuestro intento de generalización.

Tengamos en cuenta que los tiempos elevados de ejecución surgen por la necesidad de validar estadísticamente los resultados, lo que obliga a realizar 48 ejecuciones sobre cada uno de los ficheros. En un problema real no sería necesaria esta validación y se podría realizar una única ejecución para la obtención de las nueva variables.

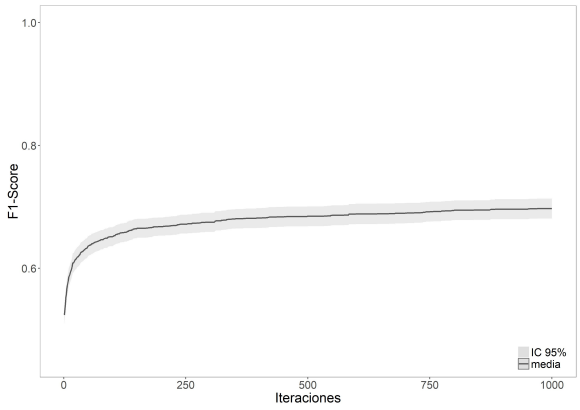

Figura 4.62: Exp6: train drive diag

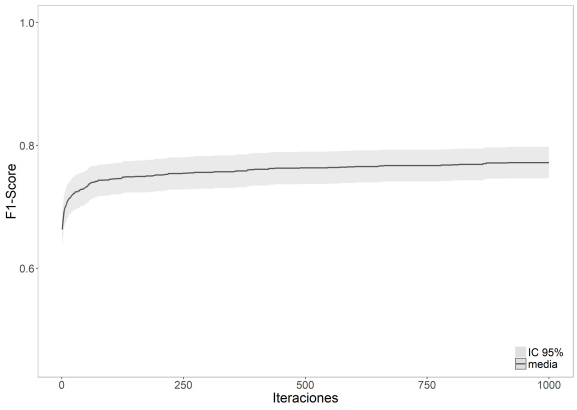

Figura 4.63: Exp6: train ecoli

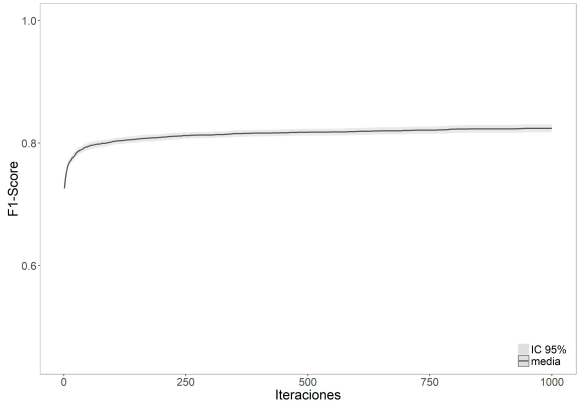

Figura 4.64: Exp6: train eeg eye

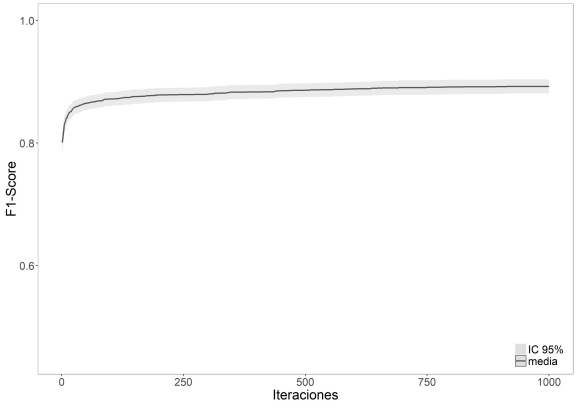

Figura 4.65: Exp6: train forest type

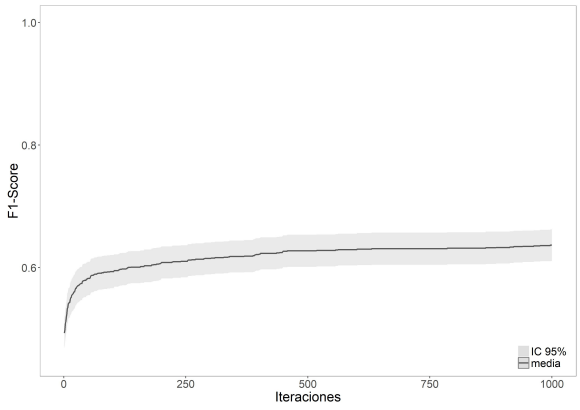

Figura 4.66: Exp6: train glass

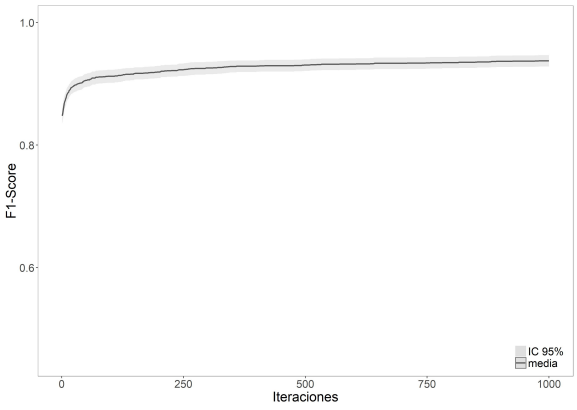

Figura 4.67: Exp6: train heart

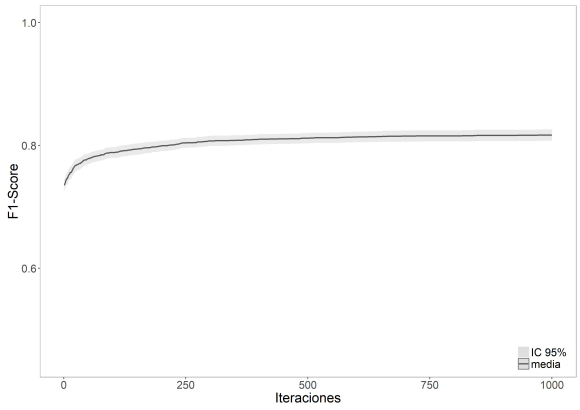

Figura 4.68: Exp6: train ilpd

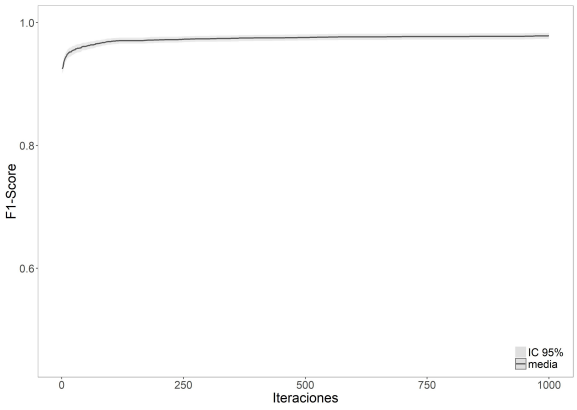

Figura 4.69: Exp6: train ion

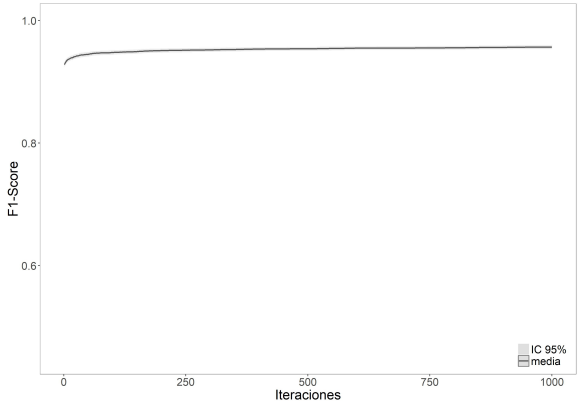

Figura 4.70: Exp6: train phishing

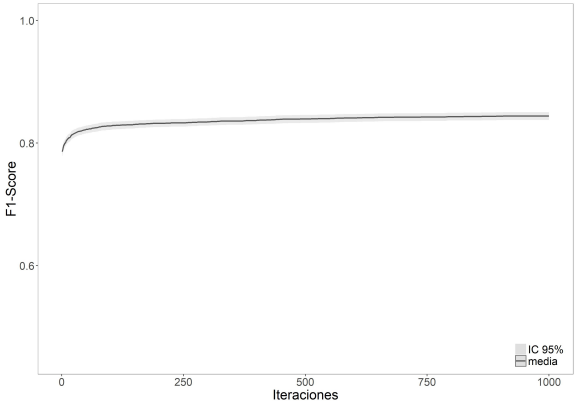

Figura 4.71: Exp6: train pima

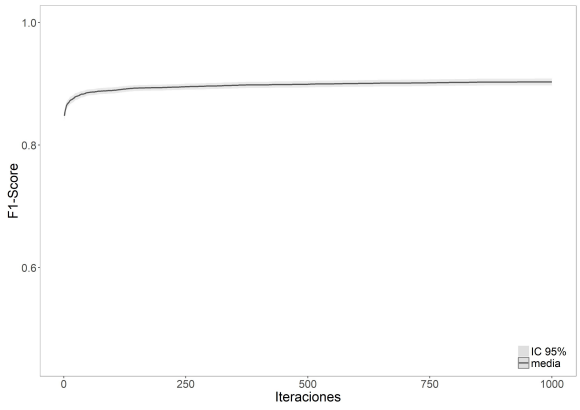

Figura 4.72: Exp6: train qsar

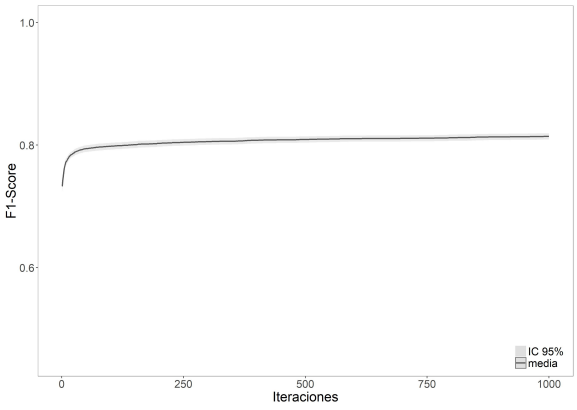

Figura 4.73: Exp6: train retinopathy

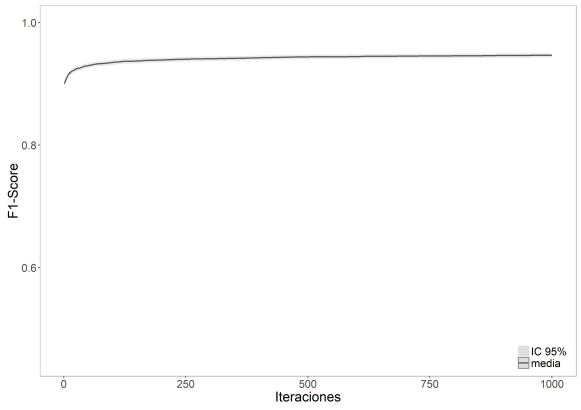

Figura 4.74: Exp6: train spam

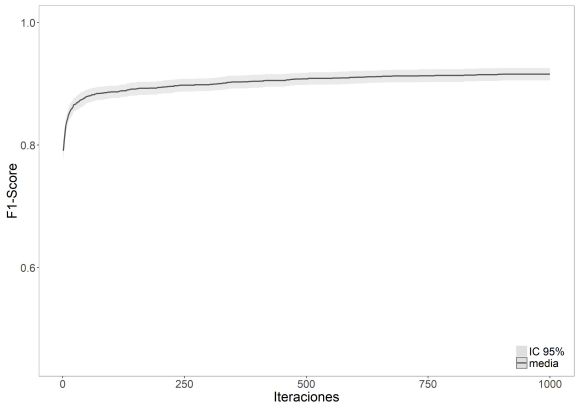

Figura 4.75: Exp6: train statlog

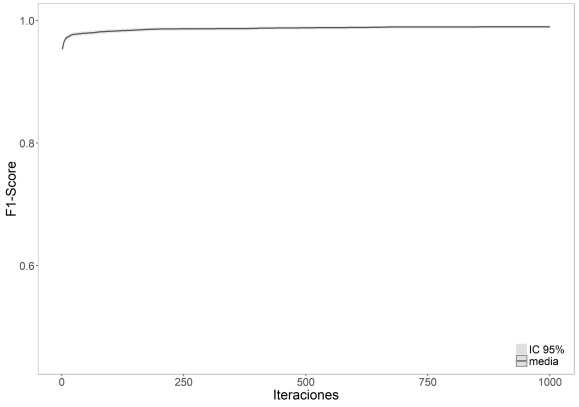

Figura 4.76: Exp6: train wdbc

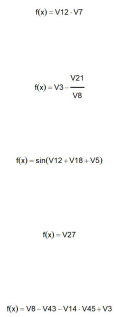

Figura 4.77: Exp6: ecuaciones drive diag

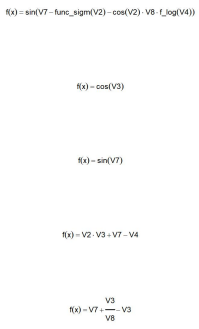

Figura 4.78: Exp6: ecuaciones ecoli

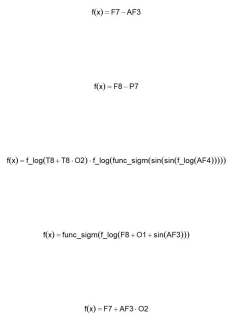

Figura 4.79: Exp6: ecuaciones eeg eye

Figura 4.80: Exp6: ecuaciones forest type

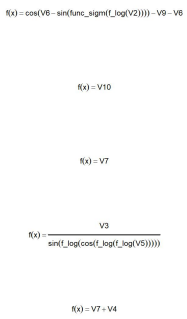

Figura 4.81: Exp6: ecuaciones glass

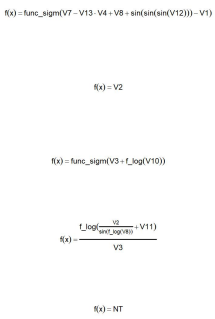

Figura 4.82: Exp6: ecuaciones heart

Figura 4.83: Exp6: ecuaciones ilpd

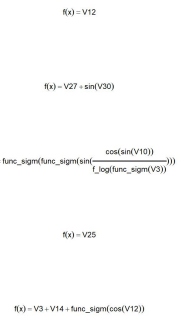

Figura 4.84: Exp6: ecuaciones ion

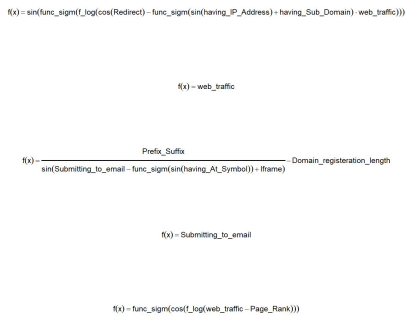

Figura 4.85: Exp6: ecuaciones phishing

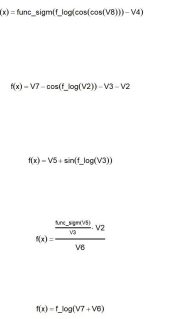

