Algoritmos de búsqueda para selección de características en modelos predictivos

Inteligencia Artificial (IS) 2023/24 – Propuesta de trabajo

Juan Galán Páez

1. Introducción al problema y conceptos básicos

1.1. Selección de características 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 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 características[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 características 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.
- **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 características es uno de los problemas bajo estudio en la actualidad (y desde hace décadas) más importante dentro del aprendizaje automático. En la actualidad 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 https://en.wikipedia.org/wiki/Feature_selection

Las numerosas técnicas de selección de características existentes se agrupan en tres familias[2]. Aunque el presente trabajo se centra en los **métodos de envoltura (*wrapper methods*)**, a continuación se introducen los tres:

- **Métodos de filtro (*filter methods*):** Estos métodos son independientes del algoritmo de entrenamiento utilizado y están basados fundamentalmente en tests estadísticos realizados sobre las variables. Usaríamos los resultados de los diferentes tests estadísticos para filtrar el conjunto de variables. Por ejemplo, seleccionar las variables predictoras con mayor correlación sobre la variable respuesta, o bien, si tenemos varias variables predictoras muy correlacionadas entre sí, nos quedamos con una y descartamos las demás.
- **Métodos de envoltura (*wrapper methods*):** En este caso, el algoritmo de entrenamiento si interviene en el proceso de selección de características. El método consiste en seleccionar diferentes subconjuntos de características, entrenar un modelo predictivo con cada uno de estos subconjuntos y evaluar su rendimiento, finalmente seleccionamos el mejor subconjunto. Este tipo de métodos son los que mayor coste computacional tienen.
- **Métodos integrados (embedded methods):** Existen algoritmos de aprendizaje que tienen sus propios mecanismos de selección de características. Durante el proceso de aprendizaje generan información útil para evaluar lo buena o mala que es una variable. Por ejemplo, los árboles de decisión usan la ganancia de información que cada variable aporta en cada nodo.

Es importante mencionar que el mejor subconjunto de características no tiene por qué ser aquel que mayor rendimiento proporcione, ya que buscamos un compromiso entre número de características (el menor posible) y rendimiento (el mayor posible). Por ejemplo, supongamos que tenemos un conjunto inicial de 50 características y tras aplicar el método de envoltura los tres mejores subconjuntos obtenidos son los siguientes:

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

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

Método de envoltura. Fuente: https://ligdigonzalez.com/metodos-de-seleccion-de-caracteristicas-machine-learning/

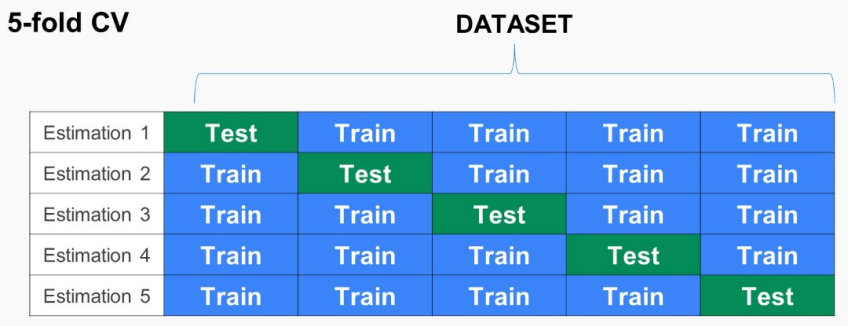
El **presente trabajo se centra en la implementación de un método de selección de características de tipo envoltura (*wrapper methods*)**. Los métodos de envoltura necesitan un algoritmo de aprendizaje, que puede ser de clasificación o regresión, para evaluar cada uno de los subconjuntos candidatos.

