*Título del trabajo (elegir uno original)*

Amine Chaghir Chikhaoui   
*dpto. Ciencias de la Computación e Inteligencia Artificial*  
*Universidad de Sevilla*Sevilla, España  
UVUS: amichachi Correo: aminechaghir1999@gmail.com

Álvaro Frías Balbuena  
*dpto. Ciencias de la Computación e Inteligencia Artificial*  
*Universidad de Sevilla*Sevilla, España  
UVUS: alvfribal Correo: alvarofb1998@gmail.com

Este trabajo desarrollado en la asignatura de Inteligencia Artificial se basa en la selección de características en modelos predictivos y forma parte del aprendizaje automático, o más específicamente a la rama del aprendizaje supervisado. La selección de características consiste en que dado un conjunto de datos con unas propiedades (características) que los clasifican, conseguir obtener aquel subconjunto o subconjuntos de características o variables predictoras que mejor ganancia nos proporcionen y por lo tanto las que más información nos aportan sobre la variable respuesta. El objetivo de este proyecto es desarrollar el algoritmo Sequential Floating Forward Selection (SFFS) que estaría dentro del aprendizaje supervisado. Para llegar a realizar este algoritmo, hemos implementado tanto un método de validación cruzada como un algoritmo llamado Sequential Forward Selection que nos servirá de modo introductorio para nuestro algoritmo objetivo.

# Introducción

En la actualidad, la selección de características es uno de los estudios más importantes relacionado con el aprendizaje automático. Este estudio consiste en el proceso de selección de subconjuntos de variables de entre todas las variables predictoras (características) de manera que se escojan aquellas que sean más relevantes en el problema que estemos intentando resolver y por lo tanto así descartaremos algunas de las características que resultan redundantes o irrelevantes. Redundante e irrelevante son dos nociones distintas, ya que una característica irrelevante puede ser redundante en presencia de otra característica relevante con la que está fuertemente unida [1].

La selección de características posee numerosas técnicas que se agrupan en tres clases diferentes, aunque en este trabajo nos centraremos en desarrollar el método de envoltura (Wrapper Methods). Este método nos permite quedarnos con él subconjunto de variables que mejor rendimiento aporte. Las principales desventajas de este método es el riesgo creciente de overfitting cuando el número de datos es insuficiente y el gran tiempo computacional cuando el número de variables es grande [2].

Cabe destacar que este trabajo el subconjunto seleccionado no tiene porqué ser aquel que mejor rendimiento proporcione, ya que buscamos un equilibrio entre el tamaño del subconjunto seleccionado (número de características seleccionadas) y el rendimiento que nos proporciona, es decir si un subconjunto tiene doce características con un rendimiento de 0.81 y otro subconjunto con seis características proporciona un rendimiento de 0.80 entonces el segundo conjunto sería la elección más óptima, ya que con la mitad de características nos proporciona casi la misma ganancia.

En este trabajo se pretende mediante el uso de validación cruzada, métodos de envoltura y estrategias de búsqueda secuenciales hacia delante y mixtas, obtener los subconjuntos de variables con mejor capacidad predictiva. Por lo que primero se ha llevado a cabo el desarrollo de un método de evaluación robusta, que se apoya en validación cruzada y en arboles de decisión, y posteriormente el algoritmo Sequential foward selection (SFS) y por último el algoritmo Sequential floating forward selection (SFFS).

La idea es obtener la capacidad predictiva que tendría un árbol de decisión que sea entrenado con un subconjunto de datos especifico, mediante el método de evaluación robusta y mediante los métodos de Sequential foward selection (SFS) o Sequential floating foward selection (SFFS) obtener una salida con aquellos subconjuntos más prometedores, es decir, los que mayor capacidad predictiva tengan.

La estructura del documento es la siguiente:

Contenido

[I. Introducción 1](#_Toc43578944)

[II. Preliminares 1](#_Toc43578945)

[A. Métodos empleados 2](#_Toc43578946)

[III. Metodología 3](#_Toc43578947)

[IV. Resultados 10](#_Toc43578948)

[V. Anexo 11](#_Toc43578949)

[VI. Conclusiones 14](#_Toc43578950)

# Preliminares

Las principales técnicas utilizadas en el desarrollo de este trabajo son las siguientes:

Métodos de envoltura (wrapper methods), en el que se entrena un modelo con cada uno de los subconjuntos obtenidos y evaluamos su rendimiento para quedarnos con el mejor.

Árboles de decisión, mediante este algoritmo obtenemos un árbol de decisión que va clasificando los datos del subconjunto escogido hasta llegar a las hojas del árbol donde se encuentra el valor de la variable respuesta que usamos para clasificar.

Validación cruzada, se utiliza para evaluar cada subconjunto de variables, obteniendo en nuestro caso su capacidad predictiva. En la validación cruzada de K iteraciones los datos se dividen en K subconjuntos, uno de ellos se utiliza para evaluar (test) y el resto K-1 se utilizan como datos de entrenamiento(train).

Estrategias de búsqueda secuenciales que se dividen en tres grupos hacia delante, es decir, empezamos con un conjunto vacío y vamos añadiendo en cada iteración una nueva característica; hacia atrás, es decir, partimos de un conjunto con todas las variables y vamos quitando una en cada iteración; y la mixta, que combina ambas direcciones, tomando una como principal y la otra como secundaria para corregir cambios no óptimos.

## Métodos empleados

La selección de características posee numerosas técnicas que se agrupan en tres clases diferentes, aunque en este trabajo nos centraremos en desarrollar el método de envoltura (Wrapper Methods). Este método consiste en seleccionar un subconjunto de características, entrenar con él un modelo predictivo, evaluar el rendimiento obtenido y por último de todos los subconjuntos nos quedamos con el que mejor rendimiento aporte. Las principales desventajas de este método es el riesgo creciente de overfitting cuando el número de datos es insuficiente y el gran tiempo computacional cuando el número de variables es grande [2].