Figura 4.86: Exp6: ecuaciones pima

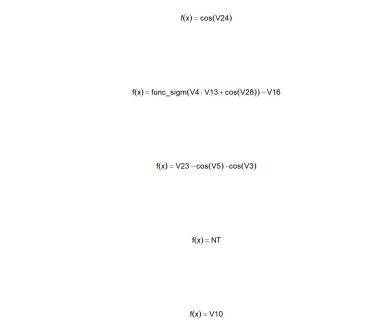

Figura 4.87: Exp6: ecuaciones qsar

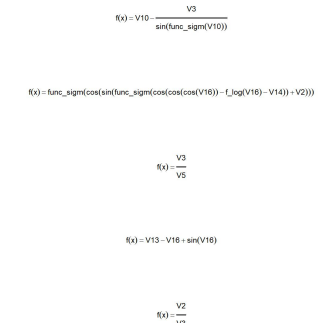

Figura 4.88: Exp6: ecuaciones retinopathy

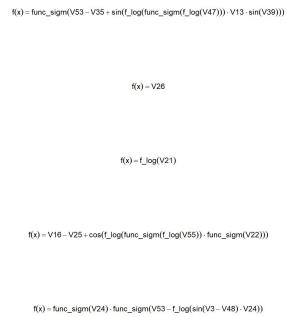

Figura 4.89: Exp6: ecuaciones spam

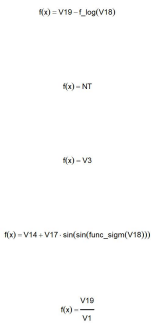

Figura 4.90: Exp6: ecuaciones statlog

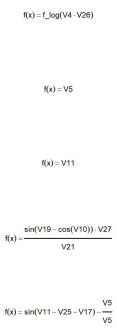

Figura 4.91: Exp6: ecuaciones wdbc

4.7. Experimento 7

En este experimento vamos a probar la configuración obtenida en el Experimento 6 con un nuevo clasificador, k-vecinos [24]. En el Apéndice C se proporciona información sobre el algoritmo y su implementación en los experimentos.

Parámetro	Valor
Variables creadas	5
Repeticiones	48
Iteraciones	1000
Población	300
Longitud gen	30
Wrapping	1
Tipo selección	Tournament
$k1$	2
$k2$	2
Elitismo	5
Selección de supervivientes	(μ,λ)
Probabilidad cruce	0.8
Probabilidad mutación	0.05
Clasificador	k-vecinos

Tabla 4.21: Parametrización Experimento 7

La configuración del Experimento 7 se refleja en la tabla 4.21, destacar que se han mantenido las 1.000 iteraciones, a pesar del coste computacional que suponen a raíz del análisis reflejado en la tabla 4.20. El aumento de iteraciones redunda en un mejor ajuste de la muestra de validación, no obstante habría que realizar un experimento más específico para identificar si esa mejora en validación repercute en una mejora global en la muestra de test o produce un aumento del sobreajuste.

Debido al coste computacional se ha abordado este experimento con una submuestra de los conjuntos de datos, en concreto hemos empleado: glass, forest-type, heart, ilpd, pima, wdbc. El criterio de selección de los conjuntos de datos se ha basado en el número de registros, con el objetivo de minimizar el tiempo de ejecución. También se ha tenido en cuenta que los conjuntos de datos seleccionado haya mejorado en unos casos y en otros no, con la metodología propuesta empleando el clasificador CART. Los resultados obtenidos se pueden ver en la tabla 4.22. Dado que el algoritmo de k-vecinos requiere del ajuste del parámetro k, se han probado tres configuraciones distintas para k tomando valores {5,10,20}.