2 https://en.wikipedia.org/wiki/Feature_selection#Main_principles

1.2. Validación cruzada

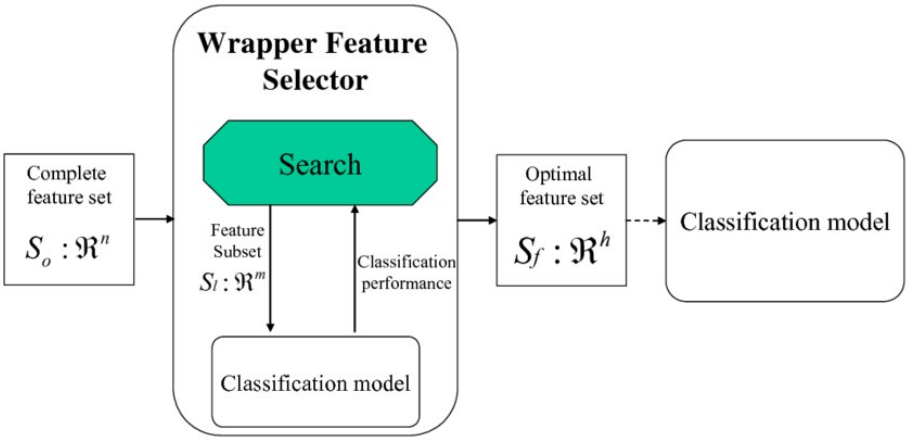
Los métodos de selección de características de tipo envoltura, necesitan evaluar cada solución candidata (subconjunto de variables). Para esto, entrenan un algoritmo de aprendizaje y evalúan su capacidad predictiva. Para reducir la aleatoriedad de los resultados, dicha evaluación se realiza mediante validación cruzada.

Veamos una definición de validación cruzada (extraída de Wikipedia)[3]: *En la validación cruzada de K iteraciones o K-fold cross-validation los datos se dividen en K subconjuntos. Uno de los subconjuntos se utiliza como datos de evaluación (datos a predecir y sobre los que evaluar el rendimiento) y el resto (K-1) como datos de entrenamiento. El proceso de validación cruzada es repetido durante k iteraciones, con cada uno de los posibles subconjuntos de datos de prueba. Finalmente, se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. Este método es muy preciso, puesto que evaluamos a partir de K combinaciones de datos de entrenamiento y de prueba. En la práctica, la elección del número de iteraciones depende de la medida del conjunto de datos.*



En la imagen podemos ver el esquema de validación cruzada para 5 folds. En cada uno de los 5 experimentos obtenemos un valor para la métrica de evaluación de la capacidad predictiva sobre el conjunto de evaluación, es decir, el segmento de color verde en cada caso. Finalmente, promediamos las 5 estimaciones.

1.3. Algoritmos de búsqueda para selección de características



3 https://es.wikipedia.org/wiki/Validación_cruzada

Los métodos de envoltura, son procesos iterativos en los que en cada iteración se selecciona un subconjunto de variables, se entrena un modelo predictivo, y se evalúa su rendimiento. Este proceso se repite para diferentes subconjuntos de características. Al terminar seleccionamos el subconjunto de características que mejor compromiso entre rendimiento y tamaño proporcione.

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

Si el conjunto tiene 5 variables, sería:
('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 $2^{\wedge N}$-1 posibles subconjuntos. Si por ejemplo, tenemos 50 variables, el número total de subconjuntos existente será de $2^{\wedge 50}$-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 diseñar estrategias de búsqueda que obtengan el mejor resultado posible probando una pequeña parte de los subconjuntos posibles.

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. De la misma forma 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.

Estrategias de búsqueda secuenciales:
Aunque existen enfoques más complejos, nos centraremos en estrategias de búsqueda sencillas, como son las secuenciales. Estas estrategias se descomponen en tres grandes grupos:

- Hacia delante: Partimos de un **conjunto vacío y en cada iteración añadimos una** nueva variable (la que consideramos mejor a priori).
- Hacia atrás: Partimos del **conjunto de todas las variables y en cada iteración eliminamos una** variable (la que consideramos peor a priori).
- Mixtas: Combinan ambas direcciones. Existe una dirección principal y la otra se usa como secundaria para 'corregir' soluciones no optimales. Por ejemplo, si tomamos la búsqueda hacia delante como principal, en cada paso, existirá la posibilidad de eliminar una variable

(es decir, dar un paso atrás) si es que eso produce una mejora, en caso contrario, no eliminamos nada.

A continuación se describen las estrategias de búsqueda objetivo de este trabajo:

A.1) Búsqueda secuencial hacia delante:

El algoritmo de búsqueda secuencial hacia delante empieza con un conjunto vacío de variables. En cada iteración se selecciona, de entre las variables pendientes, **la mejor variable** a añadir a este conjunto. Para seleccionar la mejor variable a añadir, en cada iteración se evalúan los posibles candidatos. El algoritmo termina cuando se hayan añadido todas las variables. La salida será una tabla que contenga el mejor subconjunto encontrado en cada iteración.

