Algoritmo genético para selección de atributos en modelos predictivos

Inteligencia Artificial (IS) 2021/22 – Propuesta de trabajo

Juan Galán Páez

1. Introducción

El **objetivo** de este trabajo consistirá en realizar una implementación del algoritmo genético simple que permita seleccionar el subconjunto de variables óptimo (aquel con mayor poder predictivo) para entrenar modelos predictivos sobre un conjunto de datos. Llamamos conjunto óptimo, a aquel subconjunto de variables (del conjunto de datos sobre el que se esté trabajando) que permita entrenar un modelo predictivo con el mayor rendimiento posible. Una vez implementado el algoritmo base, se propondrán mejoras. Finalmente, se proporcionan varios conjuntos de datos sobre los que experimentar con la implementación realizada y extraer conclusiones.

A continuación se introducen diferentes conceptos necesarios para contextualizar este trabajo.

1.1. *Selección de atributos y aprendizaje automático*

Una de las ramas más importante del aprendizaje automático (*machine learning*) es el aprendizaje supervisado, que proporciona algoritmos para el entrenamiento de modelos predictivos. El aprendizaje supervisado parte de un conjunto de entrenamiento en el que distinguimos una serie de variables predictoras (también denominadas atributos o características) y una variable respuesta. Esta variable respuesta es el objetivo del aprendizaje. El algoritmo de aprendizaje será capaz de encontrar patrones en las variables predictoras que le permitan, dada una nueva instancia de datos para la que estos valores son conocidos, estimar (predecir) el valor de la variable respuesta más adecuado.

El presente trabajo se centra en la selección atributos[1], es decir, la obtención de un subconjunto de variables predictoras que aporten la mayor cantidad de información posible sobre la variable respuesta. En otras palabras, el objetivo es quedarnos con las variables que más información aportarán al algoritmo de entrenamiento durante el proceso de aprendizaje y descartar aquellas que aporten menos información. ¿Por qué queremos seleccionar variables en vez de usarlas todas? ¿Por qué queremos descartar variables que aportan poca información sobre la variable respuesta, si aportan algo? Existen varias razones por las que la selección de atributos es importante, aunque todas ellas están relacionadas con la complejidad del modelo predictivo resultante. Uno de los factores que más afecta (aunque no el único) a la complejidad de del modelo predictivo obtenido es el número de variables predictoras con que ha sido entrenado:

- **Recursos computacionales:** El tiempo de entrenamiento y recursos necesitados (procesador, memoria RAM, etc.) serán menores cuanto menos variables tenga nuestro conjunto de datos.
- **Interpretabilidad:** Cuanto más sencillo sea el modelo predictivo obtenido, más fácil será interpretar las decisiones (predicciones) que este produce.

1 https://en.wikipedia.org/wiki/Feature_selection

- **Sobreajuste y capacidad predictiva:** Cuanto más variables (y menos informativas) se usen en el proceso de aprendizaje, tendremos más riesgo de que el modelo aprenda patrones falsos (ruido), es decir, patrones que existen en el conjunto de entrenamiento, pero no en la realidad. Un conjunto de variables reducido, pero muy informativas permitirá al algoritmo encontrar pocos patrones pero muy valiosos a la hora de predecir la variable respuesta.

La selección de atributos es uno de los problemas bajo estudio en la actualidad (y desde hace décadas) más importante dentro del aprendizaje automático. Hoy en día existen numerosas soluciones a este problema, sin embargo, no existe ninguna solución universal. Cada método tiene sus ventajas e inconvenientes y funciona bien sobre ciertos tipos de problemas y algoritmos y mal sobre otros.

1.2. *Selección de atributos como problema de optimización*

El problema de la selección de atributos es abordado hoy en día desde diversos enfoques, uno de ellos es el de los problemas de optimización. Un problema de optimización[2] es aquel en el que no es difícil o costoso encontrar una solución (o uno para el que directamente se conocen todas las posibles soluciones) y en el que el objetivo no es encontrar una solución válida sino la mejor solución. El concepto de mejor solución dependerá del problema a resolver en cuestión (menor error, máximo beneficio, menor tiempo, etc.). Algunos ejemplos de problemas de optimización pueden ser:

- Obtener el conjunto de parámetros para los que una función matemática alcanza el menor (o mayor) valor posible.
- Una empresa de logística desea realizar una asignación de repartidores, vehículos y paquetes, óptima. Según el caso, esto podría significar repartir todos los paquetes en el menor tiempo posible o repartir todos los paquetes en un tiempo dado, usando el menor número de repartidores y vehículos posibles.
- En general, cualquier problema relacionado con la planificación de tareas y la asignación de recursos.
- Optimización de una cartera de activos financieros. Generar un conjunto de activos financieros sobre los que invertir, de forma que desde un punto de vista global (es decir, a nivel de cartera) el riesgo sea el menor posible y el beneficio el máximo.

Los problemas de optimización se dividen en problemas de maximización y minimización según la dirección de la optimización. Adicionalmente, se denomina optimización multi-objetivo cuando se intentan optimizar varios objetivos simultáneamente, por ejemplo el tiempo y los recursos empleados en un problema de logística.