Para poder evaluar el subconjunto de variables que vamos obteniendo haremos uso de un método de tipo envoltura. Este método nos ayudará a entrenar y evaluar la capacidad predictiva obtenida. Debemos tener en cuenta que estos algoritmos y métodos tienen un componente de aleatoriedad por lo que el valor de las diferentes variables o conjuntos de variables pueden variar y provoca que aplicando las mismas soluciones los resultados pueden variar levemente. Esto lo solucionaremos gracias al uso de una técnica llamada Validación Cruzada. Esta técnica divide el conjunto de datos en datos de entrenamiento y de prueba y, mediante repetición, va calculando una media de los diferentes resultados que vamos obteniendo en las diferentes divisiones de datos de entrenamiento y de prueba que vamos eligiendo. Gracias a esto podremos conseguir unos resultados mucho más fiables y cercanos a la realidad dado que contendríamos ese factor de aleatoriedad que caracteriza a los métodos de tipo envoltura utilizados para la realización de este proyecto. A continuación, en la Fig.1, se explica de manera gráfica el funcionamiento del método de Validación Cruzada.

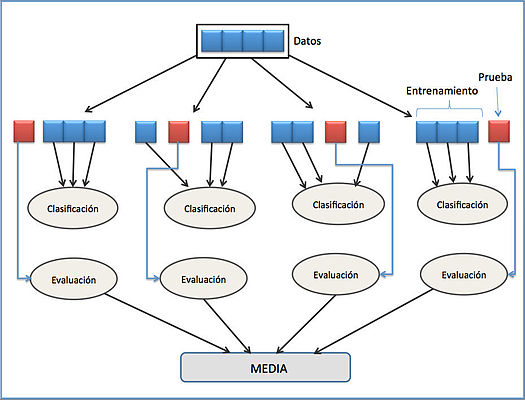


Fig.1. Descripción del funcionamiento de la Validación Cruzada

Como se ve, tenemos un conjunto de datos que, realizando diversas repeticiones, dividimos en distintos conjuntos de datos de entrenamiento(train) y conjunto de datos de prueba(test). A los conjuntos de datos de entrenamiento se le realiza una clasificación según el conjunto de prueba y, una vez terminadas todas las iteraciones se hace una media de los valores obtenidos en las evaluaciones. [3]

El algoritmo de Validación Cruzada se apoya en el algoritmo de árboles de decisión que pertenece al conjunto de técnicas de aprendizaje supervisado. Gracias al uso de este algoritmo seremos capaces de obtener un conjunto de variables denominadas variables de entrenamiento. A medida que va avanzando el algoritmo, vamos añadiendo nuevos nodos que se van distribuyendo en diferentes ramas hasta llegar a las hojas, que tendrán un valor denominado por la variable objetivo. Esta variable objetivo será nuestra variable de estudio, es decir, la intención es buscar los caminos que nos haga obtener más información sobre esta variable. [4]

La finalidad de este proyecto es ser capaces de crear dos estrategias de búsqueda, el Sequential Foward Selection (SFS) y el Sequential Floating Forward Selection (SFFS). Una estrategia de búsqueda es denominada como la realización de una serie de cálculos y procedimientos que un usuario ejecuta para obtener unos resultados que ayudan a la hora de resolver un cierto problema.

El algoritmo de búsqueda Sequential Foward Selection (SFS) consiste en dado un conjunto de datos y un conjunto de variables vacío inicialmente, realizar iteraciones en las cuales, por cada una de ellas, iremos añadiendo una variable del conjunto de datos al conjunto vacío inicial. Esta variable se elige según la ganancia de información que nos proporcione y, para saber la ganancia que nos resulta de una variable o conjunto de variables, nos apoyaremos en la Validación Cruzada anteriormente explicada. Por tanto, añadiremos al inicio la mejor variable que tengamos dentro de nuestro conjunto de datos, es decir, la que mejor ganancia nos dé y, tras esto, buscaremos que variable combinada con la variable o variables que vamos añadiendo al conjunto nos va a proporcionar mejores resultados. Gracias a este método de búsqueda vemos que, aunque una variable individualmente no sea óptima o nos proporcione poca ganancia, combinada con otras variables si sea útil para obtener información relevante de cara a la solución del problema tratado. Este algoritmo presenta un pequeño problema y es que como es un método que se basa en ir “hacia delante”, es decir, iniciar con un conjunto vacío e ir añadiendo iteración a iteración variables que nos vayan trasmitiendo mejores resultados, no estudia si puede haber variables que vayan perdiendo su valor de ganancia de información inicial, en otras palabras, que esa variable ya no sea útil en nuestro conjunto de variables solución dado que con las demás variables sin ella tendríamos una mejor o misma ganancia que con ella. Este problema se soluciona con el algoritmo de Sequential Floating Forward Selection. [4]

El algoritmo de Sequential Floating Forward Selection (SFFS) tiene la misma idea que el anteriormente nombrado solo que implementa un método de “hacia atrás”, es decir, este algoritmo si revisa si una de las variables de nuestro conjunto de soluciones ha quedado en desuso y puede ser eliminada para mejorar o igual la ganancia con menos variables. El funcionamiento de este método es el siguiente: empezando con un conjunto de datos inicial y un conjunto de variables solución vacío, gracias a la Validación Cruzada, va obteniendo las ganancias de las distintas variables del conjunto de datos y vamos añadiendo la mejor de estas al conjunto vacío. Vamos realizando iteraciones y en cada una de ellas volvemos a comprobar que variables combinadas con el conjunto de variables solución que ya tenemos va obteniendo mejores resultados. Además, una vez añadida una variable, mediante un método de eliminación (“hacia atrás”) revisa si podemos eliminar alguna variable que nos haga mejorar o igual el rendimiento actual. Gracias a este método que incorpora este algoritmo, podremos obtener conjuntos de variables solución óptimas que el algoritmo SFS dado que las soluciones serán más reducidas al tener este proceso de eliminación. [4]