1. Start with the empty set $Y_0 = \{\emptyset\}$
2. Select the next best feature $x^+ = \arg\max_{x \notin Y_k} J(Y_k + x)$
3. Update $Y_{k+1} = Y_k + x^+;\ k = k + 1$
4. Go to 2

A.2) Búsqueda secuencial hacia delante mixta:

En el algoritmo de búsqueda secuencial hacia delante mixta la estrategia principal es hacia delante y la secundaria hacia atrás Es similar al algoritmo anterior, sin embargo, en cada iteración se añade la posibilidad de eliminar una variable previamente añadida, solo si el subconjunto obtenido tras eliminar la variable tiene mejor capacidad predictiva. El algoritmo termina cuando la solución se estabiliza, es decir, el subconjunto de variables actual no es mejorable ni añadiendo ni quitando variables.
La salida será una tabla que contenga el mejor subconjunto encontrado en cada iteración. En estos algoritmos siempre devolvemos una tabla de mejores soluciones y no solo la mejor, ya que, como se comentó anteriormente, a veces preferimos elegir una solución que tiene menos rendimiento pero también menos variables.

1. $Y = \{\emptyset\}$
2. Select the best feature
   $x^+ = \arg\max_{x \notin Y_k} J(Y_k + x)$
   $Y_k = Y_k + x^+; k = k + 1$
3. Select the worst feature*
   $x^- = \arg\max_{x \in Y_k} J(Y_k - x)$
4. If $J(Y_k - x^-) > J(Y_k)$ then
   $Y_{k+1} = Y_k - x^-;\ k = k + 1$
   Go to step 3
   Else
   Go to step 2

Por último, es importante mencionar que el algoritmo que se muestra en la figura podría acabar en un bucle infinito, para evitar esto, es necesario llevar el control, de alguna forma, de las soluciones visitadas.

A.3) Búsqueda secuencial hacia atrás:

El algoritmo de búsqueda secuencial hacia atrás empieza con el conjunto de todas las variables. En cada iteración se selecciona, la peor variable a eliminar de este conjunto. Para seleccionar **la peor variable** a eliminar en cada iteración, se evalúan los posibles candidatos. El algoritmo termina cuando se hayan eliminado todas las variables. La salida será una tabla que contenga el mejor subconjunto encontrado en cada iteración.

1. Start with the full set $Y_0 = X$
2. Remove the worst feature $x^- = \arg\max_{x \in Y_k} J(Y_k - x)$
3. Update $Y_{k+1} = Y_k - x^-;\ k = k + 1$
4. Go to 2

A.4) Búsqueda secuencial hacia atrás mixta:

El algoritmo de búsqueda secuencial hacia atrás mixta es análogo a la búsqueda secuencial hacia delante mixta pero con las direcciones invertidas. En este caso, la estrategia principal es hacia atrás y la secundaria hacia delante. Es decir, en cada iteración se elimina la peor variable y se añade la posibilidad de añadir una variable previamente eliminada, solo si el subconjunto obtenido tras esta adición tiene mejor capacidad predictiva. El algoritmo termina cuando la solución se estabiliza, es decir, el subconjunto de variables actual no es mejorable ni añadiendo ni quitando variables. La salida será una tabla que contenga el mejor subconjunto encontrado en cada iteración.