Una vez introducido el concepto general de problema de optimización, veamos como quedaría enunciado el problema de la selección de atributos en este contexto: ***Obtener aquel subconjunto de atributos para el que un modelo predictivo entrenado usando dicho subconjunto, proporciona el mejor rendimiento posible sobre el conjunto de prueba***. Si medimos el rendimiento en forma de tasa de aciertos, estaríamos ante un problema de maximización, mientras que sería un problema de minimización si lo medimos como tasa de error.

2 https://es.wikipedia.org/wiki/Optimizaci%C3%B3n_(matem%C3%A1tica)

Por último, es importante mencionar que el mejor subconjunto de atributos no tiene por qué ser aquel que mejor rendimiento proporcione ya que buscamos un compromiso entre número de atributos (el menor posible) y rendimiento (el mayor posible). Por ejemplo, supongamos que tenemos un conjunto inicial de 50 atributos y tras resolver el problema de optimización, obtenemos las siguientes tres mejores soluciones:

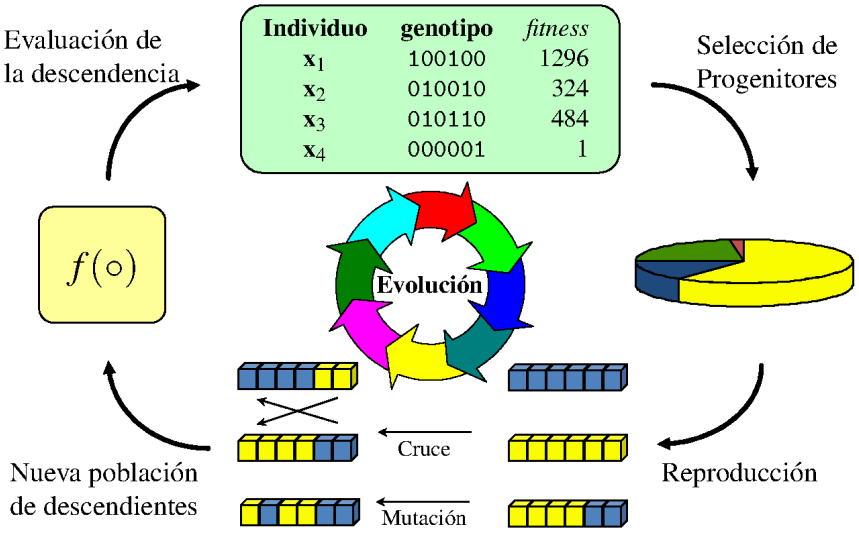
Conjunto	N.º variables	Tasa de aciertos
Conjunto A	45	95,3%
Conjunto B	25	94,5%
Conjunto C	10	89,1%

El *conjunto A* es el que mayor tasa de aciertos proporciona, mientras que el *conjunto C* es el que menor número de variables tiene. Sin embargo, el conjunto que proporciona un mejor compromiso entre número de atributos y tasa de aciertos es el *conjunto B*.

Teniendo en cuenta, esto, podríamos enunciar el problema como uno de optimización multi-objetivo, en el que buscamos el menor subconjunto de atributos con el mayor rendimiento. El abordaje de problemas de optimización multi-objetivo es más complejo, por lo que empezaremos por el primer enunciado propuesto.

Existe un gran número de técnicas para resolver problemas de optimización, entre las que destacan los algoritmos genéticos.

1.3. Algoritmos genéticos



Los algoritmos genéticos se refieren a una familia de algoritmos de optimización basados en población. Los algoritmos basados en población son aquellos en los que en cada iteración no se procesa una sola solución candidata, sino un conjunto de soluciones (población). De forma intuitiva, estos algoritmos implementan un proceso iterativo en el que en cada iteración se va evolucionando un conjunto de soluciones candidatas, de forma que se van descartando las peores y se van

generando nuevas soluciones realizando pequeñas modificaciones sobre las mejores soluciones encontradas hasta el momento. Nótese que en el contexto de este trabajo una solución candidata es un subconjunto de atributos.

Artículo de referencia: Para que los alumnos se familiaricen con los algoritmos genéticos se proporciona la siguiente referencia:
http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf

Las secciones 1 y 2 del documento introducen el algoritmo genético simple (véase el pseudocódigo de la página 2) que es en el que se centrará este trabajo. Adicionalmente, se recomienda la lectura de las primeras 13 páginas, es decir, hasta el apartado 3.6, incluido. Esto proporcionará una mayor comprensión de los diferentes conceptos involucrados en este tipo de algoritmos.

1.4. ¿Por qué necesitamos aplicar algoritmos genéticos en la selección de atributos?