Los resultados obtenidos, en función de la configuración de la tabla 4.21 y los valores empleados para k, no son buenos. La metodología no es capaz de generar nuevas variables a partir de F_0 que

k	Conjunto de datos	$F1(F_0)$	$F1(F_A)$	Diferencia	p-valor	Sobreajuste	Mejora $\alpha = 0{,}05$
5	forest_type	70.25	70.14	-0.11	0.543874	54.17	0
5	glass	32.01	30.24	-1.77	0.9519	47.92	0
5	heart	80.65	81.59	0.95	0.070513	35.42	0
5	ilpd	65.44	66.31	0.86	0.059795	41.67	0
5	pima	72.46	71.71	-0.75	0.959113	56.25	0
5	wdbc	94.92	94.27	-0.65	0.99835	58.33	0
10	forest_type	69.43	69.67	0.24	0.39572	37.5	0
10	glass	30.09	29.91	-0.18	0.594374	35.42	0
10	heart	80.34	81.05	0.7	0.094278	27.08	0
10	ilpd	68.61	69.13	0.52	0.129285	37.5	0
10	pima	73.18	73.99	0.81	0.031773	33.33	1
10	wdbc	95.03	95.17	0.14	0.28437	35.42	0
20	forest_type	70.05	69.86	-0.19	0.575874	41.67	0
20	glass	37.63	39.41	1.78	0.277619	28.57	0
20	heart	64.68	64.68	0	NaN	0	0
20	ilpd	70.21	70.47	0.26	0.193623	39.58	0
20	pima	72.18	73.14	0.96	0.015286	29.17	1
20	wdbc	94.21	94.38	0.18	0.195387	33.33	0

Tabla 4.22: Resultados Experimento 7

proporcionen una mejora significativa al usar el algoritmo de los k-vecinos. Los mejores resultados se obtienen con k=10, consiguiéndose solo en uno de los casos unas diferencia significativa con $\alpha = 0{,}05$.

Dados estos resultados 4.21 no se ha continuado ejecutando el resto de ficheros, dado que el método queda en entredicho al no obtener resultados positivos casi en el 50 % de los casos.

4.7.1. Conclusiones

La metodología propuesta, con las configuraciones probadas, no da buenos resultados. Por tanto no se puede concluir el planteamiento probado con el árbol de decisión tipo CART [28] en los experimentos anteriores sea generalizable de forma directa.

Estos resultados no descartan que con otra configuración del algoritmo genético y con una prueba exhaustiva sobre los posibles valores de k se pudieran mejorar los resultados. En cualquier caso si podemos concluir con el método no es extrapolable de forma directa y necesita un ajuste fino para cada algoritmo de clasificación, lo que hasta cierto punto le resta utilidad.

Se podrían realizar pruebas más exhaustivas probando con:

- Diferentes valores de configuración para el algoritmo genético.
- Diferentes valores de k.
- Controlar la multicolinealidad de la variables generadas.

- Usar PCA: puede ser de interés aplicar un Análisis de Componentes Principales [29] dado que el algoritmo k-vecinos se basa en la distancia entre observaciones para hacer una clasificación. Dado que no estamos controlando que puedan entrar en el conjunto F_A variables repetidas o con una alta correlación con alguna de las variables de F_0 podríamos mejorar el rendimiento evitando parte del sobreajuste obtenidos.

- Probar con conjuntos de datos con un mayor número de registros, dado que de los análisis que hemos realizado el algoritmo si es capaz de ajustar la muestra de validación. El problema surge cuando el modelo aprendido se extrapola sobre la muestra de test. Un número mayor de registros en muestra de validación y test podría conseguir una mayor generalidad a la hora de clasificar.

4.8. Experimento 8

En este experimento vamos a probar la configuración obtenida en el Experimento 6 con un nuevo clasificador, regresión logística [24]. En el Apéndice B se proporciona información sobre el algoritmo y su implementación en los experimentos, incluyendo el proceso previo de las variables necesario.

Parámetro	Valor
Variables creadas	5
Repeticiones	48
Iteraciones	1000
Población	300
Longitud gen	30
Wrapping	1
Tipo selección	Tournament
$k1$	2
$k2$	2
Elitismo	5
Selección de supervivientes	(μ,λ)
Probabilidad cruce	0.8
Probabilidad mutación	0.05
Clasificador	Regresión logística

Tabla 4.23: Parametrización Experimento 8

La configuración del Experimento 8 se refleja en la tabla 4.23, destacar que se han mantenido las 1.000 iteraciones, a pesar del coste computacional que suponen a raíz del análisis reflejado en la tabla 4.20. Como ya hemos comentado en el experimento anterior, el aumento de iteraciones redunda en un mejor ajuste de la muestra de validación, no obstante habría que realizar un experimento más específico para identificar si esa mejora en validación repercute en una mejora global en la muestra de test o produce un aumento del sobreajuste.

Los resultados obtenidos se pueden ver en la tabla 4.24. Los resultados son moderados ya que conseguimos una mejora casi en la mitad de los casos, en concreto en el 46.66 % de los ficheros analizados.

En la tabla 4.25 se muestra una comparativa entre los resultados obtenidos en el Experimento 6 y los obtenidos en el Experimento 8. Todos los valores son mejores para el Experimento 6 desde los valores del F1-Score, tato para F_0 como para F_A, las diferencias medias o el sobreajuste. Destacar que la capacidad del clasificador CART es mejor que la regresión logística, sobre los conjuntos de datos analizados. En concreto con el CART empleado en el Experimento 6 tenemos un $F1(F_0)$ de 71.71 frente al obtenido en el Experimento 8 de 62.22, la diferencia es más que notable.

Conjunto de datos	$F1(F_0)$	$F1(F_A)$	Diferencia	p-valor	Sobreajuste	Mejora $\alpha = 0{,}05$
drive_diag	12.06	17.83	5.77	0.000002	14.58	1
ecoli	37.65	36.7	-0.95	0.620996	47.92	0
eeg_eye	56.24	58.09	1.85	0.016577	33.33	1
forest_type	48.19	56.24	8.05	0.001631	33.33	1
glass	16.90	19.17	2.28	0.050752	33.33	1
heart	80.65	75.8	-4.85	0.997996	66.67	0
ilpd	68.59	66.39	-2.2	0.958385	54.17	0
ion	84.85	82.44	-2.41	0.911152	45.83	0
phishing	91.44	92.09	0.65	0.018514	20.83	1
pima	72.00	72.34	0.34	0.374389	37.50	0
qsar	77.76	77.52	-0.24	0.604120	43.75	0
retinopathy	58.17	64.04	5.87	0.000077	16.67	1
spam	86.71	85.86	-0.86	0.834740	43.75	0
statlog	61.73	62.82	1.09	0.315129	39.58	0
wdbc	80.36	85.34	4.98	0.007031	33.33	1
Total	62.22	63.51	1.29		37.63	46.66 %

Tabla 4.24: Resultados Experimento 8

Experiento	$F1(F_0)$	$F1(F_A)$	Diferencia	Sobreajuste	Mejora $\alpha = 0{,}05$
6	71.70	74.69	2.99	24.16	86.66 %
8	62.22	63.51	1.29	37.63	46.66 %

Tabla 4.25: Comparativa: Exp6 vs Exp8

En los gráficos de 4.92 al 4.106 se muestra la evolución de la función de evaluación con la muestra de validación.

En las figuras de la 4.108 a la 4.121 mostramos algunas de las ecuaciones generadas con la metodología propuesta para el Experimento 8.