2. Objetivos

El **objetivo principal** de esta propuesta es desarrollar nuestro propio algoritmo de selección de características. Según lo visto anteriormente, el objetivo es implementar un algoritmo de **búsqueda secuencial** y un algoritmo de **búsqueda secuencial mixta**. Según la **convocatoria**, se implementarán los algoritmos hacia delante o hacia atrás y se aplicarán a problemas de regresión o clasificación. En todos los casos, será necesario implementar un método de evaluación robusta que se detallará en la siguiente sección. Por último, se proporcionan dos conjuntos de datos sobre los que se deben aplicar los algoritmos desarrollados. 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 o problema concreto. Para esto, el código debe estar parametrizado según sea necesario.
- Implementar un método de evaluación robusta, que a partir de un conjunto de datos de entrada, un subconjunto de variables del mismo y un algoritmo de clasificación/regresión proporcione una estimación robusta de la capacidad predictiva que tendría ese algoritmo entrenado sobre el subconjunto de datos proporcionado. La métrica de evaluación a usar dependerá de la convocatoria.
- Implementar los algoritmos de **búsqueda secuencial** y **búsqueda secuencial mixta** que, a partir de un conjunto de datos de entrada y un algoritmo de clasificación/regresión, devuelva una tabla con los subconjuntos de variables más prometedores, es decir, aquellos con mayor capacidad predictiva.
- Se proporcionan dos 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 completo. Para esto, es importante, realizar un primer experimento en el que se evalúe la capacidad predictiva del conjunto de variables completo. Esta medida de rendimiento servirá de referencia para evaluar la calidad de las soluciones que nuestro algoritmo de búsqueda proporciona.
- Realizar diferentes experimentos con los algoritmos de selección de características implementados, aplicándolos sobre los conjuntos de datos proporcionados. En estos experimentos, además de buscar el mejor conjunto de características, se probarán diferentes hiperparámetros de los algoritmos de clasificación/regresión con el objetivo de mejorar todavía más el rendimiento.
- Documentar el trabajo en un fichero con formato de artículo científico, explicando con precisión las decisiones de diseño en la implementación de los algoritmos de búsqueda. Se debe mostrar que se ha entendido el problema de la selección de características en general y el funcionamiento de los algoritmos desarrollados en particular. 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.

3. Detalles de implementación

En esta sección se proporciona una guía para el correcto desarrollo del trabajo.

Se pueden usar funciones ya implementadas para el preprocesado de datos, cálculo de métricas, entrenamiento de algoritmos de clasificación/regresión o evaluación mediante validación cruzada. No se permite el uso de funciones de alto nivel que realicen el procedimiento de búsqueda.

Antes de comenzar a implementar el trabajo, se recomienda que el alumno se familiarice con:
- Las librerías para manipulación de conjuntos de datos Pandas y Numpy.
- El framework de aprendizaje automático Scikit-learn.
- Los conjuntos de datos proporcionados.
- La problemática de la selección de características en general.
- Los algoritmos de búsqueda a implementar. Se recomienda entenderlos bien antes de empezar a escribir código.
- La práctica de aprendizaje automático.

A continuación se proporciona información que podrá servir de guía para la implementación de los diferentes algoritmos, sin embargo, no es exhaustiva ni es obligatorio seguirla. Por último, téngase en cuenta que de los 4 algoritmos que se describen en las subsecciones 3.2 a 3.5 solo se deberán implementar dos en cada **convocatoria** (ver sección 4).

3.1. Evaluación de soluciones:

Una de las piezas fundamentales del procedimiento de búsqueda es la función de evaluación de las soluciones candidatas que permitirá elegir la solución más prometedora en cada paso. En esta sección detallamos como sería un procedimiento de evaluación robusto. Para obtener una evaluación robusta, debemos realizar varios experimentos y promediar la medida de rendimiento

obtenida. Es decir, aunque la propia validación cruzada ya nos proporciona cierta robustez en la estimación, debido a la aleatoriedad existente en los procesos a usar, además vamos a promediar varios experimentos basados en validación cruzada.

Scikit-learn proporciona una función (*cross_val_score*)[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 algoritmo. Existen decenas de métricas[5] para evaluar la capacidad predictiva de un algoritmo y más adelante se indicará la métrica más adecuada en cada caso. Si desea especificar más de una métrica de rendimiento, entonces deberá usar el método cross_validate[6]. En cualquiera de estos dos métodos se recomienda usar el parámetro *n_jobs* para ejecutar experimentos en paralelo y acelerar el entrenamiento (consulte la documentación para más información).

Se propone el siguiente método para obtener una evaluación robusta de cada subconjunto de variables:

Entrada:

- Datos: Conjunto de datos completo. Contiene todas las filas y todas las columnas (variables predictoras y variable respuesta)
- Variables: Subconjunto de variables (sus nombres) a evaluar. Es un subconjunto de las columnas contenidas en Datos. Es importante recordar, que la variable respuesta nunca podrá ser incluida en el conjunto de datos de entrenamiento.
- Algoritmo de entrenamiento: Instancia del algoritmo de entrenamiento a usar para entrenar un modelo con las variables seleccionadas
- N_Exp: número de repeticiones del experimento por validación cruzada. Valores recomendados entre 5 y 20. Cuando mayor sea el valor, más robusto será el resultado pero mayor el tiempo de cálculo.
- CV: Número de pliegues (folds) a considerar en la validación cruzada (entre 5 y 10).