Para realizar estos dos algoritmos anteriormente nombrados, hemos tenido que realizar tres métodos auxiliares que ayuden a la correcta realización de estos. Hemos necesitado un método en el que haremos uso de la Validación Cruzada, un método para el cálculo de las variables con mayor ganancia para añadir al conjunto de variables solución, que ha sido implementado tanto en el algoritmo de Sequential Forward Selection (SFS) como en el Sequential Floating Forward Selection (SFFS) y, un método para el cálculo de la variable que peor ganancia nos proporciona a nuestro conjunto de soluciones para el algoritmo de Sequential Floating Forward Selection. La idea de estos métodos son los siguientes:

* El método de validación cruzada nos hará calcular la ganancia de una variable o conjunto de variables. Para ello necesitamos por una parte la variable objetivo, es decir, la variable que vamos a estudiar y de la que queramos sacar información y el conjunto de las demás variables. Haremos uso de la librería sklearn y mediante métodos que implementa esta librería seremos capaces de calcular las ganancias de las variables.
* El método de cálculo de la mejor variable hace uso del método anteriormente nombrado para su funcionamiento. En este método, iremos recorriendo las distintas variables e iremos comparando las ganancias obtenidas de cada una de estas variables junto con las variables solución que ya hayamos elegido. Vamos guardando la variable que mayor ganancia va resultando por cada iteración hasta haber recorrido todas las variables devolviendo finalmente la variable con más ganancia que será añadiendo a nuestro conjunto de variables solución.
* El método de cálculo de peor variable también hace uso del método de validación cruzada para calcular las ganancias de las variables. En este método, una vez realizado el método anterior, recorremos el conjunto de variables solución actual y vamos comprobando si, eliminando una de las variables, mejoramos la ganancia actual. Una vez recorridas todas las variables de este conjunto, nos quedamos con el nuevo conjunto si hemos eliminado alguna variable o con la misma solución inicial en el caso de que no sea capaz de mejorar la ganancia mediante una eliminación.

# Metodología

En esta sección nos encargaremos de hablar detalladamente sobre cada uno de los métodos implementados en el desarrollo del trabajo.

Para empezar, hablaremos del algoritmo de **Validación Cruzada** explicado con anterioridad, mediante el cual obtenemos el rendimiento de los diferentes subconjuntos de variables.

Este algoritmo recibe como entrada el subconjunto de datos, las variables a evaluar, el número de repeticiones del experimento de validación cruzada (N\_Exp) y el número de folds (divisiones) que debemos considerar en la validación cruzada (Cv). [5]

Fig.2. Validación Cruzada

**Procedimiento Validación Cruzada**

**Entrada**:

* Subconjunto de Datos
* Conjunto de variables a evaluar
* N\_Exp (número de repeticiones del experimento)
* Cv (folds de la validación cruzada)

**Salida**:

* Un float equivalente al rendimiento promedio.

**Algoritmo**:

1. Obtener variable resultado, es la última del conjunto de variables originales
2. Obtener conjuntos de entrenamiento, y prueba para aplicar validación cruzada mediante train\_test\_split
   1. variables = conjunto de variables
   2. y = variable resultado
   3. test\_size = tamaño del conjunto de prueba
3. Realizar el experimento de validación cruzada N\_Exp veces

3.1. Obtener el promedio de la validación cruzada

4. Devolver Promedio

En cuanto a la ejecución del algoritmo, lo que se busca es obtener el rendimiento de cada subconjunto tras un número de experimentos.

En nuestro caso, en el método obtenemos primero la variable resultado, y ahora teniendo las variables de la entrada del método, utilizamos el método train\_test\_split del módulo model\_selection de sklearn que divide arrays o matrices en conjuntos aleatorios de entrenamiento(train) y pruebas (tests), para obtener un conjunto de prueba y un conjunto de entrenamiento, en función del conjunto de variables, de la variable resultado (obtenida anteriormente), del tamaño del conjunto de pruebas que en nuestro caso queremos que un 20 por ciento del total del conjunto sea tomado para evaluar el resultado y por lo tanto el parámetro **test\_size** lo igualamos a 0.2 y por último el **random\_state** que controla la combinación aleatoria aplicada al conjunto de datos antes de realizar el Split (“división”) de los datos, en nuestro caso hemos optado por utilizar el valor 123. [6]

Ahora debemos calcular la validación cruzada un número determinado de veces, en nuestro caso este valor nos los proporciona la variable de entrada N\_Exp. La validación cruzada se lleva a cabo usando el método cross\_val\_score que nos proporciona el conjunto de rendimientos calculados en función del valor que tengamos como Cv, es decir, los folds (divisiones) en los que se dividirá el conjunto de todos los datos. El método cross\_val\_score toma como parámetros:

* estimator: es el objeto usado para dividir o clasificar los datos, en nuestro caso utilizaremos árboles de decisión, exactamente, el algoritmo DecisionTreeClassifier.
* X: El conjunto de datos a clasificar
* Y: variables que debemos predecir
* cv: Determina la estrategia de división de los datos al aplicar la validación cruzada, es decir si hacemos cv = 5 se dividirá el conjunto de datos en 5 diferentes.
* scoring: Especifica qué medida se tomará para evaluar el rendimiento o capacidad predictiva de un conjunto, en nuestro caso usaremos la tasa de aciertos balanceada (balanced\_acurracy)

Fig.3 Pseudocódigo calcular mejor variable SFS

En el caso de estimator, al usar DecisionTreeClassifier [7] debemos especificar los argumentos que necesita este algoritmo, en nuestro caso especificamos, max\_depth, que es la máxima profundidad del árbol y debe ser mayor que cero, y el random\_state, utilizamos el valor 123.

El conjunto de datos a clasificar,X, se declara igual al conjunto de entrenamiento X\_train, e “y” como el conjunto y\_train obtenido al realizar el train\_test\_split. El parámetro cv lo declaramos con el valor de Cv tomado como entrada del método y scoring lo declaramos igual a balanced\_acurracy.