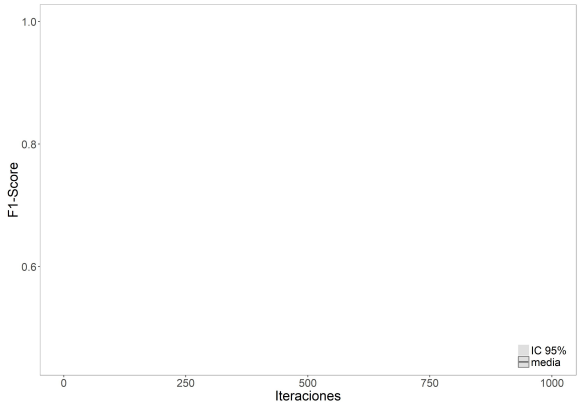

Figura 4.92: Exp8: train drive diag

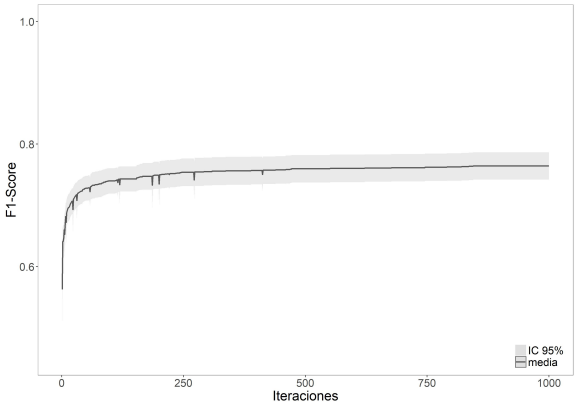

Figura 4.93: Exp8: train ecoli

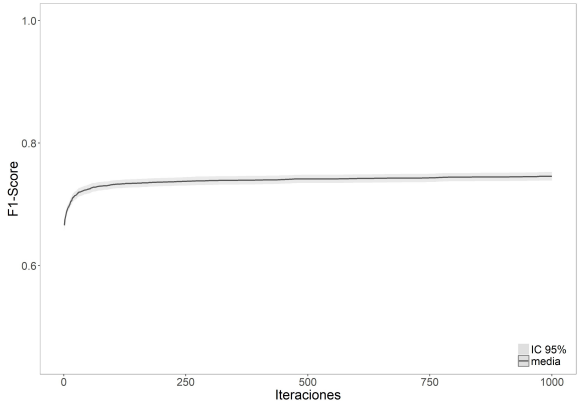

Figura 4.94: Exp8: train eeg eye

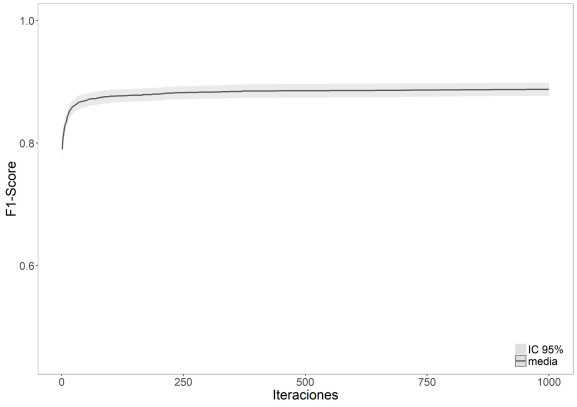

Figura 4.95: Exp8: train forest type

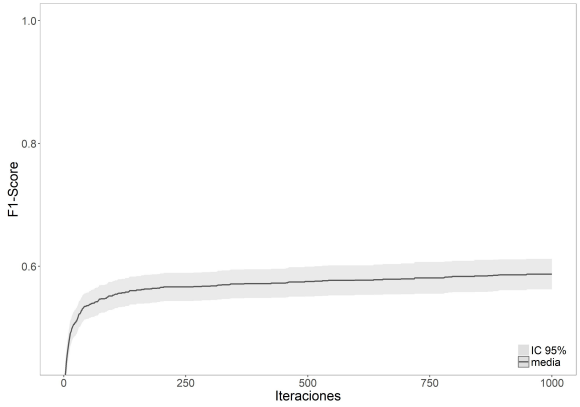

Figura 4.96: Exp8: train glass

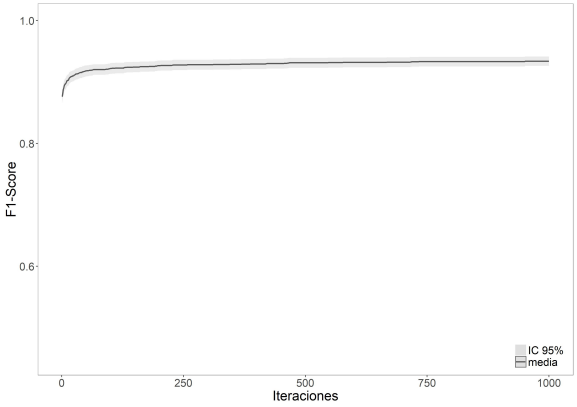

Figura 4.97: Exp8: train heart

4.8.1. Conclusiones

Los resultados obtenidos no son tan buenos como los del Experimento 6 pero tampoco permiten descartar la aplicabilidad del método, con casi un 50 % de los conjuntos de datos ofreciendo mejora el método propuesto podría ser valioso con el clasificador regresión logística.

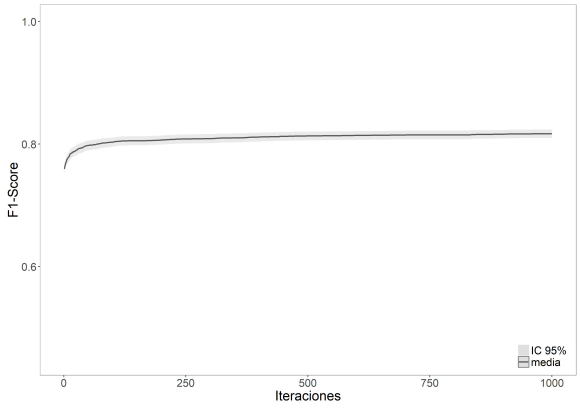

Figura 4.98: Exp8: train ilpd

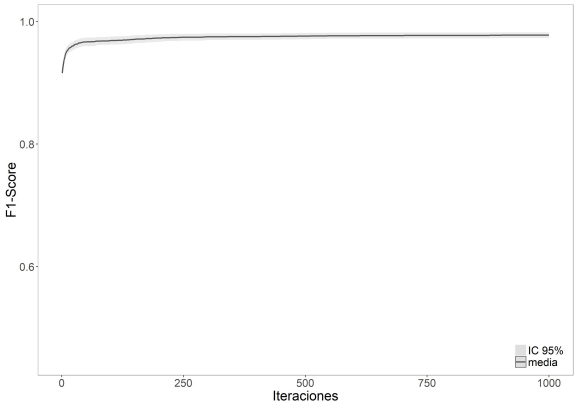

Figura 4.99: Exp8: train ion

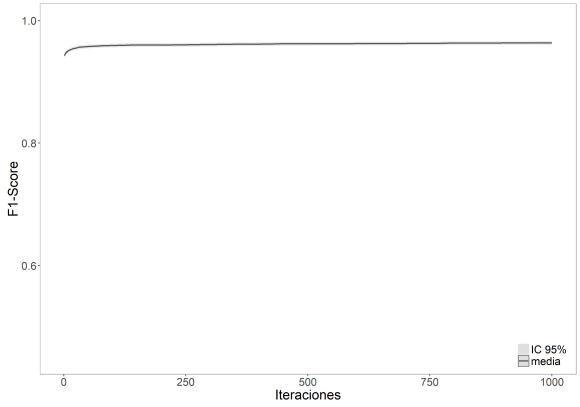

Figura 4.100: Exp8: train phishing

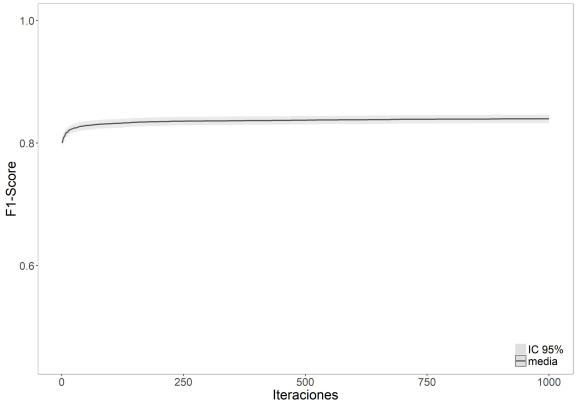

Figura 4.101: Exp8: train pima

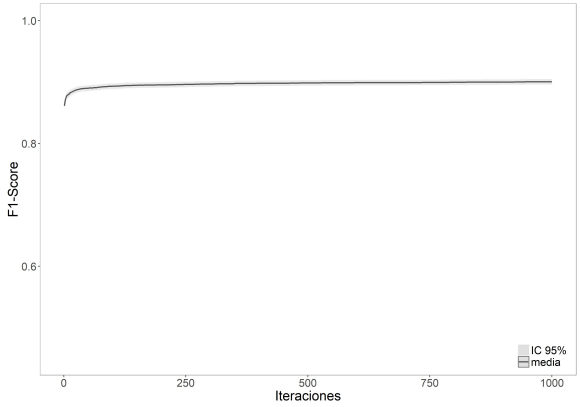

Figura 4.102: Exp8: train qsar

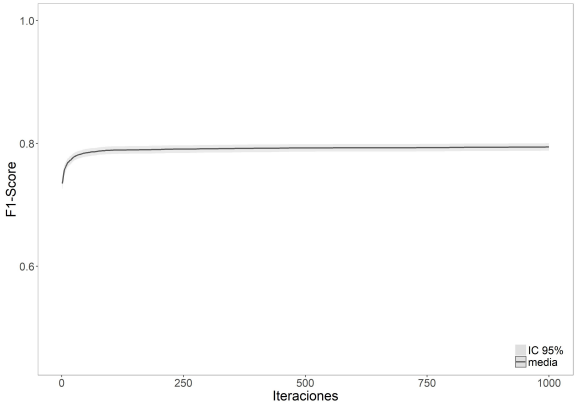

Figura 4.103: Exp8: train retinopathy

Hay que tener en cuenta que el clasificador empleado es muy sensible a las variables correlacionadas y necesita de un tratamiento previo de la información exhaustivo por tanto posibles puntos de mejora, para intentar elevar los resultados obtenidos acercándonos a los conseguidos en el Experimento 6 serían:

- Mejorar el tratamiento previo de las variables de F_A y de las nuevas que se obtengan.

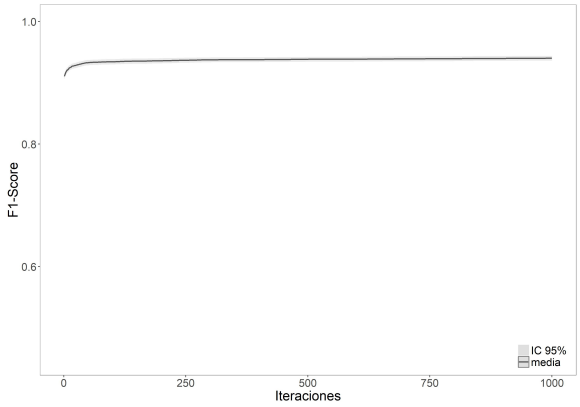

Figura 4.104: Exp8: train spam

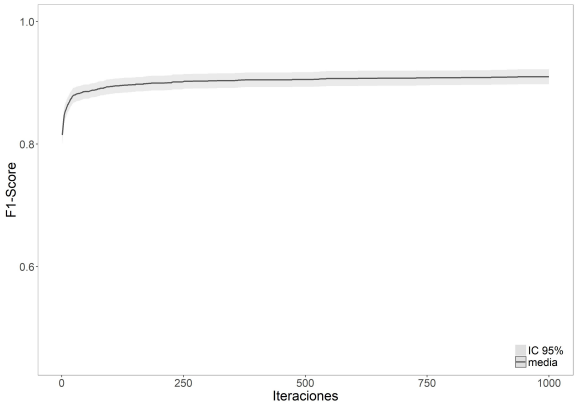

Figura 4.105: Exp8: train statlog

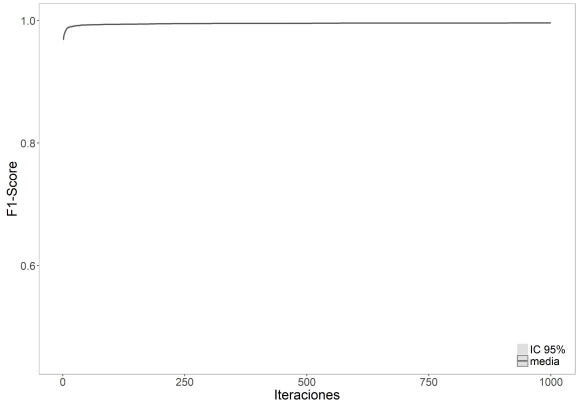

Figura 4.106: Exp8: train wdbc

- Ser más restrictivo con las correlaciones. En este experimento hemos permitido correlaciones inferiores a 0.8, puede que bajando ese umbral se consigan mejoras resultados.
- Usar un PCA para eliminar generar componente independientes.
- Probar con otras configuraciones del algoritmo genético. Podría ocurrir que la configuración obtenida en el Experimento 6 no fuera generalizable como la más eficiente para cualquier clasificador.