Los algoritmos genéticos representan un método que permite analizar de forma eficiente el espacio de soluciones (en nuestro caso, todos los subconjuntos de atributos que se pueden extraer del conjunto de datos). Para esto, este tipo de algoritmos son capaces de alcanzar un compromiso adecuado entre exploración (buscar nuevas soluciones prometedoras) y explotación (mejorar una solución prometedora ya encontrada). En otras palabras, intentan encontrar la mejor solución, explorando el menor número de soluciones posible. Nótese que los algoritmos genéticos no garantizan la solución óptima, sino que proporcionan una lo suficientemente buena. Para garantizar la obtención de una solución óptima, necesitaríamos realizar una exploración exhaustiva, es decir, evaluar todas las soluciones posibles y quedarnos con la mejor.

La búsqueda exhaustiva no es viable:
El enfoque más sencillo e intuitivo, sería generar y evaluar todos los posibles subconjuntos de atributos.

Por ejemplo, sea un conjunto de 3 variables, el conjunto de todos los subconjuntos posibles sería el siguiente:
('A', 'B', 'C') → [('A',), ('B',), ('C',), ('A', 'B'), ('A', 'C'), ('B', 'C'), ('A', 'B', 'C')]
En total son 7 subconjuntos posibles.

Sea el conjunto tiene 5 variables:
('A', 'B', 'C', 'D', 'E') → [('A',), ('B',), ('C',), ('D',), ('E',), ('A', 'B'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'C'), ('B', 'D'), ('B', 'E'), ('C', 'D'), ('C', 'E'), ('D', 'E'), ('A', 'B', 'C'), ('A', 'B', 'D'), ('A', 'B', 'E'), ('A', 'C', 'D'), ('A', 'C', 'E'), ('A', 'D', 'E'), ('B', 'C', 'D'), ('B', 'C', 'E'), ('B', 'D', 'E'), ('C', 'D', 'E'), ('A', 'B', 'C', 'D'), ('A', 'B', 'C', 'E'), ('A', 'B', 'D', 'E'), ('A', 'C', 'D', 'E'), ('B', 'C', 'D', 'E'), ('A', 'B', 'C', 'D', 'E')]
En total son 31 subconjuntos posibles.

La regla general es que dado un conjunto de N variables, existen 2^{N}-1 posibles subconjuntos. Si, por ejemplo, tenemos 50 variables, el número total de subconjuntos existente será de 2^{50}-1, es decir, 1.125.899.906.842.623 subconjuntos posibles.

Esta explosión combinatoria hace inviable el uso de una búsqueda exhaustiva, por lo que es necesario aplicar técnicas, como los algoritmos genéticos, que permiten realizar una exploración eficiente del espacio de soluciones.

Por último, es importante aclarar la razón por la que debemos evaluar subconjuntos de variables. Es decir, ¿por qué no evaluar la capacidad predictiva de cada variable de forma independiente y luego seleccionar aquellas con mayor capacidad predictiva? La respuesta es que existen interacciones no lineales entre variables o dicho de otro modo, dos variables que por separado tienen baja capacidad predictiva, pueden ser buenos predictores si son usados de forma conjunta. Del mismo modo, dos variables altamente predictivas pueden ser redundantes (ambas aportan la misma información sobre la variable respuesta), y por tanto, podemos prescindid de una de ellas sin perder capacidad predictiva.

2. Objetivos

El **objetivo principal** de este trabajo consistirá en realizar una implementación del algoritmo genético simple que permita seleccionar el mejor subconjunto de variables (aquel con mayor poder predictivo) para entrenar modelos predictivos sobre un conjunto de datos. El mejor subconjunto de variables (del conjunto de datos sobre el que se esté trabajando) será aquel que permita entrenar un modelo predictivo con el mayor rendimiento posible. Adicionalmente, se explorará el comportamiento del algoritmo variando los diferentes parámetros y operadores que lo configuran, con el objetivo de obtener el mejor resultado posible sobre los conjuntos de datos de prueba que se proporcionan. Este objetivo se descompone en los siguientes objetivos específicos:

- Todo el desarrollo realizado debe ser generalizable, es decir, no debe estar vinculado a un conjunto de datos concreto.
- Implementación del algoritmo genético simple.
- Implementar los operadores de cruce, mutación y selección seleccionados por los alumnos.
- Implementar una función de evaluación robusta, que a partir de un conjunto de datos de entrada y un individuo, proporcione una estimación robusta de la capacidad predictiva que tendría un árbol de decisión entrenado sobre el subconjunto de atributos codificado en el individuo.
- Se proporcionan varios conjuntos de datos en los que se ha comprobado que es posible obtener subconjuntos de variables que mejoren el rendimiento proporcionado por el conjunto de variables original. Estos conjuntos de datos se usarán para experimentar con el algoritmo desarrollado, mostrar su funcionamiento y obtener configuración del algoritmo genético que mejor funciona con cada conjunto.
- Documentar el trabajo en un fichero con formato de artículo científico, explicando con precisión las decisiones de diseño en la implementación de los algoritmos. Se debe mostrar que se ha entendido el problema de la selección de atributos en general y el funcionamiento de los algoritmos genéticos. De la misma forma se interpretarán los resultados obtenidos en los diferentes experimentos. Por último, de forma razonada se elegirá el mejor subconjunto de variables para cada conjunto de datos.
- Realizar una presentación de los resultados obtenidos en la defensa del trabajo.