Como hemos mencionado anteriormente la profundidad mínima del árbol de decisión es uno ya que debe ser mayor que cero, por esta razón, inicializamos la variable i a 1 y comenzamos el bucle while que termina en N\_Exp más 1 para equilibrar que hemos empezado la i en 1. Durante este bucle se realizan las validaciones cruzada N\_Exp veces, obteniendo una lista con todos los rendimientos medios de los diferentes experimentos y finalmente el rendimiento o capacidad predictiva final como la media de todos los resultados guardados en la lista.

El algoritmo Sequential foward selection (SFS) es una versión simplificada del objetivo de nuestro trabajo que es el SFFS.

Este algoritmo se apoya en una función auxiliar que se encarga de calcular la mejor solución a añadir al conjunto que llevamos acumulado como solución actual, en función del rendimiento que tendría el nuevo conjunto con esa variable, de esta manera elegimos la mejor variable a añadir en cada iteración.

La función auxiliar se llama calcular\_mejor\_variable y su pseudocódigo es el siguiente:

**Calcular mejor variable**

**Entrada**:

* Conjunto de datos
* Conjunto de variables (sin la variable predictora)
* Variable resultado o variable objetivo
* Conjunto de solución actual.

**Salida**:

* La mejor variable para añadir a la solución actual.

**Algoritmo**:

1. Obtenemos una solución temporal como la copia de la solución actual
2. Comenzamos un bucle recorriendo todas las posibles variables para añadir
   1. Escogemos una variable
   2. La añadimos a la solución temporal
   3. Calculamos el rendimiento o capacidad predictiva de la nueva solución temporal
   4. Nos quedamos con la que mayor rendimiento tenga
   5. Guardamos la variable añadida en esa iteración como la mejor variable.
   6. Eliminamos la variable de la solución temporal.
3. Devolvemos la mejor variable

Este método ha sido desarrollado para una mayor limpieza de código, por esta razón al calcular la mejor variable para añadir a la solución actual en método auxiliar nuestro método principal consta de menos líneas de código.

Recibiendo como entrada el conjunto de datos, el conjunto de variables (sin la variable resultado), la variable resultado y el conjunto de solución actual.

La ejecución del método funciona tal que primero declaramos una solución temporal que es igual a la solución actual que traemos acumulada durante. A continuación, nos declaramos un bucle for para recorrer todas y cada una de las posibles variables a añadir que no estén en la solución actual y nos quedamos con aquella cuya capacidad predictiva sea la mayor. Por último, eliminamos la variable procesada de la solución temporal para poder añadir una variable nueva que procesar.

Cuando se procesen todas las variables devolvemos aquella que mejor capacidad predictiva o rendimiento haya tenido al ser añadida a la solución actual.

Una vez explicado el método auxiliar en el que se apoya el algoritmo Sequential Foward Selection procederemos a la explicación de la metodología y decisiones tomadas en el desarrollo de éste.

El pseudocódigo es el siguiente:

**Sequential Forward Selection**

**Entrada**:

* Conjunto de datos
* Variable resultado o variable objetivo
* D número de variables a probar.

**Salida**:

* Tabla con cada una de las soluciones seleccionadas.

**Inicialización**:

1. Inicializar conjunto vacío solución actual, donde se irá guardando la solución.
2. Contador K = 0 para contar las iteraciones.

**Algoritmo**:

1. While K sea menor que D
   1. Seleccionamos la mejor variable para añadir a solución actual, de entre todas las variables que no se encuentren ya en solución actual

[**Método auxiliar ‘calcular\_mejor\_variable’**]

* 1. Calculamos su ganancia
  2. La añadimos a la solución
  3. Vamos añadiendo cada solución actual a una lista de soluciones
  4. Vamos añadiendo cada ganancia a una lista de ganancias
  5. Vamos insertando en un diccionario las entradas {rendimiento: solución\_actual}

1. Devolvemos una tabla con todos los conjuntos con su tamaño y ganancia, y una tabla en formato html.

Este algoritmo recibe como parámetro de entrada el conjunto de datos con todas las variables predictoras y una variable respuesta u objetivo y un parámetro D que indica el número máximo de variables a probar, en caso de que no se proporcione D, el número máximo a probar será el total de todas las variables. Como salida este algoritmo devuelve en nuestro caso cada una de las soluciones obtenidas con su capacidad predictiva y su tamaño.

Antes de comenzar con la ejecución, realizamos un proceso de inicialización en el cual inicializamos la variable K=0 que será el contador de las iteraciones realizadas, un conjunto vacío que será donde este la solución actual, un conjunto vacío ‘Lista’ en el que iremos guardando la solución actual de cada iteración y un diccionario en el cual guardaremos los pares rendimiento solución para poder mostrarlos en una tabla.

En cuanto a la ejecución, declaramos un bucle para realizar las iteraciones desde K hasta el valor de D que tengamos, en el caso de no se especifique el valor de D este será el conjunto de todas las variables. Por cada iteración elegimos la mejor variable para añadir al conjunto mediante el método calcular\_mejor\_variable(), eliminamos la variable seleccionada del conjunto inicial de variables para que el algoritmo no pueda volver a procesarla y así nos evitamos repeticiones, a continuación añadimos esta variable al conjunto de solución actual, calculamos su rendimiento y guardamos la solución en la Lista creada en la inicialización e introducimos en el diccionario el par clave: rendimiento y valor: solución de la lista correspondiente a esa iteración ( se puede obtener pidiendo de la lista la solución que se encuentre en la posición K que es el número de iteración hasta el momento). Incrementamos el valor del contador para la siguiente iteración.

Una vez finalizado el bucle, mediante un método auxiliar que se explicara a continuación obtenemos una tabla en html a la que llamamos ‘archivo\_sfs’ y una tabla para imprimir por consola que son las dos salidas de nuestro algoritmo.

El **algoritmo Sequential floating foward selection (SFFS)** [4] este es el algoritmo objetivo de nuestro trabajo, y es una evolución del método implementado anteriormente, el **Sequential foward selection (SFS).** La diferencia entre este método y el anterior es que en este no solo se va añadiendo variables, sino que también buscamos eliminar alguna variable si con eso mejoramos la capacidad predictiva del resultado.