.

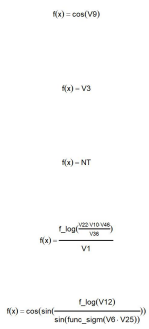

Figura 4.107: Exp8: ecuaciones drive diag

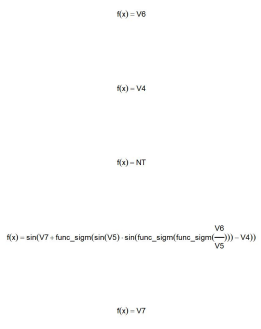

Figura 4.108: Exp8: ecuaciones ecoli

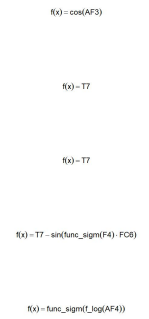

Figura 4.109: Exp8: ecuaciones eeg eye

Figura 4.110: Exp8: ecuaciones forest type

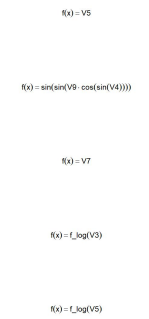

Figura 4.111: Exp8: ecuaciones glass

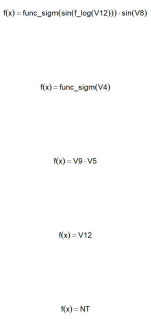

Figura 4.112: Exp8: ecuaciones heart

Figura 4.113: Exp8: ecuaciones ilpd

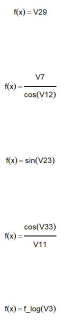

Figura 4.114: Exp8: ecuaciones ion

Figura 4.115: Exp8: ecuaciones phishing

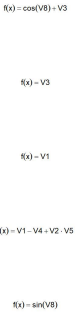

Figura 4.116: Exp8: ecuaciones pima

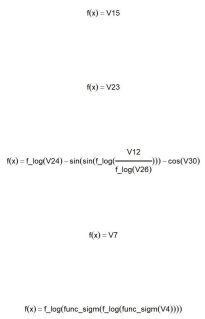

Figura 4.117: Exp8: ecuaciones qsar

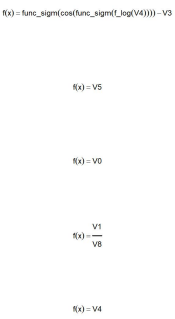

Figura 4.118: Exp8: ecuaciones retinopathy

$$f(x) = f\_log(f\_log(f\_log(sin(func\_sigm(sin(V14))))))$$

$$f(x) = cos(V49 \cdot f\_log(f\_log(V54)) - V50)$$

$$f(x) = cos(V55)$$

$$f(x) = cos(f\_log(V56))$$

$$f(x) = V10 - f\_log(V21) + V44$$

Figura 4.119: Exp8: ecuaciones spam

$$f(x) = V11$$

$$f(x) = sin(V3 + V18 + func\_sigm(sin(V15)))$$

$$f(x) = V3$$

$$f(x) = sin(cos(V2 - V17))$$

$$f(x) = V6$$

Figura 4.120: Exp8: ecuaciones statlog

$$f(x) = \frac{V13}{V18}$$

$$f(x) = V22$$

$$f(x) = V31$$

$$f(x) = \frac{cos(cos(f\_log(V16)))}{V24} - func\_sigm(V22 \cdot V22)$$

$$f(x) = V4$$

Figura 4.121: Exp8: ecuaciones wdbc

Capítulo 5

Conclusiones

En el presente documento se han realizado 8 experimentos para evaluar la metodología propuesta. El objetivo de dicha metodología era encontrar un procedimiento general que permitiera la creación de variables derivadas, a partir de un conjunto inicial F_0, mediante su combinación y la de operadores matemáticos. La construcción de las nuevas variables, que junto con las iniciales pasan a forma el conjunto F_A, se realiza mediante un algoritmo evolutivo, en concreto mediante un algoritmo genético como motor de búsqueda basado en el paradigma de evolución gramatical.

El ajuste de los parámetros del algoritmo genético y la construcción de las funciones evolutivas se ha llevado a cabo trabajando con el algoritmo de clasificación CART. Este trabajo se ha desarrollado en los Experimentos 1 al 6.

En los Experimentos 7 y 8 se ha probado, partiendo de los resultados obtenidos en el Experimento 6 y usando los mismo parámetros de configuración, el comportamiento de la metodología con dos clasificadores nuevos, a saber: k-vecinos y regresión logística.

Los resultados obtenidos con el clasificador CART han sido muy satisfactorios. Se ha obtenido un porcentaje de conjuntos de datos en los que la metodología mejora de forma significativa los resultados obtenidos, frente al conjunto de variables iniciales, del 86.66 % lo que consideramos un éxito. Además la mejora media ha sido de casi 3 puntos porcentuales del F1-Score sobre el conjunto de variables iniciales $F1(F_A)$, frente al F1-Score sobre el conjunto de variables ampliado $F1(F_0)$.

Los resultados obtenidos con los otros dos clasificadores usados, k-vecinos y regresión logística, han sido más irregulares. Si bien la regresión logística ha mostrado un comportamiento moderado pero interesante, llegando a casi el 50 % de conjuntos en los que hay mejora y un incremento de 1.3 puntos porcentuales, con k-vecinos los resultados obtenidos han sido malos. No se han realizado

todas las ejecuciones dado que de los seis conjuntos de datos empleados, para diferentes valores del parámetro de k (número de vecinos), solo en uno de los conjuntos se ha obtenido mejoría para cada k probado.

Dado que de los tres clasificadores evaluados en uno se han obtenido valores muy significativos, en otro moderados y en el tercero malos, no podemos garantizar que la metodología propuesta pueda ser generalizable de forma directa a cualquier tipo de clasificador. Entendemos que hay tres puntos que lo dificultan:

- **Ajuste fino de parámetros:** Hemos trabajado tanto para el ajuste fino como para las metodologías de cruce y mutación con el algoritmo de clasificación CART. Esto puede haber supeditado en cierta medida los valores obtenidos y podría darse el caso de que los mejores valores para CART no lo sean para otro clasificador.

- **Tratamiento de variables:** Los árboles de decisión tienen capacidad para trabajar con todo tipo de variables con una necesidad mínima de pre-procesado, lo que les hace ideales para la metodología propuesta. Otros clasificadores necesitan un procesado previo de las variables para garantizar que cumplen determinadas cualidades, sin ellas el rendimiento del clasificador se ve muy mermado. Dado que en el proceso de construcción de las nuevas variables se emplea, como función de evaluación el resultado del clasificador sobre la muestra de validación, el que haya distorsiones en las variable de entrada o algún tipo de anomalía perjudica el proceso de aprendizaje y por tanto los resultados obtenidos.

- **Parámetros del clasificador:** Como ocurre en el árbol de decisión, con el uso de la librería rpart y su parámetro "cp", o en el del número de vecinos, k, en el algoritmo de los k-vecinos, los clasificadores pueden presentar sus propios parámetros de configuración que también deben ser optimizados y que pueden depender del conjunto de datos con el que se esté trabajando.

Parece muy complicado llegar a identificar un conjunto de parámetros que sean universales para todos los conjuntos de datos que se puedan abordar, tanto en la configuración del algoritmo evolutivo como en los parámetros del propio clasificador. No obstante hemos identificado de forma clara el potencial de la metodología propuesta y consideramos que es muy interesante a la hora de abordar problemas de clasificación debido a:

- La creación de nuevas variables derivadas, relacionadas con la variable objetivo puede permitir la obtención de un mayor conocimiento del problema y mejorar su explicabilidad.

- La introducción de las nuevas variables derivadas puede aportar una mejora en los resultados obtenidos por el clasificador.

De los puntos que nos hemos fijado inicialmente como desarrollables hemos conseguido abordar en el presente trabajo los siguientes:

- La metodología propuesta se ha presentado de forma transparente y aporta un sistema de medición y evaluación de resultados respaldado por un test estadístico.
- Hemos probado la metodología con tres tipos de clasificadores, a saber: árbol de decisión, k-vecinos y regresión logística.
- Hemos probado diferentes operadores en la gramática. Si bien los resultados no han mostrado una mejoría relevante al añadir operadores más complejos a los probados inicialmente se abre una vía de trabajo que pensamos es de interés.
- Se ha realizado una evaluación del rendimiento de los clasificadores empleando el conjunto inicial de variables F_0 frente al conjunto ampliado F_A.

La metodología propuesta ha demostrado ser interesante a la hora de mejorar el rendimiento de los algoritmos de clasificación, si bien necesita un ajuste fino para cada tipo de clasificador y posiblemente para cada conjunto de datos, no pudiéndose garantizar que funcione de forma satisfactoria en todos los casos.