Salida:

- Métrica de rendimiento promedio obtenida.

Ejecución:

1. Seleccionar del conjunto de datos de entrada, el subconjunto de columnas (variables) que queremos evaluar.
2. Repetir N_Exp veces y promediar el resultado:
   2.1. Realizar experimento de validación cruzada (siendo CV el número de folds) mediante la función ‘cross_val_score’
3. Devolver el resultado promedio.

Consideraciones:

- *Antes de comenzar con la implementación, se recomienda que el alumno haga pruebas de entrenamiento y predicción con modelos individuales para familiarizarse con los datos y la métrica de evaluación.*
- *Mientras desarrollamos nuestros algoritmos, se recomienda usar N_Exp=1 y CV=3 para que los experimentos sean más rápidos. Una vez que todo esté funcionando correctamente pondremos los valores adecuados.*

4 https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
5 https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter
6 https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html

3.2. (A.1) Búsqueda secuencial hacia delante:

Descripción detallada del algoritmo de búsqueda secuencial hacia delante:

Entrada:
- Conjunto de datos con N variables predictoras y una variable respuesta.
- Otros parámetros que considere necesarios.

Salida:
- Tabla con las combinaciones obtenidas en cada iteración, su tamaño y su rendimiento.

Inicialización:
- *SolucionActual*: Almacena el mejor conjunto de variables obtenido en cada iteración. Inicialmente está vacío.

Ejecución:
Desde K=1 hasta N:
1. Seleccionar la mejor variable para añadir a *SolucionActual*. Por cada variable V del conjunto original de variables que no se encuentre en *SolucionActual*:
   1.1. *SolucionTemporal* = *SolucionActual* + V
   1.2. Evaluar, mediante evaluación robusta (3.1), *SolucionTemporal* y guardar su rendimiento.
2. Seleccionar la mejor *SoluciónTemporal* (la que proporcione mayor rendimiento) y hacer *SolucionActual* = *MejorSolucionTemporal*

Devolver tabla con cada una de las *MejorSolucionTemporal*, el tamaño y el rendimiento de cada una. Es decir, la tabla contendrá K soluciones, cada una de un tamaño, donde la primera será de tamaño 1 y la última de tamaño K.

3.3. (A.2) Búsqueda secuencial hacia delante mixta:

La descripción proporcionada anteriormente (sección 1.3) podría acabar en un bucle infinito, para evitar esto, es necesario llevar el control de las soluciones ya procesadas. A continuación se propone una versión que realiza un control de variables visitadas (el control también podría realizarse a nivel de soluciones visitadas). Esta solución es una propuesta y no es obligatorio ceñirse a la misma.

Entrada:
- Conjunto de datos con N variables predictoras y una variable respuesta.
- Otros parámetros que considere necesarios.

Salida:
- Tabla con las combinaciones obtenidas en cada iteración, su tamaño y su rendimiento.

Inicialización:
- *SolucionActual*: Almacena el mejor conjunto de variables obtenido en cada iteración. Inicialmente está vacío.
- *Añadidos*: Almacena las variables que han sido añadidas.
- *Eliminados*: Almacena las variables que han sido eliminadas.

Ejecución:
Mientras que no se cumpla la CondicionDeParada:

---------AÑADIR MEJOR VARIABLE---------

1. Seleccionar la **mejor** variable para añadir a *SolucionActual*. Por cada variable V del conjunto original de variables que no se encuentre en *SolucionActual* ni en *Añadidos*:
   1.1. *SolucionTemporal* = *SolucionActual* + V
   1.2. Evaluar, mediante evaluación robusta (3.1), *SolucionTemporal* y guardar su rendimiento.
2. Seleccionar la mejor *SoluciónTemporal* (la que proporcione mayor rendimiento) y hacer *SoluciónActual* = *MejorSolucionTemporal*
3. Actualizar *Añadidos:* Añadimos a esta lista la nueva variable añadida a *SoluciónActual.*

---------ELIMINAR LA PEOR VARIABLE (SOLO SI HAY MEJORA)---------

4. Seleccionar la **peor** variable para eliminar de *SolucionActual*. Por cada variable V de *SolucionActual* que no se encuentre en *Eliminados*:
   4.1. *SolucionTemporal* = *SolucionActual* – V
   4.2. Evaluar, mediante evaluación robusta (3.1), *SolucionTemporal* y guardar su rendimiento.
5. Seleccionar la mejor *SoluciónTemporal* (la que proporcione mayor rendimiento). Solo si el rendimiento de la mejor Solución temporal es superior al rendimiento de la mejor solución obtenida en el punto 2, entonces: *SoluciónActual* = *MejorSolucionTemporal*. En este caso, actualizar *Eliminados* añadiendo la variable eliminada.
6. Evaluar condición de parada (se detalla a continuación).