Para que el trabajo pueda ser evaluado, se deben satisfacer TODOS los objetivos específicos al completo: el trabajo debe ser original, estar correctamente implementado y funcionar perfectamente, los experimentos se deben haber llevado a cabo y analizados razonadamente, el documento debe ser completo y contener un mínimo de 6 páginas, y se debe realizar la defensa con una presentación de los resultados obtenidos.

3. Descripción del trabajo

A continuación se proporciona información para el correcto desarrollo del trabajo.

3.1. Algoritmo genético simple

```
BEGIN /* Algoritmo Genetico Simple */
    Generar una poblacion inicial.
    Computar la funcion de evaluacion de cada individuo.
    WHILE NOT Terminado DO
    BEGIN /* Producir nueva generacion */
        FOR Tamaño poblacion/2 DO
        BEGIN /*Ciclo Reproductivo */
            Seleccionar dos individuos de la anterior generacion,
            para el cruce (probabilidad de seleccion proporcional
            a la funcion de evaluacion del individuo).
            Cruzar con cierta probabilidad los dos
            individuos obteniendo dos descendientes.
            Mutar los dos descendientes con cierta probabilidad.
            Computar la funcion de evaluacion de los dos
            descendientes mutados.
            Insertar los dos descendientes mutados en la nueva generacion.
        END
        IF  la poblacion ha convergido THEN
            Terminado := TRUE
    END
END
```

Como se ha comentado, el objetivo principal del trabajo es realizar una implementación del algoritmo genético simple aplicado al problema de selección de atributos. Esto implica las siguientes tareas:

Documentación: Aunque se ha proporcionado un documento de referencia, se recomienda que los alumnos busquen documentación adicional sobre algoritmos genéticos en la medida que lo consideren necesario. Es importante tener en cuenta el papel que juegan cada una de las fases del algoritmo, así como los diferentes tipos de operadores que podemos usar en ellas.

Representación del individuo: Definir la representación del individuo que será usada para codificar cada solución potencial a nuestro problema, es decir, para codificar los posibles subconjuntos de atributos que vayan a ser explorados por nuestro algoritmo. Nótese que los individuos siempre tienen el mismo número de genes. El tipo de gen más frecuente es el binario, aunque también pueden emplearse números enteros.

Implementación del algoritmo genético: Se recomienda realizar una implementación genérica del algoritmo genético simple, de forma que no esté vinculada a una función de evaluación u operadores de cruce, mutación y selección concretos. En otras palabras, una implementación en la que sea posible sustituir la función de evaluación de individuos o el operador de cruce entre dos individuos, sin realizar modificaciones adicionales.

Operadores de cruce, mutación y selección: Seleccionar e implementar los operadores de selección, cruce y mutación con los que se desee configurar el algoritmo genético. Se podrán probar más de un operador de cada tipo. En la **sección 3.3** se enumeran los operadores más comunes.

Criterio de parada: Seleccionar e implementar un criterio de parada (o más bien una combinación de estos) que permita decidir se termina la ejecución del algoritmo. Normalmente, estos criterios de parada buscan detectar que se ha encontrado una solución lo suficientemente buena o que ya ha pasado demasiado tiempo (iteraciones) sin que se encuentren soluciones mejores a la actual. En la **sección 3.4** se proporciona más información sobre los criterios de parada.

Función de evaluación de individuos: Definir el procedimiento de evaluación de los individuos (***fitness*** en inglés). Dado que un individuo representa un subconjunto de atributos, el procedimiento de evaluación entrenará y evaluará modelos predictivos usando el conjunto de atributos (columnas del conjunto de datos) codificado en el individuo. Por tanto, la evaluación del individuo vendrá dada por la capacidad predictiva de dichos modelos predictivos. En la **sección 3.2** se proporciona información detallada sobre cómo debe ser dicho procedimiento de evaluación de individuos.
Por un lado, debe tener en cuenta la posibilidad de evaluar el individuo (subconjunto de atributos) mediante más de un tipo de algoritmo predictivo (e.g. árbol de decisión, KNN o red neuronal). Por otro lado, debe repetir cada experimento de evaluación, mediante validación cruzada, varias veces, para mitigar el componente aleatorio que existe en la mayoría de algoritmos de aprendizaje automático.

Penalización de soluciones no válidas: Como se ha comentado, tan importante como obtener modelos predictivos con el mejor rendimiento posible es obtener modelos predictivos sencillos, es decir, que usen pocos atributos. Teniendo en cuenta esto, nuestro algoritmo tendrá un parámetro que permita fijar el tamaño máximo del subconjunto de atributos a seleccionar. En ese caso, se debe obtener el subconjunto de atributos que proporcione mayor rendimiento en los modelos y cuyo tamaño no supere el máximo establecido. Esta restricción debe ser implementada como una **penalización gradual** en la función de evaluación. Es decir, reduciremos cierta magnitud el valor de evaluación de los individuos que incumplan la restricción. Para que el algoritmo genético pueda asimilar estas restricciones de forma correcta, es necesario que sean graduales, es decir, que la penalización que recibe un individuo que excede la restricción por dos atributos debe ser mayor que la del individuo que la excede por un atributo.