Capítulo 6

Trabajos futuros

Como posible evolución de la metodología propuesta se podría trabajar en:

- Construir una muestra de entrenamiento y validación dinámica. En cada iteración del algoritmo genético, siendo los mismo registros totales, se seleccionarían diferentes muestras de entrenamiento y validación. De esta forma el mejor individuo debería serlo en sucesivas iteraciones, siendo validado con diferentes muestras. La muestra de test no se usaría en ningún momento, hasta la evaluación final de los dos bloques de variables F_0 y F_A.

- Desligar el proceso de la necesidad del clasificador en la fase de construcción de variables. Con el planteamiento actual la función de evaluación de cada individuo se mide en función del F1-score obtenido empleado el clasificador sobre el conjunto de variables ampliado en la muestra de validación. La idea sería que el análisis de la función de evaluación de cada individuo se hiciera teniendo en cuenta la calidad de las nuevas variables generadas independientemente del clasificador que se vaya a emplear. Se podrían usar métricas para medir la "calidad" de las variables como el Gain Ratio, índice de Gini... La elección de la métrica podrá repercutir en los resultados obtenidos con distintos clasificadores, por ejemplo el Gain Ratio lo emplea el C4.5 y C5 a la hora de seleccionar las variables que van a formar parte del árbol de decisión.

- En una línea similar al punto anterior se podría probar con el uso del Run test como evaluador. En el desarrollo del TFM se han realizado pruebas y ofrece buenos resultados. El problema es que necesita variables continuas y con cardinalidad alta para que la ordenación y medición de las rachas tenga validez.

Otra campo de investigación que emana del que hemos abordado y donde podríamos trasladar las conclusiones obtenidas en el presente estudio es el de la sumarización de variables, que abre una vía de investigación nueva y a la vez complementaria.

El número de trabajos sobre sumarización de datos es realmente escaso, más si se aborda desde la perspectiva de la Computación Evolutiva. Este hecho es curioso dado que en la práctica es un tema de gran interés debido a la proliferación, en las últimas décadas, de los repositorios de información empresariales. Los Data Warehouses y Data Marts, cuentan con datos históricos para las variables, situación que provoca que previo a la aplicación de técnicas de modelado sea necesario un proceso previo de sumarización.

Los estudios encontrados sobre técnicas de sumarización de datos mediante técnicas evolutivas se centran en el algoritmo DARA [38] que trabaja sobre datos sin variable objetivo. Por tanto entendemos que existe un amplio camino de análisis y experimentación en:

- Sumarización de datos mediante evolución gramatical, aplicada sobre problemas con variable objetivo.
- Sumarización de variables tanto continuas como discretas.
- Aplicación sobre problemas reales de negocio.

También se podría abordar de forma simultánea el problema de sumarización y creación de variables. Generando variables nuevas partiendo de un conjunto de datos historificado.

Apéndice A

Árbol decisión CART

En este apartado vamos a analizar el algoritmo de árbol de decisión empleado y la parametrización del mismo.

En la realización de los experimentos se ha empleado el software estadístico R [50], que ofrece diferentes librerías con algoritmos de árboles de decisión. En nuestro caso hemos empleado el paquete rpart [49], con el que estamos habituados a trabajar, y por la facilidad que ofrece a la hora de realizar la poda óptima.

La librería rpart esta basada en los árboles tipo CART propuestos por Breiman, Friedman, Olshen y Stone en el trabajo Classification and Regression Trees [28], de 1984.

Como características principales de este algoritmo podemos señalar que las particiones de cada rama del árbol son binarias y que la métrica de corte, por defecto, es el índice de Gini. La implementación de rpart facilita el parámetro "cp" que indica el nivel de ajuste global del árbol obtenido en el entrenamiento. Es de gran importancia porque esto nos posibilita expandir el árbol de forma completa introduciendo un "cp" muy bajo, para después analizar el corte óptimo. Una vez que hemos determinado el "cp" óptimo podemos procede a realizar una poda.

El empleo del parámetro cp nos permite garantizar que el proceso de evaluación de las nuevas variables obtenidas es riguroso y que el árbol con las variables originales y el ampliado es exhaustivo, sin llegar al sobre-aprendizaje. Este punto es muy relevante dado que podríamos haber sido conservadores indicando un número de hijos por hoja alto evitando el crecimiento excesivo del árbol. Evidentemente esto hubiera penalizado la performance del modelo y hubiera facilitado la aparición de diferencias significativas entre el modelo entrenado con F_0 y el modelo entrenado con el conjunto de variables ampliado F_A. Al emplear este técnica garantizamos que el algoritmo está trabajando al

máximo y que por tanto las diferencias, en caso de que existan, se deben a la mejora en calidad de las nuevas variables y no al aprovechamiento de una menor prospección del espacio de búsqueda.

Para mostrar un ejemplo de como trabajamos con el parámetro cp se ha realizado una ejecución con el conjunto de datos QSAR. Se ha empleado una de las ejecuciones del Experimento 1, obteniendo las nuevas variables ampliadas y aplicando la metodología general. Para ello se construyen los dos árboles de decisión con el conjunto de variables iniciales F_0 y el ampliado F_A. Se calcula el "c"p óptimo y se podan los árboles en ese punto.

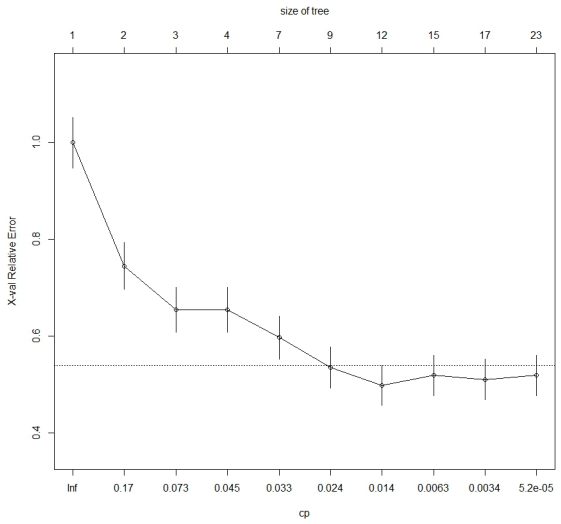

Figura A.1: Error vs cp en árbol con F_0

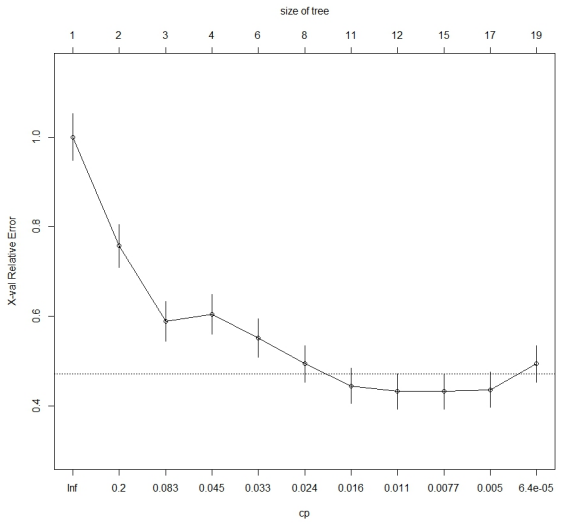

Figura A.2: Error vs cp en árbol con F_A

En las figuras A.1 y A.2, mostramos la evolución del error en función de la complejidad del árbol, esto es el tamaño del mismo, medida a través del parámetro "cp". Podemos ver que tanto en el árbol construido con el conjunto de variables iniciales A.1, como el construido con el conjunto de variables ampliado A.2, se alcanza el mínimo antes de llegar al valor de cp fijado por defecto (cp=0.000001). Esto nos indica que a partir de dicho punto mínimo se está incrementado el error y por tanto el algoritmo está sobre-aprendiendo. Nuestro objetivo, para optimizar los resultados del clasificador, será podar el árbol en el punto de mínimo error, eliminando el resto de estructura que provoca el sobre-aprendizaje.

En las figuras A.3 y A.3 se muestra el resultado de podar el árbol obtenido con el conjunto de variables F_0 en función del "cp" óptimo. Podemos ver como la rama inferior derecha del árbol se ha eliminado.

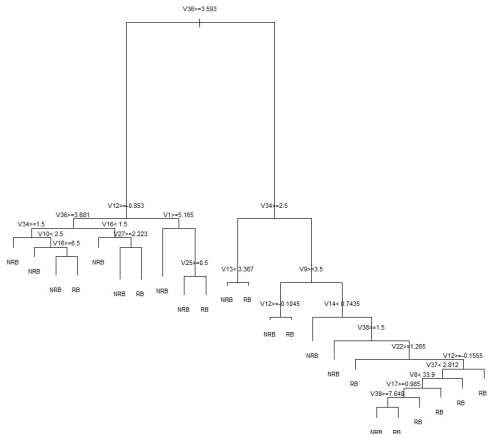

Figura A.3: Árbol con F0

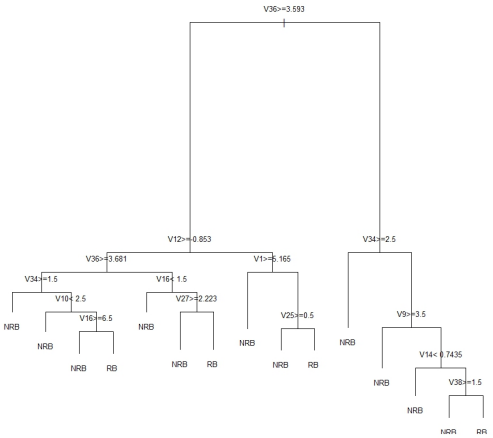

Figura A.4: Árbol con F0 podado

Apéndice B

Regresión logística

La regresión logística [24] es uno de los métodos clásicos más empleados en el ámbito de la asignación de clases. Este clasificador permite trabajar con variables objetivo binarias del tipo $y_i \in \{0, 1\}$, devolviendo una probabilidad de que cada registro pertenezca a una de las clases. Estas características y su facilidad de interpretación, dado que es conceptualmente similar a una regresión lineal han contribuido a la generalización de su uso en múltiples campos.