Este método también se apoya en dos métodos auxiliares que explicaremos antes de explicar las decisiones tomadas con el SFFS, uno de los métodos es el que usamos para calcular la mejor variable y otro con el que calculamos la peor variable para eliminar.

Procedemos a explicar el método usado para calcular la mejor variable. Este método recibe como entrada, los datos, el conjunto de variables a probar, la variable resultado, el conjunto con la solución actual, el conjunto de añadidas.

Y como salida obtenemos, la mejor variable, la solución temporal con la variable añadida, y el rendimiento o capacidad predictiva que tiene.

En la ejecución de este algoritmo mientras se cumpla que el contador i es menor que el tamaño del conjunto de variables, vamos seleccionando variables, comprobamos si esta variable está contenida en el conjunto de solución actual o en el conjunto de variables añadidas. Si no está contenida entonces comprobamos el rendimiento que tiene el añadirla al conjunto de solución actual. Esto se hace añadiendo la variable a la solución temporal y evaluando su rendimiento o capacidad predictiva con la validación cruzada, Si el valor obtenido es mayor que el valor calculado de las variables anteriores y guardado en ac, entonces guardamos esa variable como la mejor variable, actualizamos ac para que sea igual a este nuevo rendimiento que es el más grande hasta el momento, se elimina la variable de la solución temporal para tener el conjunto de nuevo para probar otra variable y se incrementa el índice para la siguiente iteración. Si el rendimiento no es mayor que el que tenemos en ac entonces eliminamos la variable de la solución temporal e incrementamos el índice hacia la siguiente iteración.

Fig.4. Pseudocódigo Sequential Forward Selection

Por otro lado, en el caso de que la variable ya esté contenida en la solución actual o en el conjunto de variables añadidas entonces directamente solo incrementamos el índice hacia la siguiente iteración.

Por último, a la solución temporal le añadimos la mejor variable calculada y devolvemos la mejor variable, la solución temporal y el rendimiento.

A continuación, se muestra el pseudocódigo del método:

Fig.6. Pseudocódigo calcular peor variable SFFS

**Calcular peor variable**

**Entrada**:

* Conjunto de datos.
* Variable resultado.
* Conjunto de solución actual.
* Conjunto de variables eliminadas

**Salida**:

* La peor variable, el rendimiento del conjunto sin la peor variable, solución sin la peor variable.

**Inicialización:**

1. Un rendimiento inicial a 0 (rendimiento\_res)
2. Un contador i a 0
3. Una copia de la solución actual guardada en la solución temporal.

**Algoritmo:**

1. Mientras el contador i sea menor que el tamaño de las variables.
2. Obtenemos la variable i del conjunto de variable
3. Si la variable no está contenida en el conjunto de eliminadas.
   1. Eliminamos la variable a la solución temporal
   2. Calculamos el rendimiento que tendría con la validación cruzada
   3. Si el rendimiento es mayor que el rendimiento inicial (rendimiento\_res).
      1. Peor variable es la variable actual.
      2. Actualizamos el rendimiento inicial con el rendimiento actual.
      3. Incrementamos el contador i
   4. Si el rendimiento es menor.
      1. Incrementamos el contador i
   5. Volvemos a dejar la solución temporal como una copia de la solución actual recibida, para la siguiente iteración.
4. Si la variable está contenida en el conjunto de eliminadas.
   1. Incrementamos el contador i
5. Eliminamos de la solución temporal la peor variable obtenida
6. Devolvemos la peor variable, el rendimiento y la solución temporal.

**Calcular mejor variable**

**Entrada**:

* Conjunto de datos
* Conjunto de variables (sin la variable predictora)
* Variable resultado o variable objetivo
* Conjunto de solución actual.
* Conjunto de variables añadidas

**Salida**:

* La mejor variable para añadir a la solución actual, solución con la variable añadida y el rendimiento.

**Inicialización:**

1. Un acumulador ac a 0
2. Un contador i a 0
3. Una copia de la solución actual guardada en la solución temporal.

**Algoritmo**:

1. Mientras el contador i sea menor que el tamaño de las variables.
2. Obtenemos la variable i del conjunto de variable
3. Si la variable no está contenida en solución actual ni está contenida en el conjunto de añadidas
   1. Añadimos la variable a la solución temporal
   2. Calculamos el rendimiento que tendría con la validación cruzada
   3. Si el rendimiento es mayor que el acumulador que llevamos que es el máximo rendimiento hasta el momento
      1. Mejor variable es la variable actual
      2. Actualizamos el acumulador ac con el rendimiento actual
      3. Eliminamos la variable de la solución temporal
      4. Incrementamos el contador i
   4. Si el rendimiento es menos
      1. Incrementamos el contador i
      2. Eliminamos la variable de la solución temporal
4. Si la variable está contenida en la solución actual y en el conjunto de añadidas
   1. Incrementamos el contador i
5. Añadimos a la solución temporal la mejor variable obtenida
6. Devolvemos la mejor variable, la solución temporal y el rendimiento

Ahora procederemos a explicar el otro método auxiliar, el que empleamos para calcular la peor variable de la solución actual que llevamos hasta el momento, para comprobar si eliminando esta variable mejoramos en algo el rendimiento de la solución actual.

Este método recibe como entrada el conjunto de los datos, la variable resultado, el conjunto de solución actual y el conjunto de variables eliminadas.

Como salida obtenemos la peor variable, el rendimiento de la solución temporal sin esa variable y la solución temporal.

Fig.5. Pseudocódigo calcular mejor variable SFFS

En cuanto a las decisiones tomadas en el desarrollo de este método son muy parecidas a las tomadas para calcular la mejor variable. Para empezar, inicializamos una variable llamada rendimiento\_res a cero y representará el rendimiento que devolveremos al procesar todas las variables y actualizarlo con el rendimiento resultante de eliminar la peor variable, por otro lado, un contador i a cero para controlar las iteraciones hasta procesar todo el conjunto de variables actuales y una copia de la solución actual, llamada solución temporal.