3.2. Función de evaluación

Una de las piezas fundamentales del algoritmo es la función de evaluación de individuos (las soluciones candidatas) que servirá para seleccionar los mejores individuos y descartar los peores. En esta sección detallamos como sería un procedimiento de evaluación robusto. Para obtener una evaluación robusta, debemos realizar varios experimentos mediante **validación cruzada**[3] y promediar la tasa de aciertos obtenida. Es decir, aunque la propia validación cruzada ya nos proporciona cierta robustez en la estimación, debido a la aleatoriedad existente en la mayoría de algoritmos de aprendizaje automático, además vamos a promediar varios experimentos de validación cruzada. Con esto buscamos que dos evaluaciones diferentes sobre el mismo individuo, sean lo más parecidas posible.

Para esto, el framework de python Scikit-learn nos proporciona una función[4] para realizar evaluaciones mediante validación cruzada, por lo que no será necesario implementarla. Esta función nos devolverá el valor para la métrica seleccionada (que indica la capacidad predictiva) de nuestro modelo.

3 https://es.wikipedia.org/wiki/Validación_cruzada
4 https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

Importante: Existen decenas de métricas[5] para evaluar la capacidad predictiva de un algoritmo, donde la más sencilla es la tasa de aciertos (predicciones_acertadas/total_predicciones) o *accuracy* en inglés. Para este trabajo se desea utilizar la tasa de aciertos balanceada[6], en inglés *balanced_accuracy_score*. Para usarla, bastará con especificar el valor '*balanced_accuracy*' en el parámetro '*scoring*' de la función de validación cruzada (*cross_val_score*). Se recomienda usar el parámetro n_jobs=-1 de *cross_val_score* para paralelizar la ejecución.

En líneas generales, el procedimiento de evaluación debe ser como sigue:

- Tendrá un parámetro que indique el número de repeticiones (experimentos mediante validación cruzada) a realizar (e.g. 5) y promediar.
- Los experimentos se realizarán usando el subconjunto de atributos codificado en el individuo que está siendo evaluado.
- Dado que cada ejecución por validación cruzada devuelve varias medidas de rendimiento, también debemos promediarlas.
- El usuario podrá configurar el algoritmo genético para considerar más de un algoritmo de aprendizaje automático en la evaluación (e.g. árboles de decisión y KNN). En ese caso, primero se realizará una evaluación robusta para cada algoritmo y luego se combinarán las puntuaciones obtenidas.

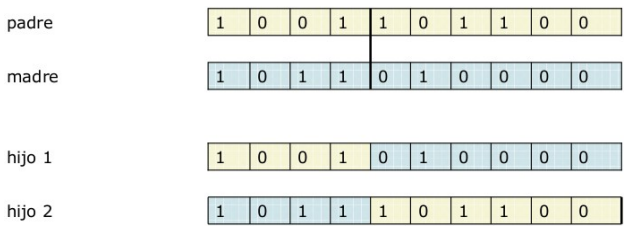
Importante: Dado que la evaluación de cada individuo es costosa, el algoritmo debe almacenar, durante toda la ejecución, los individuos ya procesados y su valoración, para evitar evaluarlos de nuevo. En el algoritmo genético es muy frecuente que los individuos se repitan, por tanto, la primera vez que se evalúe un individuo se almacenará su valoración para volver a usarla cada vez que el individuo vuelva a aparecer en la población, sin tener que repetir cálculos. **Sin este detalle, el tiempo de ejecución del algoritmo sería excesivo.**

3.3. Operadores de cruce, mutación y selección

En esta sección se enumeran los operadores más comunes usados en algoritmos genéticos. Los alumnos tendrán libertad para elegir los operadores que desean usar. Aunque en esta sección se describen los más comunes, está permitido usar cualquiera existente en la literatura.

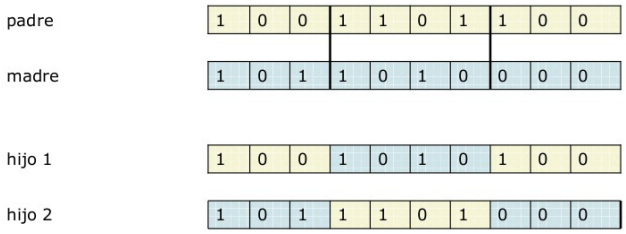
Operadores de cruce: Reciben dos individuos y devuelven los individuos que resultan de combinarlos.

Cruce en un punto:



5 https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter
6 https://scikit-learn.org/stable/modules/model_evaluation.html#balanced-accuracy-score

Cruce en dos puntos:



Cruce multipunto: Similar al caso anterior pero considerando más segmentos de intercambio.

Cruce uniforme: Para cada posición de cada hijo se decide aleatoriamente de qué progenitor hereda.

Operadores de mutación: Producen modificaciones aleatorias en un solo individuo. Con cierta probabilidad se modifica aleatoriamente el valor de uno o varios genes del individuo.