Los problemas de clasificación abordados por la regresión logística pueden presentar variables objetivo binarias, con dos categorías, o múltiples, con más de dos categorías. En esta aproximación superficial nos centraremos en el caso binario, más delante comentaremos como se ha abordado el caso de variables objetivo con más de dos categorías, como ocurre en los ficheros: glass, forest_type, ecoli, statlog y drive_diag.

La variable dependiente puede tomar dos valores con probabilidad:

$$y_i = \begin{cases} 1 & P(1) = \quad p_i \\ 0 & P(0) = \quad 1 - p_i \end{cases}$$

Aplicamos una transformación logística sobre p_i:

$$p_i = \frac{1}{1 + e^{-\beta' x_i}}$$

Esta relación se puede escribir como:

$$log(\frac{p_i}{1 - p_i}) = \beta' x_i = \beta_0 + \beta_1 x_{il} + ... + \beta_p x_{ip}$$

Donde el primer término corresponde al logit, igual al predictor lineal clásico.

El modelado automático de un clasificador logístico requiere de un trabajo previo con los datos, en concreto la metodología implementada incluye:

- **Escalado:** Procedemos a realizar un re-escalado de variables originales y de las nuevas en el intervalo [0,1]. Dependiendo del método empleado puede acelerar la convergencia del algoritmo.
- **Valores infinitos y NA:** Si las nuevas variables generadas introducen valores infinitos o NA se realiza una imputación de esas observaciones. En el caso de los valores infinitos sustituimos por el máximo del resto de valores cuando la variable toma $+\infty$ y por el mínimo en el caso de $-\infty$. En el caso de valores NA los sustituimos por la media de la variable.
- **Desviación típica cero:** eliminamos estas variables dado que no tiene información y por tanto no aportan valor al clasificador.
- **Multicolinealidad:** Eliminamos las variables que tienen una alta correlación de Pearson [20], en concreto más de 0.8, de esta forma evitamos el deterioro del clasificador.

Además, dado que los clasificadores logísticos son particularmente sensibles a la distribución de las clases se ha procedido a balancear la muestra de forma que todas las categorías estén igualmente representadas. Se ha empleado un método de adicción, de forma que disponemos de tantas observaciones por clase como las que tiene la clase mayoritaria antes del balanceo en la muestra de entrenamiento.

En R [50] hay diferentes paquetes que permiten obtener un clasificador basado en regresión logística, no obstante, que se adapten bien al problema propuesto presenta ciertas dificultades. Los detallamos a continuación porque ha sido relevante a la hora de implementar el experimento:

- **glm:** Es la función habitual para el ajuste de modelos lineales y de modelos logísticos. Presenta el inconveniente de trabajar solo con variables objetivo binarias, lo que para nosotros es un problema dado que tenemos los conjuntos glass, forest_type, ecoli, statlog y drive_diag con clases que no son binarias. La solución más simple es realizar l modelos, uno para cada una de las categorías de la clase. Transformando un problema de clasificación múltiple en l problemas de clasificación binaria. Obtendríamos, por tanto, l vectores de probabilidad y seleccionaríamos la clase con mayor probabilidad para cada observación. Esto implica un desarrollo en R, que hemos realizado y evaluado.
- **mlogit [10] y mnlogit [22]:** Estos dos paquetes requieren una adecuación previa de los datos que presentaba un fuerte impacto en el código ya desarrollado y por tanto se han descartado.

- **vgam [52]:** Este paquete ofrece la posibilidad de trabajar con variables dependientes no binarias y ha sido testado. Ofrece problemas con conjuntos de variables que presentan multicolinealidad. El tiempo de ejecución es muy elevado.

- **nnet [43]:** Este paquete de redes neuronales ofrece la posibilidad de ajustar un modelo logístico empleando la función "multinom".

Después de analizar los paquetes, realizando la implementación y evaluando de resultados y tiempos de ejecución, hemos decidido emplear el "nnet", en concreto la función "multinom". Los tiempos de ejecución que ofrece son muy buenos así como su estabilidad. Hemos mantenido el número máximo de iteraciones en 100 en la fase evolutiva, pasando a 1.000 iteraciones de la red en los modelos finales de evaluación.

La estructura de red neuronal para implementar una regresión logística cuenta con una capada de entrada, con una neurona por cada una de las variables introducidas más otra para el término independiente y una capa de salida con tantas neuronas como clases tenga la variable objetivo. En el caso logístico, que es muy básico, no contamos con capa oculta. Para terminar de implementar el modelo logístico hay que emplear como función de activación la sigmoide definida por:

$$P(t) = \frac{1}{1 + e^{-t}}$$

Apéndice C

k-vecinos

El clasificador de k-vecinos [24] destaca por su simplicidad y en muchos casos eficiencia. El método consiste en asignar la clase a cada registro de la muestra de test que corresponde a la clase mayoritaria de los k-vecinos más próximos de la muestra de train. La proximidad se mide a través de una distancie euclídea, de forma usual.

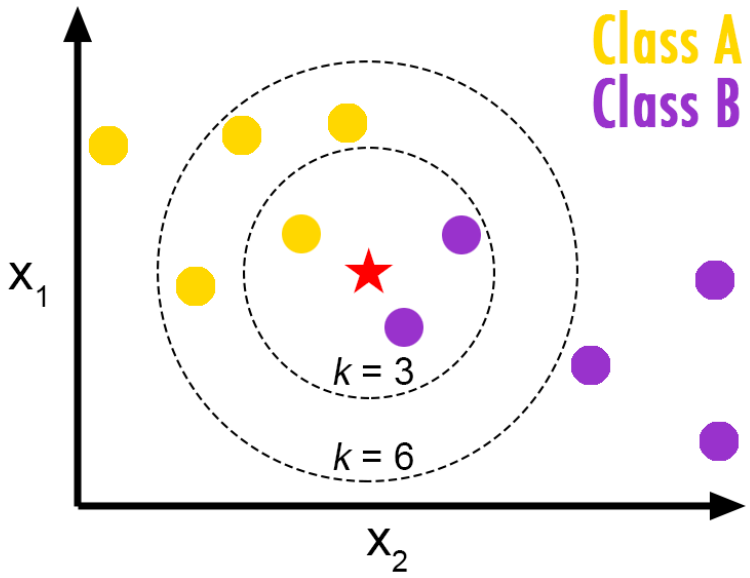

Figura C.1: Esquema de los k-vecinos

En la figura C.1 se muestra un ejemplo de la forma de trabajar del algoritmo. Los puntos de entrenamiento se representan por círculos, mientras que el de test es una estrella. Para k=3 tenemos un primer circulo interior, con dos observaciones de la clase B y solo una de la clase A, por tanto

asignaríamos la clase B al caso de train. Si cambiamos el k y pasamos a seis vemos como la clase asignada al caso de test cambiaría, dado que ahora contamos con cuatro observaciones de la clase A y solo 2 de la clase B, por tanto asignaríamos la clase A. En esta figura C.1 se aprecia gráficamente como el valor de k empleado es determinante a la hora de usar el algoritmo de los k-vecinos.

Este algoritmo se deriva del 1-NN o vecino más próximo que solo tiene en cuenta un único vecino. La ventaja que aporta el K-NN es que al emplear $k > 1$ se disminuye el sobreajuste.

El inconveniente de este tipo de algoritmo es la selección del valor de k óptimo, puesto que un valor demasiado grande hará que perdamos eficacia en la clasificación al ser demasiado generalistas. Un valor demasiado pequeño de k hará que no seamos capaces de generalizar bien y sobreajustemos la muestra de train, perdiendo performance en test.

Para la implementación en R [50] del algoritmo de k-vecinos se han evaluado las librerías:

- **Class (Functions for Classification):** Cuenta con el método knn [42], la performance en cuanto a tiempo de ejecución que ofrece es buena, no obstante han surgido problemas que no se han podido resolver a la hora de paralelizar el procesamiento con el paquete doParallel [9] , por este motivo se ha descartado su uso.

- **FNN (Fast Nearest Neighbor Search Algorithms and Applications):** Cuenta con el método knn, presenta una performance similar al paquete class [42], en esta ocasión la paralelización, como en el resto de implementaciones no ha dado problemas.

- **kknn (Weighted k-Nearest Neighbors for Classification, Regression and Clustering):** Este paquete cuenta kknn [46], se ha descartado su uso dado que los tiempos de ejecución eran superiores a los obtenidos con el paquete FNN [5].

Hay que destacar que antes de usar el algoritmo k-vecinos en R [50] se ha realizado una estandarización previa, el objeto es que todas las variables tengan el mismo peso a la hora de calcular la distancia. En concreto hemos empleado una transformación tipo max-min [24].

Apéndice D

Contraste de medias

El contraste para comparar las medias de las n ejecuciones realizadas con cada conjunto de datos vendría dado por:

$$H_0 : \mu_1 = \mu_2$$

$$H_1 : \mu_1 \neq \mu_2$$

No obstante en este caso, debido al diseño de los experimentos que hemos realizado, como método de validación estadística de los resultados obtenidos, hemos empleado una comparación por pares. Dado que realizamos n ejecuciones, en las que generamos tres muestra aleatoria del conjunto de datos en cada una, y devolvemos los resultados del F1-Score para el clasificador ejecutado sobre el conjunto de variables F_0 y el F_A, podemos calcular la diferencia que hay entre ambos valores, para cada una de las ejecuciones.