Ahora que tenemos todo inicializado, comenzamos un bucle en el que entraremos siempre que el contador i sea menor que el tamaño de la solución actual menos uno. Una vez dentro del bucle seleccionamos la variable i y comprobamos si está contenida o no en el conjunto de variables ya eliminadas. En el caso de que no esté contenida, entonces eliminamos la variable de la solución temporal, calculamos el rendimiento o capacidad predictiva de esa solución temporal. Si este rendimiento es mayor que el rendimiento\_res que hemos inicializado al principio, entonces guardamos la peor variable como la variable actual, actualizamos ese rendimiento\_res con el rendimiento de esta solución temporal, (ahora en el rendimiento\_res es el mayor rendimiento) e incrementamos el índice i, para la siguiente iteración. En el caso de que el rendimiento de esta solución temporal sea menor que el rendimiento\_res, entonces solo incrementamos el índice i para ir a la siguiente iteración. Además, también eliminamos la peor variable de la solución temporal para dejarla como al principio y preparada para probar la siguiente variable, de igual manera en el caso de que la variable se encuentra ya en el conjunto de eliminadas entonces también solo incrementamos el índice i.

Tras salir del bucle, ya tenemos la peor variable y el rendimiento, entonces actualizamos la solución temporal quitándole la peor variable. Por último, devolvemos la peor variable, el rendimiento y la solución temporal.

Ahora que hemos explicado los dos métodos auxiliares necesarios para el correcto desarrollo del algoritmo **Sequential floating foward selection (SFFS)** procederemos a la explicación de la metodología seguida en su desarrollo.

Como entrada este método recibe el conjunto de datos y la variable predictora y como salida obtenemos una tabla para imprimir por consola y una tabla en formato html (esto se consigue con dos métodos auxiliares que explicaremos más adelante).

En cuando al desarrollo del algoritmo, comenzamos con la parte de inicialización, en esta parte obtenemos el conjunto de variables en una lista extraída del conjunto de todos los datos del fichero. Eliminamos la variable resultado recibida como entrada del conjunto de variables para no procesarla cuando comencemos a buscar la mejor solución, ya que es la variable que pretendemos evaluar. Además, nos declaramos un conjunto vacío donde se irá guardando la solución actual al que llamaremos solución\_actual, un conjunto vacío para guardar las variables que vamos añadiendo y procesando, un conjunto vacío donde vamos añadiendo las variables eliminadas durante la ejecución del algoritmo, un diccionario donde guardaremos el rendimiento(clave) y la solución actual(valor) para poder mostrar la salida en una tabla con todas las soluciones, una variable k a cero para contabilizar las iteraciones que llevamos, un conjunto vacío llamado lista\_solucion donde guardar las soluciones y un conjunto vacío llamado lista\_ganancia donde ir guardando todas las ganancias.

Mientras el tamaño de las variables añadidas sea menor que el tamaño del conjunto de variables, es decir, que aún no se han añadido y procesado todas las variables. Entramos en el bucle de ejecución del algoritmo. Primero utilizamos el método para calcular la mejor variable para añadir al conjunto de solución actual, según hemos explicado anteriormente este método nos devuelve la mejor variable, la mejor solución temporal y su rendimiento, entonces por ahora igualamos la solución actual con el valor de la mejor solución temporal obtenida del método calcular mejor variable y añadimos la mejor variable al conjunto de variables que ya han sido procesadas y añadidas al conjunto de solución actual. En este punto si el tamaño de la solución actual es menor o igual que dos no tiene sentido eliminar una variable.

Esto se debe a que la primera variable que elegimos es la mejor de todo el conjunto, es decir, la que mayor capacidad predictiva (rendimiento) tiene, por lo tanto eliminarla no tiene sentido, y en el caso de que tengamos dos variables en el conjunto, la segunda variable que se añade es la variable que junto con la variable inicial mejor capacidad predictiva aporta, y es mayor que la capacidad predictiva de la primera variable sola, por tanto si la eliminamos nos quedamos con la primera variable, que tiene peor capacidad predictiva y esto no mejora nuestra solución y en el caso de que se elimine la primera variable, no tendría sentido ya que supuestamente esta variable es mejor que la segunda ya que fue la primera en ser elegida y por tanto eliminarla también empeoraría el resultado.

Por lo que en el caso de que el tamaño de la solución actual no sea mayo que dos, solo añadimos la solución actual a la lista de soluciones actuales, la ganancia actual a la lista de ganancias y añadimos al diccionario resultado el par de valores rendimiento: solución actual e incrementamos el contador k que usaremos para aplicar una condición de parada.

Por otro lado, si el tamaño del conjunto de solución actual es mayor que dos entonces al añadir la tercera variable si procedemos con el proceso de eliminación. En este caso hacemos uso del método explicado anteriormente ‘calcular por variable’, a este método le pasamos el conjunto de datos, el conjunto de todas las variables, la variable resultado, el conjunto de la solución actual, y el conjunto de variables añadidas y obtenemos como salida la peor variable de esta solución actual, el rendimiento que tiene y la solución temporal sin esa peor variable.

Una vez tenemos estas variables, comprobamos si ese rendimiento sin la peor variable es mayor o igual que el rendimiento de la solución actual con todas las variables. En caso de que lo sea, entonces actualizamos el contado k a cero de nuevo, actualizamos el rendimiento con el nuevo rendimiento sin la peor variable, añadimos la peor variable al conjunto de variables ya eliminadas para que no se vuelva a intentar eliminarla, y añadimos al diccionario el par de valores rendimiento: solución actual, además añadimos la solución actual a la lista de soluciones y el rendimiento a la lista de ganancias.