Operadores de selección: Definen el criterio para seleccionar los individuos de la generación actual que formarán parte de la siguiente. La idea es seleccionar los mejores individuos (según la función de evaluación) incluyendo con cierta probabilidad individuos con peor valoración, para mantener así la diversidad de la población.

Selección probabilística o método de la ruleta: La probabilidad de que un individuo sea seleccionado es proporcional a su valoración. Por tanto, se selecciona un individuo aleatoriamente, pero tendrán más probabilidad de ser seleccionados aquellos con mayor valoración.

Selección por torneo de tamaño K: Se seleccionan aleatoriamente (sin tener en cuenta la evaluación) K individuos y de estos K individuos se selecciona el que tenga mayor valoración.

Para producir la nueva generación, se repite el proceso de selección tantas veces como individuos tenga la población. Nótese que un individuo puede ser seleccionado más de una vez.

3.4. Criterio de parada

Dado que los algoritmos genéticos son aproximados, es decir, no garantizan encontrar una solución óptima, debemos decidir cuando detener la ejecución del algoritmo. Se puede usar más de un criterio de parada, en ese caso, el que se cumpla antes será el que tenga efecto:

- Número máximo de generaciones (iteraciones).
- Número de iteraciones sin que se consiga mejorar la mejor solución encontrada hasta el momento.
- Umbral sobre la varianza de la valoración de la población actual. Si la varianza de la valoración de la población es muy baja, significa que la mayor parte de los individuos son similares y que, por tanto, la población se está estancando y las posibilidades de encontrar una solución mejor son muy bajas. El valor del umbral debe ser ajustado para el problema y suelen ser valores pequeños, como por ejemplo, 0.001.
- Los alumnos podrán usar otros criterios de parada que consideren oportunos.

3.5. Parámetros de entrada del algoritmo

El algoritmo podrá ser configurado mediante los siguientes parámetros de entrada:

- Tamaño de la población: número de individuos que componen la población a evolucionar.
- Estimadores (lista de algoritmos): Uno o varios algoritmos objetivo de la optimización. Se usarán los diferentes algoritmos de clasificación disponibles en Scikit-learn. Esta lista contiene objetos ya instanciados de los diferentes algoritmos.
- Número de repeticiones de la evaluación: El número de veces que queremos repetir la evaluación de cada modelo, mediante evaluación cruzada, para obtener una valoración robusta, según se describió en el apartado 3.2.
- Probabilidad de cruce: Probabilidad de que dos individuos se crucen y sean sustituidos por sus descendientes. En caso contrario, los individuos pasarán sin cambios a la siguiente generación.
- Probabilidad de mutación: Probabilidad de que un individuo mute.
- Número máximo de atributos: Solo se considerarán soluciones válidas aquellas que contengan un número de atributos inferior a este parámetro. Las soluciones no válidas serán penalizadas según se comentó anteriormente.
- Criterio de parada: Según el o los criterios de paradas seleccionados, se incluirán los parámetros necesarios para su configuración.
- Operadores de cruce, mutación y selección: Según los operadores seleccionados, es posible que se necesiten incluir parámetros de configuración adicionales.
- Otros: Cualquier parámetro que los alumnos estimen oportuno para configurar sus algoritmos.

3.6. Salida del algoritmo

El algoritmo almacenará, además de la mejor solución encontrada hasta el momento, todas las soluciones exploradas, junto con su valoración, en cada una de las generaciones (iteraciones) y devolverá las mejoras. Esto es importante, ya que normalmente la mejor solución no estará en la última generación, sino que aparecerá en iteraciones intermedias.

Los alumnos podrán decidir como presentar los resultados obtenidos por el algoritmo. Adicionalmente, se recomienda imprimir información (estadísticos que describan a la población y otra información que se considere oportuna) en cada iteración, de forma que se pueda seguir la convergencia del algoritmo.

3.7. Conjuntos de datos

Se proporcionan cuatro conjuntos de datos en el que se ha comprobado que es posible obtener subconjuntos de variables que mejoren el rendimiento proporcionado por el conjunto de variables completo:

- Conjunto de datos sobre el Titanic:[7] Contiene información de pasajeros que viajaban el Titanic. El objetivo es predecir, a partir de las características de los pasajeros, quién sobrevive y quién no. El nombre de la columna que contiene la variable respuesta es s*urvived*.

7 https://www.kaggle.com/c/titanic/data

- Conjunto de datos sobre cáncer de mama:[8] Contiene información sobre características físicas de diferentes masas tumorales. El objetivo es predecir, a partir de estas características, si la masa representa un cáncer benigno o maligno. El nombre de la columna que contiene la variable respuesta es *diagnosis*.
- Conjunto de datos sobre la enfermedad de Parkinson:[9] Contiene información tanto sobre personas sanas como de personas que tienen la enfermedad. Dicha información consiste en características extraídas de grabaciones de voz. La variable respuesta, que indica si el registro corresponde a una persona con Parkinson o no, se llama *status*.
- Conjunto de datos sobre el Síndrome del ovario poliquístico (PCOS):[10] Conjunto de datos con indicadores fisiológicos de personas que pueden padecer la enfermedad o no. La variable respuesta, que indica si el registro corresponde a una persona con PCOS o no, se llama *PCOS (Y/N)*.