Este método se ajusta mejor a la metodología empleada puesto que la elección de las muestras de entrenamiento, validación y test puede afectar a los resultados del mismo.

Esta forma de evaluar la diferencia de dos medias, frente a la tradicional comparación, reduce la variabilidad y por tanto tiene una mayor verosimilitud a la hora de determinar si una diferencia es o no significativa.

Para las n ejecuciones independientes tendremos que la j-ésima diferencia por pares viene dada por:

$$d_j = y_{1j} - y_{2j}$$

$$j = 1, 2, ..., n$$

Pasaremos a realizar el test sobre las diferencias

$$\mu_d = \mu_1 - \mu_2$$

$$H_0 : \mu_d = 0$$

$$H_1 : \mu_d \neq 0$$

El estadístico empleado para validar esta hipótesis es

$$t_0 = \frac{\bar{d}}{S_d/\sqrt{n}}$$

donde

$$\bar{d} = \frac{1}{n}\sum_{j=1}^{n} d_j$$

es la media muestral de las diferencias y

$$S_d = \left[\frac{\sum_{j=1}^{n}(d_j - \bar{d})^2}{n-1}\right]^{\frac{1}{2}}$$

es la desviación estándar muestral de las diferencias. Por tanto $H : \mu_d = 0$ debe ser rechazada, o lo que es lo mismo no hay diferencia entre emplear el conjunto inicial de variables F_0 y el ampliado F_A, si $|t_0| > t_{\alpha/2,n-1}$. Donde $1-\alpha$ es el nivel de confianza y $n-1$ son los grados de libertad de la distribución t.

La explicación completa de este método se puede encontrar en el libro de Montgomery: Diseño y Análisis de Experimentos [7].

Apéndice E

F1-score

En el desarrollo de los experimentos se empleará la medida F-Score como función de evaluación del algoritmo evolutivo en cada iteración.

La elección de esta medida de calidad del modelo de clasificación como función de evaluación se basa en su capacidad de dar una evaluación del ajuste del modelo ajeno al balanceo de los datos. Esta medida es igualmente fiable en conjuntos balanceados y no balanceados y por tanto nos brinda un arco estable sobre el que trabajar.

La métrica F-Score se define por:

$$Fscore_{\mu} = \frac{(\beta^2 + 1)P_{\mu}.R_{\mu}}{\beta^2.P_{\mu}.R_{\mu}}$$

donde la "Precisión"la calculamos como

$$P_{\mu} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i + fp_i)}$$

y el "Recall" como:

$$R_{\mu} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i + fn_i)}$$

En nuestro caso daremos el mismo peso a la precisión y al recall, empleando el coeficiente $\beta = 1$, obteniendo la métrica F1-Score.

Como referencia para la métrica F-Score se puede consultar el estudio general sobre medidas de performance en modelos de clasificación de Sokolova [47].

Bibliografía

[1] Uci machine learning repository. `https://archive.ics.uci.edu/ml/datasets.html`.

[2] Weight of Evidence Module. *STATISTICA Formula Guide*, 2013.

[3] Arauzo-Azofra A, Aznarte J L, and Benítez J M. Empirical study of feature selection methods based on individual... *Expert Systems with Applications 38*, page 8170–8177, 2011.

[4] R.J. Anderson. *Pattern recognition and machine learning.* Springer, 2006.

[5] A. Beygelzimer, S. Kakadet, and J. Langford. Package fnn versión 1.1. `https://cran.r-project.org/web/packages/FNN/FNN.pdf`, 2015.

[6] Krier C, François D, Rossi F, and Verleysen M. Feature clustering and mutual information for the selection of variables in spectral data . *European Symposium on Artificial Neural Networks*, pages 157–162, Abril 2007.

[7] Montgomery D C. *Diseño y Análisis de Experimentos.* Grupo Editorial Iberoamérica, 1991.

[8] Ryan C, Collins JJ, and O'Neill Michael. Grammatical Evolution: Evolving Programs for an Arbitrary Language. 1998.

[9] R. Calaway, Revolution Analytics, S. Weston, and D. Tenenbaum. Package doparallel versión 1.0.10. `https://cran.r-project.org/web/packages/doParallel/doParallel.pdf`, 2015.

[10] Y. Croissant. Package mlogit 0.2-4. `https://cran.r-project.org/web/packages/mlogit/mlogit.pdf`, 2013.

[11] Gavrilis D, Tsoulos I G, and Dermatas E. Selecting and constructing features using grammatical evolution. *Pattern recognition letters*, 2008.

[12] Gavrilis D, Tsoulos I G, Georgoulas G, Stylios C, Bernardes J, and Groumpos J. Introducing Grammatical Evolution for FHR analysis and classification.

[13] Gavrilis D, Tsoulos I G, Georgoulas G, and Glavas E. Classification of Fetal Heart Rate using Grammatical Evolution.

[14] Eiben A E and Smith J E. *Introduction to Evolutionary Computing.* Springer, 2007.

[15] Otero F, Siva M, Freitas A, and Nievola J. Genetic Programming for Atribute Construction in Data Mining. 2003.

[16] Chandrashekar G and Sahin F. A survey on feature selection methods. *Computers and Electrical Engineering 40*, pages 16–28, 2014.

[17] Forman G. Feature Selection for Text Classification. *Information Services and Process Innovation Laboratory*, Mayo 2007.

[18] John G, Kohavi F, and Ptleger K. Irrelevant features and the subset selection problem. *International Conference on Machine Learning*, pages 121–129, 1994.

[19] Pagallo G. Learning DNF by decision trees. *Proceeding of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[20] DeGroot M. H. *Probabilidad y Estadística*. Addison Wesley Iberoamericana, 1988.

[21] Holland J. H. *Adaptation in natural and artificial systems*. MIT Press, 1993.

[22] A. Hasan, W. Zhiyu, and A. S. Mahani. Package mlogit 1.2.5. `https://cran.r-project.org/web/packages/mnlogit/mnlogit.pdf`, 2016.

[23] Guyon I and Elisseeff A. An introduction to variable and feature selection. *Journal of Machine Learning Research 3*, pages 1157–1182, Marzo 2003.

[24] Hernández J., Ramírez M. J., and Ferri C. *Introducción a la Minería de Datos*. Prentice Hall, 2008.

[25] Hu Y J. *Constructive Induction: Covering Attribute Spectrum*. Kluwer, 1998.

[26] Hu Y J and Kibler D. Generation of Attributes for Learning Algorithms. *National Conference on Artificial Intelligence*, page 806–811, 1996.

[27] Krawiec K. Genetic Programming-based Construction of Features for Machine Learning and Knowledge Discovery Tasks. *Genetic Programming and Evolvable Machines*, 2002.

[28] Breiman L, Friedman J, Stone J, and Olshen R A. *Classfication and regressions trees*. Taylor & Francis, 1984.

[29] Cuadras C M. *Nuevos Métodos de Análisis Multivariante*. CMC Editions, 2014.

[30] Smith M and Bull L. Genetic Programming with a Genetic Algorithm for Feature Construction and Selection. 2005.

[31] S. Markovitch and D. Rosenstein. Feature Generation Using General Constructor Functions. *Machine Learning, 49*, 2002.

[32] Lavrač N., Džeroski S., and Grobelnik M. Learning nonrecursive definitions of relations with LINUS. *Machine Learning—EWSL-91*, pages 265–281, 1991.

[33] Vapnik V N. *Statistical Learning Theory*. Wiley, 1995.

[34] F. Noorian and A. Mihirana de Silva. Package gramevol versión 2.0-2. `https://cran.r-project.org/web/packages/gramEvol/index.html`, 2015.

[35] Box G E P and Cox D R. An Analysis of Transformations. *Research Methods Meeting of the Society*, pages 211–252, Abril 1964.

[36] Espejo P, Romero C, Hervás C, and Ventura S. Programación genética gramatical para el descubrimiento de reglas de clasificación. 2003.

[37] Sondhi P. Feature Construction Methods: A Survey. 2010.

[38] Alfred R. Dara: Data summarisation with feature construction. In IEE computer society, editor, *The title of the book*, pages 830–835. Second Asia International Conference on Modelling & Simulation, 2008.

[39] Kohavi R and George H J. Wrappers for feature subset selection. *Artificial Intelligence 97*, pages 273–324, 1997.

[40] Koza J R. *Genetic Programming*. MIT Press, 1994.

[41] Quinlan J R. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.

[42] B. Ripley and W. Venables. Package class versión 7.3-14. https://cran.r-project.org/web/packages/class/class.pdf, 2015.

[43] B. Ripley and W. Venables. Package nnet 7.3-12. https://cran.r-project.org/web/packages/nnet/nnet.pdf, 2016.

[44] D. Roth and K. Small. Interactive feature space construction using semantic information. *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, 2009.

[45] Goswami S. Feature Selection : A Practitioner View. *Information Technology and Computer Science*, Octubre 2014.

[46] K. Schliep, K. Hechenbichler, and A. Lizee. Package kknn versión 1.3.1. https://cran.r-project.org/web/packages/kknn/kknn.pdf, 2016.

[47] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, (45):427–437, 2009.

[48] Bäck T, Fogel D B, and Michalewicz T. *Evolutionary Computation*. IoP, 2000.

[49] T. Therneau, B. Atkinson, and B. Ripley. Package rpart versión 4.1-10. https://cran.r-project.org/web/packages/rpart/index.html, 2015.

[50] V.A. The r project for statistical computing version 3.1.0. http://www.r-project.org/, 2014.

[51] V.A. R studio versión 0.98.945. http://www.rstudio.com/, 2014.

[52] T. W. Yee. Package vgam 1.0-4. https://cran.r-project.org/web/packages/VGAM/VGAM.pdf, 2017.