En el caso de que el rendimiento sin la peor variable no sea mayor que el rendimiento de la solución actual, entonces nos quedamos con esa solución actual, la guardamos en la lista de soluciones, guardamos su rendimiento en la lista de ganancias, introducimos en el diccionario el par de valores rendimiento: solución actual e incrementamos el contador k en una unidad.

**Sequential floating forward selection: [cont]**

15. [cont]

c. Si el nuevo rendimiento es mayor que el rendimiento de la solución actual

* + 1. Contador k = 0
    2. Solucion\_actual = solucion\_temporal (sin la peor variable)
    3. Actualizar rendimiento
    4. Eliminadas + peor variable
    5. Al diccionario de soluciones finales añadimos el par rendimiento: solución actual.
    6. Añadimos solucion a lista de soluciones
    7. Añadimos rendimiento a lista de ganancias.

1. Si el nuevo rendimiento no es mayor que el rendimiento de la solución actual.
   1. Guardamos la solución actual en la lista de solucion
   2. Guardamos el rendimiento en la lista de ganancias
   3. Añadimos al diccionario de soluciones finales el par de valores rendimiento: solución
   4. Incrementamos el contador k en uno

16. Si el contador k es mayor o igual a 10

a. Nos salimos del programa

17. Con un bucle for recorremos la lista de todas las soluciones procesadas

a. Guardamos en un diccionario cada solución con su rendimiento

18. Mediante el método auxiliar imprimir datos en html, pasándole el diccionario con todas soluciones obtenemos un archivo con la tabla en formato html

19. Mediante el método auxiliar mostrar datos ordenados pasándole el diccionario con las soluciones finales obtenemos la tabla resultado.

20. Devolvemos la tabla html y la tabla resultado.

El contador k se encarga de contar las iteraciones que lleva el algoritmo realizadas. Lo utilizamos para aplicar una condición de parada de tal manera que si el algoritmo lleva 10 iteraciones sin eliminar ninguna variable el algoritmo directamente para. Si en alguna iteración el algoritmo elimina una variable el contador se pone a cero y comienza de nuevo a contar las iteraciones.

Por esta razón en nuestro algoritmo realizamos la comprobación de si el valor k ha igualado o superado el valor 10, en ese caso se muestra un mensaje notificando que se ha dado la condición de parada y el algoritmo sale del bucle con los datos que tenga hasta ese momento.

Fig.7. Pseudocódigo Sequential Floating Forward Selection

**Sequential floating forward selection**

**Entrada**:

* Conjunto de datos.
* Variable resultado.

**Salida**:

* Tabla con las soluciones finales y una tabla en html con todo el conjunto de soluciones en cada iteración.

**Inicialización:**

1. Lista con todas las variables excepto la variable resultado
2. Conjunto solución actual vacío
3. Conjunto de variables añadidas vacío
4. Conjunto de variables eliminadas vacío
5. Un contador k a 0 para contar las iteraciones
6. Diccionario para las soluciones finales vacío
7. Diccionario para todas las soluciones procesadas vacío.
8. Conjunto vacío, donde tener una lista de las soluciones.
9. Conjunto vacío donde tener una lista de las ganancias.

**Algoritmo:**

1. Mientras el tamaño de las variables añadidas sea menor que el tamaño del conjunto de las variables.
2. Llamamos al método calcular mejor variable y obtenemos la mejor variable, solución temporal con esa variable y el rendimiento
3. Solución actual = solución temporal
4. Añadidas + mejor variable
5. Si el tamaño del conjunto de soluciones actuales es menor o igual que dos.
   1. Añadimos la solución actual a la lista de soluciones
   2. Añadimos el rendimiento a la lista de ganancias
   3. Añadimos al diccionario de soluciones finales el par rendimiento: solución
   4. Incrementamos el índice k.
6. Si el tamaño del conjunto de soluciones actuales es mayor que dos.
   1. Llamamos al método calcular peor variable
   2. Obtenemos la peor variable, la solución temporal sin esa variable y el nuevo rendimiento.

Para facilitar el entendimiento de la ejecución del código se han colocado algunos prints que van mostrando la traza de ejecución. Por otro lado, fuera del bucle, mostramos la lista de variables añadidas y la lista de variables eliminadas.

Por último, para poder obtener las salidas de nuestro algoritmo, es decir, la tabla con las soluciones finales y la tabla en html con todas las soluciones obtenidas de iterar en el bucle, necesitamos dos diccionarios uno con los datos de las soluciones

finales y otro con los datos de todas las soluciones obtenidas. El primero ya lo tenemos durante la ejecución del algoritmo, y el segundo lo obtenemos mediante un for recorriendo la lista de soluciones. En este for imprimimos cada solución con su ganancia y du tamaño y además añadimos al diccionario cada uno de los pares de valores rendimiento: solución.

Ahora que tenemos los dos diccionarios emplearemos dos métodos auxiliares creados en la clase imprimir\_datos\_ordenados.py. Uno de estos métodos toma el diccionario con todas las soluciones procesadas y nos devuelve una tabla con todas ellas en formato html (el método mostrar\_datos\_en\_html) y el otro toma el diccionario con las soluciones guardadas y nos muestras una tabla para mostrar en consola (el método datos\_ordenados).

Una vez tenemos las dos tablas las devolvemos en el return del método.

Anteriormente en la explicación del algoritmo Sequential Floating Forward Selection, se han mencionado dos métodos auxiliares utilizados para obtener una tabla en html y una tabla normal para imprimir por consola con las soluciones.

El primero de ellos es el método imprimir datos en html que recibe como entrada el conjunto de datos y devuelve como salida una tabla en html con los datos.

El pseudocódigo de este método es el siguiente: [8]

**Mostrar datos en html**

**Entrada**:

* Diccionario con los datos

**Salida**:

* Archivo HTML con una tabla con los datos.