Devolver tabla con cada una de las MejorSolucionTemporal (obtenida al final de cada iteración), el tamaño y el rendimiento de cada una.

Consideraciones:

- *Condición de parada: La condición de para solo puede ser cierta una vez que todas las variables han sido procesadas por el proceso de adición, es decir, Añadidos contiene todas las variables originales. Llegados a este punto, usaremos un contador para contar las iteraciones que transcurren sin que se eliminen variables (recordemos que ya no se podrán añadir más variables). Si en alguna iteración una nueva variable es eliminada, entonces reseteamos el contador a cero. Si el contador alcanza un cierto umbral M (por ejemplo 10 iteraciones) se considera que se cumple la condición de parada y se detiene el algoritmo. Este criterio de parada más complejo (en vez de parar cuando ya todas las variables hayan sido añadidas) nos permitirá obtener soluciones más robustas, ya que el objetivo del umbral de M iteraciones es conseguir una solución estable (garantizar que se han eliminado todas las variables que no son imprescindibles). La razón de esta condición es la misma por la que usamos un procedimiento de evaluación robusta, es decir, la aleatoriedad existente en cada uno de los experimentos realizados.*

3.4. (A.3) Búsqueda secuencial hacia atrás:

Descripción detallada del algoritmo de búsqueda secuencial hacia atrás:

Entrada:

- Conjunto de datos con N variables predictoras y una variable respuesta.
- Otros parámetros que considere necesarios.

Salida:

- Tabla con las combinaciones obtenidas en cada iteración, su tamaño y su rendimiento.

Inicialización:

- *SolucionActual*: Almacena el mejor conjunto de variables obtenido en cada iteración. Inicialmente contiene todas las variables.

Ejecución:

Desde K=N hasta 1:

1. Seleccionar la peor variable para eliminar de *SolucionActual*. Por cada variable V de *SolucionActual*:
   1.1. *SolucionTemporal* = *SolucionActual* - V
   1.2. Evaluar, mediante evaluación robusta (3.1), *SolucionTemporal* y guardar su rendimiento.
2. Seleccionar la mejor *SoluciónTemporal* (la que proporcione mayor rendimiento) y hacer *SolucionActual* = *MejorSolucionTemporal*

Devolver tabla con cada una de las *MejorSolucionTemporal*, el tamaño y el rendimiento de cada una. Es decir, la tabla contendrá K soluciones, cada una de un tamaño, donde la primera será de tamaño 1 y la última de tamaño K.

3.5. (A.4) Búsqueda secuencial hacia atrás mixta:

La descripción proporcionada anteriormente (sección 1.3) podría acabar en un bucle infinito, para evitar esto, es necesario llevar el control de las soluciones ya procesadas. A continuación se propone una versión detallada que realiza un control de variables visitadas (el control también podría realizarse a nivel de soluciones visitadas). Esta solución es una propuesta y no es obligatorio ceñirse a la misma.

Entrada:

- Conjunto de datos con N variables predictoras y una variable respuesta.
- Otros parámetros que considere necesarios.

Salida:

- Tabla con las combinaciones obtenidas en cada iteración, su tamaño y su rendimiento.

Inicialización:

- *SolucionActual*: Almacena el mejor conjunto de variables obtenido en cada iteración. Inicialmente contiene todas las variables.
- *Añadidos*: Almacena las variables que han sido añadidas.
- *Eliminados*: Almacena las variables que han sido eliminadas.