Importante: los ficheros que se proporcionan han sido modificados, por lo que es necesario trabajar con estos y no con los que se pueden obtener en internet. Los conjuntos de datos han sido tratados para que el alumno no tenga que realizar ningún tipo de preprocesado o limpieza en los datos. Estos están listos para ser usados con algoritmos de aprendizaje de Scikit-learn. La variable respuesta es siempre la última columna.

3.8. Experimentación

Una vez que se ha completado la implementación del algoritmo y que se ha verificado que funciona correctamente, se experimentará con al menos dos de los conjuntos de datos propuestos. El objetivo de esta experimentación es entender y documentar el comportamiento del algoritmo con cada conjunto de datos al variar los parámetros de entrada. Téngase en cuenta que la configuración que proporcione mejores resultados con un conjunto de datos puede no ser óptima al usarse sobre otro conjunto.

Si se decide no trabajar con todos los conjuntos de datos, se recomienda que uno de los que se excluyan sea el del Titanic ya que contiene menos atributos.

3.9. Otros detalles y recomendaciones de implementación

Caché de evaluaciones: Como se comentó en el apartado 3.2, es imprescindible almacenar los individuos ya evaluados para evitar repetir cálculos de forma innecesaria. En caso contrario, la ejecución tomaría demasiado tiempo.

Desarrollo y depuración: Según los parámetros que se hayan especificado en el algoritmo, el conjunto de datos usado y los recursos computacionales disponibles, algunas ejecuciones pueden tardar mucho tiempo (hasta 60 minutos). Por tanto, en la fase de implementación y depuración, se recomienda usar configuraciones que requieran poco tiempo de ejecución:

- Tamaño de población reducido
- No usar repeticiones en el procedimiento de evaluación robusta, es decir, fijar el parámetro que controla el número de repeticiones a 1.
- Empezar usando un solo estimador

8 https://www.kaggle.com/uciml/breast-cancer-wisconsin-data
9 https://archive.ics.uci.edu/ml/datasets/parkinsons
10 https://www.kaggle.com/datasets/prasoonkottarathil/polycystic-ovary-syndrome-pcos

- Implementar la caché de evaluaciones desde el principio.
- Empezar con el conjunto de datos sobre el Titanic o sobre el Parkinson (incluso seleccionando el 50% de filas).

Una vez que se haya depurado el código, pasaríamos a hacer experimentos con parámetros más realistas y que proporcionen mejores resultados.

Experimento de referencia: Se recomienda realizar un primer experimento de referencia para cada conjunto de datos y los diferentes conjuntos de estimadores seleccionados. La idea es obtener medida de rendimiento de referencia, mediante la función de evaluación, usando todos los atributos del conjunto de datos. Esta referencia servirá para compararla con los resultados obtenidos mediante el algoritmo implementado, ya que se estera que estos últimos sean mejores.

Librerías externas: Se pueden usar funciones ya implementadas para el cálculo de métricas, entrenamiento de modelos predictivos o evaluación mediante validación cruzada. Las principales librerías a considerar para el trabajo serán, Pandas y/o Numpy, Scikit-learn y opcionalmente Matplotlib o Seaborn si se desea incluir algún gráfico. No se permite el uso de funciones de alto nivel que implementen el algoritmo genético o alguna de sus partes, como pueden ser los operadores de cruce, mutación y selección.

Documentación previa: Antes de comentar a implementar el trabajo, se recomienda que los alumnos se familiaricen con:

- Las librerías para manipulación de conjuntos de datos Pandas y Numpy.
- El framework de aprendizaje automático Scikit-learn.
- El entrenamiento de modelos predictivos y su evaluación mediante validación cruzada.
- Los conjuntos de datos proporcionados.
- La problemática de la selección de atributos en general.
- Los algoritmos de genéticos, así como los diferentes operadores de cruce, mutación y selección existentes. Se recomienda entenderlos bien antes de empezar a escribir código. Para esto, además de consultar la documentación que se considere oportuna, se debe haber leído y comprendido el documento de referencia, al menos, las 13 primeras páginas. Como ayuda para empezar a desarrollar el trabajo, puede empezar a implementar el ejemplo 2.2 (encontrar el máximo de la función $f(x) = x^2$) del documento de referencia, si lo considera necesario.

3.10. Presentación del código

El trabajo debe presentarse en forma de notebook de jupyter que puede ser desarrollados tanto con jupyter-notebook como jupyter-lab, ambos disponibles en el paquete Anaconda. Se proporcionarán las implementaciones solicitadas, así como su aplicación sobre los conjuntos de datos proporcionados. El código y los experimentos realizados deben estar debidamente documentados, las decisiones tomadas debidamente justificadas y los resultados obtenidos deben ser interpretados.

3.11. Documentación

En el fichero `plantilla-trabajo.doc` se muestra una sugerencia de estructura y formato de estilo artículo científico correspondiente a la documentación del trabajo. Este formato es el del

IEEE conference proceedings, cuyo sitio web *guía para autores* [6] ofrece información más detallada y plantillas para Word y Latex.

El artículo deberá tener una extensión **mínima de 6**, y la estructura general del documento debe ser como sigue: en primer lugar realizar una **introducción** al trabajo explicando el objetivo fundamental, incluyendo un breve repaso de antecedentes en relación con la temática del trabajo. A continuación, describir la **estructura** del trabajo, las **decisiones de diseño** que se hayan tomado a lo largo de la elaboración del mismo, y la **metodología** seguida al implementarlo (nunca poner código, pero sí pseudocódigo), y seguidamente detallar los **experimentos** llevados a cabo, **analizando los resultados** obtenidos en cada uno de ellos. Por último, el documento debe incluir una sección de **conclusiones**, y una **bibliografía** donde aparezcan no solo las referencias citadas en la sección de introducción, sino cualquier documento consultado durante la realización del trabajo (incluidas las referencias web a páginas o repositorios).

3.12. Presentación y defensa

El día de la defensa se deberá realizar una pequeña presentación (PDF, PowerPoint o similar) de 10 minutos en la que participarán activamente todos los miembros del grupo que ha desarrollado el trabajo. Esta presentación seguirá a grandes rasgos la misma estructura que el documento, pero se deberá hacer especial mención a los resultados obtenidos y al análisis crítico de los mismos. Se podrá usar un portátil (personal del alumno), diapositivas y/o pizarra. En los siguientes 10 minutos de la defensa, el profesor procederá a realizar preguntas sobre el trabajo, que podrán ser tanto del documento como del código fuente. Adicionalmente, el profesor podrá proporcionar un nuevo conjunto de datos con el que probar el algoritmo implementado.

4. Criterios de evaluación

Para que el trabajo pueda ser evaluado, se deberá satisfacer los objetivos concretos descritos en el apartado 1 (todos y cada uno de ellos, si no, el trabajo obtendrá una nota de suspenso). Uno de los alumnos del equipo deberá subir a través del formulario disponible en la página de la asignatura un fichero comprimido .zip, que contenga:

- **Una carpeta con el código fuente**. **S**e proporcionarán instrucciones sobre como reproducir la experimentación realizada. El código fuente se entregará en forma de notebook de Jupyter. Dicha carpeta debe contener todo lo necesario para ejecutar el notebook. Además, la carga de datos desde el notebook debe realizarse usando rutas relativas para que la ejecución pueda replicarse en otro ordenador sin problemas.
- **El documento – artículo en formato PDF**. Deberá tener una extensión mínima de 6 páginas. Deberá incluir toda la bibliografía consultada (libros, artículos, technical reports, páginas web, códigos fuente, diapositivas, etc.) en el apartado de referencias, y mencionarlas a lo largo del documento.

Para la evaluación se tendrá en cuenta el siguiente criterio de valoración, considerando una nota máxima de 4 en total para el trabajo:

- **El código fuente y resultados obtenidos (hasta 2 puntos)**: Se valorará la claridad y buen estilo de programación, corrección, eficiencia y usabilidad de la implementación, y calidad de los comentarios. La claridad del fichero README.txt también se valorará. En ningún

caso se evaluará un trabajo con código copiado directamente de Internet o de otros compañeros.

En este trabajo está permitido utilizar en parte librerías de Python existentes, aunque en este apartado se valorará exclusivamente el código original desarrollado por los alumnos.

Con respecto a la experimentación, se valorará la calidad y completitud de los experimentos realizados con el algoritmo implementado. Además, se tendrá en cuenta el rendimiento del algoritmo, la completitud de la implementación y experimentación realizadas, así como los resultados obtenidos. Por último, no se tendrán en cuenta aquellos resultados experimentales que no sean reproducibles.

- **El documento – artículo científico (hasta 1 punto)**:
  - Se valorará el uso adecuado del lenguaje y el estilo general del documento (por ejemplo, el uso de la plantilla sugerida).
  - Se valorará en general la claridad de las explicaciones, el razonamiento de las decisiones, y especialmente el análisis y presentación de resultados en las secciones de experimentación y conclusiones.
  - Igualmente, no se evaluará el trabajo si se detecta cualquier copia del contenido del documento. La sección de referencias deberá incluir menciones a todas las pertinentes fuentes consultadas.
- **La presentación y defensa (hasta 1 puntos)**: se valorará la claridad de la presentación y la buena explicación de los contenidos del trabajo, así como, especialmente, las respuestas a las preguntas realizadas por el profesor.

IMPORTANTE: Cualquier **plagio**, **compartición de código** o uso de material que no sea original y del que no se cite convenientemente la fuente, significará automáticamente la **calificación de cero** en la asignatura para **todos los alumnos involucrados**. Por tanto, a estos alumnos **no se les conserva**, ni para la actual ni para futuras convocatorias, **ninguna nota** que hubiesen obtenido hasta el momento. Todo ello sin perjuicio de las correspondientes **medidas disciplinarias** que se pudieran tomar.