Ejecución:

Mientras que no se cumpla la CondicionDeParada:

---------ELIMINAR LA PEOR VARIABLE---------

1. Seleccionar la **peor** variable para eliminar de *SolucionActual*. Por cada variable V de *SolucionActual* que no esté en *Eliminados*:
   1.1. *SolucionTemporal* = *SolucionActual* - V
   1.2. Evaluar, mediante evaluación robusta (3.1), *SolucionTemporal* y guardar su rendimiento.
2. Seleccionar la mejor *SoluciónTemporal* (la que proporcione mayor rendimiento) y hacer *SoluciónActual* = *MejorSolucionTemporal*
3. Actualizar Eliminados*:* Añadimos a esta lista la variable eliminada de *SoluciónActual.*

---------AÑADIR LA MEJOR VARIABLE (SOLO SI HAY MEJORA)---------

4. Seleccionar la **mejor** variable para añadir a *SolucionActual*. Por cada variable V del conjunto original de variables que no se encuentre en *SolucionActual* ni en *Añadidos*:
   4.1. *SolucionTemporal* = *SolucionActual* + V
   4.2. Evaluar, mediante evaluación robusta (3.1), *SolucionTemporal* y guardar su rendimiento.
5. Seleccionar la mejor *SoluciónTemporal* (la que proporcione mayor rendimiento). Solo si el rendimiento de la mejor Solución temporal es superior al rendimiento de la mejor solución obtenida en el punto 2, entonces: *SoluciónActual* = *MejorSolucionTemporal*. En este caso, actualizar *Añadidos* añadiendo la variable añadida.
6. Evaluar condición de parada (se detalla a continuación).

Devolver tabla con cada una de las MejorSolucionTemporal (obtenida al final de cada iteración), el tamaño y el rendimiento de cada una.

Consideraciones:

- *Condición de parada: La condición de para solo puede ser cierta una vez que todas las variables han sido procesadas por el proceso de eliminación, es decir, Eliminados contiene todas las variables originales. Llegados a este punto, usaremos un contador para contar las iteraciones que transcurren sin que se añadan variables (recordemos que ya no se podrán eliminar variables). Si en alguna iteración una nueva variable es añadida, reseteamos el contador a cero. Si el contador alcanza un cierto umbral M (por ejemplo 10 iteraciones) se considera que se cumple la condición de parada y se detiene el algoritmo. Este criterio de parada más complejo (en vez de parar cuando ya todas las variables hayan sido eliminadas) nos permitirá obtener soluciones más robustas, ya que el objetivo del umbral de M iteraciones es conseguir una solución estable (garantizar que se han añadido todas las variables relevantes). La razón de esta condición es la misma por la que usamos*

un procedimiento de evaluación robusta, es decir, la aleatoriedad existente en cada uno de los experimentos realizados.

3.6. Otras consideraciones:

- *El objetivo final de este trabajo es la implementación del algoritmo de búsqueda secuencial mixta. Se solicita también la implementación de la búsqueda secuencial simple por dos motivos. 1) Al ser la búsqueda simple más sencilla, pero similar a la mixta, nos proporciona un acercamiento gradual al problema, de forma que una vez que se ha implementado la primera búsqueda, la segunda será más fácil de abordar. 2) En caso de no conseguir resultados satisfactorios en la implementación de la búsqueda secuencial mixta, dispondremos la búsqueda simple para realizar experimentos y presentar resultados.*
- *Se proporcionan dos conjuntos de datos en los que se ha comprobado (usando árboles de decisión) que es posible obtener subconjuntos de variables que mejoren el rendimiento proporcionado por el conjunto de variables completo.*
- *La imagen muestra un ejemplo de salida esperada para cualquiera de los dos algoritmos de búsqueda (en los algoritmos A2 y A4 pueden aparecer valores repetidos en la columna size): Cada fila corresponde a cada una de las MejorSolucionTemporal generadas por el algoritmo. La columna* ***solution*** *contiene una lista con los nombres de las variables que contiene esa solución,* ***score*** *contiene una medida de la capacidad predictiva del algoritmo entrenado con esa solución y* ***size*** *contiene el número de variables de la solución. Nótese que el formato de la salida es libre, no hay que imitar la imagen. Use la estructura de datos que considere oportuna (listas de listas, diccionarios, etc.). Lo importante es que la salida contenga los elementos solicitados. Los resultados se presentarán, preferiblemente, en orden descendente, es decir, las mejores soluciones al principio y las peores al final.*

solution	score	size
[Initial, SibSp, Deck, Fare_cat, Title]	0.815761	5
[Initial, SibSp, Deck, Fare_cat, Title, Sex]	0.815761	6
[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.815547	7
[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.809736	8
[Initial, SibSp, Deck, Fare_cat]	0.809470	4
[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.809409	12
[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.808948	11
[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.807754	9
[Initial, SibSp, Deck]	0.807720	3
[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.805330	13
[Initial, SibSp]	0.805218	2
[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.804569	10
[Initial]	0.783354	1
[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.782744	14
[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.768328	15

4. Descripción del trabajo y su evaluación

4.1. Primera convocatoria – Junio de 2024:

Algoritmos de búsqueda a implementar:

- A3) Búsqueda secuencial hacia **atrás**
- A4) Búsqueda secuencial hacia **atrás** mixta

Tipo de tarea: **Clasificación**

Algoritmos de entrenamiento:

- Árboles de decisión de clasificación (DecisionTreeClassifier)[7]
- Otro algoritmo de clasificación a elección de los alumnos

Medida de rendimiento principal:

- **Tasa de aciertos balanceada**[8] (para usarla junto con el método de validación cruzada, basta con establecer el parámetro *scoring='balanced_accuracy')*.

Conjuntos de datos:

- **Pendiente de publicación.**

4.2. Segunda y tercera convocatoria – Julio y noviembre de 2024:

Algoritmos de búsqueda a implementar:

- A1) Búsqueda secuencial hacia **delante**
- A2) Búsqueda secuencial hacia **delante** mixta

Tipo de tarea: **Regresión**

Algoritmos de entrenamiento:

- Árboles de decisión de regresión (DecisionTreeRegressor)[9]
- Otro algoritmo de regresión a elección de los alumnos

Medida de rendimiento principal:

- **R2** - Coeficiente de determinación[10] (para usarla junto con el método de validación cruzada, basta con establecer el parámetro *scoring='r2')*.

Conjuntos de datos:

- **Pendiente de publicación.**

4.3. Presentación del código

El trabajo debe presentarse en forma de cuadernos (notebooks) de jupyter que pueden ser desarrollados tanto con jupyter-notebook como jupyter-lab, ambos disponibles en el paquete

7 https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
8 https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html
9 https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
10 https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

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. Para mejorar la claridad del cuaderno, se recomienda que las clases o funciones desarrolladas se proporcionen en un módulo Python (.py) y se importen para ser usadas en el cuaderno.

4.4. Documentación y entrega

El trabajo deberá documentarse siguiendo un formato de artículo científico, con una extensión mínima de 6 páginas. En la página web de la asignatura se pueden encontrar plantillas donde se sugiere una estructura general. Estas plantillas siguen el formato de los IEEE conference proceedings, cuyo sitio web guía para autores[11] ofrece información más detallada. El documento entregado deberá estar en formato PDF. Se valorará el uso del sistema LaTeX[12].

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).

La entrega del trabajo consistirá de **un único fichero comprimido zip** conteniendo la memoria, el código implementado (algoritmo y experimentos) en forma de cuaderno de Jupyter y los conjuntos de datos necesarios para ejecutar los experimentos. Es importante usar **rutas relativas** para que el profesor pueda ejecutar el cuaderno en otro ordenador sin modificar el código. Todos los resultados obtenidos y mencionados en el documento deben ser reproducibles en el cuaderno de Jupyter.

4.5. 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.

11 https://www.ieee.org/conferences/publishing/templates.html
12 https://es.wikibooks.org/wiki/Manual_de_LaTeXa

5. Criterios de evaluación

Para la evaluación del trabajo se tendrán en cuenta los siguientes criterios, considerando una nota total máxima de 4 puntos:

Código fuente, experimentación y resultados (2 puntos): se valorará la claridad y buen estilo de programación, corrección, eficiencia y usabilidad de la implementación, y calidad de los comentarios. La documentación del código y los experimentos en el cuaderno de Jupyter también se valorará. En ningún caso se evaluará un trabajo con código copiado directamente de Internet o de otros compañeros. Con respecto a la experimentación, se valorará la calidad y completitud de los experimentos realizados. Además, se tendrá en cuenta el rendimiento del algoritmo y el conjunto de mejores parámetros proporcionado. Por último, no se tendrán en cuenta aquellos resultados experimentales que no sean reproducibles.

El documento – artículo científico (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.

La presentación y defensa (1 punto): 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.