**Algoritmo**:

1. Declarar un conjunto con un primer conjunto con los nombres de las columnas
2. Obtener las claves del diccionario
3. Por cada clave del conjunto de claves del diccionario
   1. Calcular el valor asociado
   2. Calcular el tamaño
   3. Concatenar estos valores con el array inicializado al principio con el nombre de las columnas
4. Una vez procesadas todas las claves. Obtener un DataFrame con todos los datos concatenados en el array.
5. Devolver ese DataFrame convertido en HTML

Fig.8. Pseudocódigo mostrar datos en tabla html

Inicializamos un conjunto en que se empezaremos con un conjunto con los nombres de las columnas en su interior.

Obtendremos el conjunto de todas las claves contenidas en el diccionario, en nuestro caso, el diccionario tiene como claves los rendimientos y como valor las soluciones, entonces obtendremos todas las claves y mediante un bucle las recorremos todas. Como nuestro objetivo es mostrar la solución, el tamaño y el rendimiento por cada clave (rendimiento) debemos obtener su valor (lista con la solución) y el tamaño del valor es decir el tamaño de la lista de la solución. Una vez tengamos todo esto lo concatenamos con el conjunto inicializado al principio en el mismo orden, primero solución, luego tamaño y por último el valor. Vamos concatenando por cada iteración hasta que procesemos todas las variables, entonces lo pasamos por un DataFrame de pandas para obtener la tabla y lo devolvemos como un archivo html.

El otro método es el método mostrar datos ordenados, este método recibe como entrada el diccionario con las soluciones finales y devuelve una tabla ordenada por rendimiento sin elementos repetidos mostrando tres columnas Solución, Tamaño y Rendimiento.

Para empezar, se ordena el diccionario que llega como entrada, por su clave en este caso por el rendimiento. Una vez ordenado se inicializa un conjunto nuevo llamado lista\_final vacío y un conjunto llamado lista\_valores con una copia del diccionario ordenado entero. A continuación, mediante un bucle recorremos cada conjunto de soluciones que hay en la lista\_valores (No olvidar que venimos de un diccionario por tanto debemos tener en cuenta que el elemento[0] es el rendimiento y elemento[1] es el conjunto con la solución, que es lo que nos interesa a nosotros ahora), seleccionamos un elemento si no está en la lista\_final lo añadimos y si ya está en la lista final lo borramos del diccionario.

Ahora que tenemos el diccionario ordenado y sin repeticiones, nos creamos un conjunto llamado df en el cual insertamos un primer conjunto con las columnas que queremos imprimir en la tabla y comenzamos a recorrer el diccionario ordenado. Por cada elemento del diccionario lo concatenamos con el conjunto df que ya nos hemos creado y lo guardamos y así hasta que los recorramos todos.

Por último, pasamos el conjunto por un DataFrame de pandas para obtener la tabla y lo devolvemos.

Fig. 9 Mostrar tabla con los datos ordenados

**Mostrar datos ordenados**

**Entrada**:

* Diccionario con los datos

**Salida**:

* Tabla con los datos ordenados.

**Inicialización:**

* Conjunto vacío para las soluciones no repetidas
* Conjunto con una copia del diccionario ordenado

**Algoritmo**:

1. Ordenar el diccionario recibido por la entrada en función del rendimiento (clave)
2. Mientras i este dentro del rango 0-tamaño copia del diccionario
   1. Escoger un elemento de la copia del diccionario
   2. Si el elemento no está en el conjunto que inicializamos vacío
      1. Se añade el elemento
   3. Si está contenido en el conjunto
      1. Se elimina del diccionario ordenado
3. Declarar un conjunto con un primer conjunto con los nombres de las columnas
4. Por cada elemento del conjunto del diccionario ordenado
   1. Concatenar el conjunto inicializado anteriormente con uno nuevo que contenga los campos del diccionario en el mismo orden que las columnas, por ejemplo, Mejor Solución, Tamaño, Rendimiento
   2. Guardar conjunto
5. Una vez procesadas todas las claves. Obtener un DataFrame con todos los datos concatenados en el conjunto.
6. Devolver ese DataFrame.

Además, hemos creado una pequeña interfaz para que se pueda elegir en cada ejecución que archivo se quiere probar y sobre qué algoritmo y en función de esas elecciones se muestran unos datos u otros o se generan unos archivos html con las tablas u otras.

Este archivo se encuentra en archivo\_a\_ejecutar.py. Este archivo viene con indicación que hacen super sencillo ejecutar los algoritmos. Lo primero que hacemos es que el usuario introduzca mediante dos inputs el algoritmo que desea ejecutar y el nombre del archivo sobre el que desea que se ejecute y en función de estas decisiones controlamos que se ejecuta con algunas condiciones que tenemos implementadas. En el caso de que el usuario introduzca mal alguno de los campos el programa termina.

En cada caso generalmente se llama al algoritmo que devuelve dos tablas una normal para imprimir por consola y un archivo HTML con la tabla, el archivo lo escribimos en un archivo de la carpeta en formato HTML para que podamos abrirlo y verlo y la tabla directamente la imprimimos por consola con toda la traza que tenga el algoritmo (en el caso de SFFS).

# Resultados

En esta sección detallaremos los diferentes resultados que hemos obtenido para los dos conjuntos de datos proporcionados para la realización del algoritmo Sequential Floating Forwar Selection (SFFS):

* Para el conjunto de datos de titanic.csv, y con la configuración de N\_Exp=15, CV=10, random\_state=123 y test\_size=0’2 en validación cruzada, nos da los siguientes resultados:

|  |  |  |
| --- | --- | --- |
| MEJOR SOLUCION | TAMAÑO | RENDIMIENTO |
| ['Initial', 'SibSp', 'Deck', 'Fare\_cat', 'Alone', 'Sex'] | 6 | 0.806923 |
| ['Initial', 'SibSp', 'Deck', 'Fare\_cat', 'Alone'] | 5 | 0.806804 |
| ['Initial', 'SibSp', 'Deck', 'Fare\_cat'] | 4 | 0.806389 |
| ['Initial', 'SibSp', 'Deck'] | 3 | 0.802649 |
| ['Initial', 'SibSp'] | 2 | 0.799869 |
| ['Initial'] | 1 | 0.782526 |

Viendo los resultados obtenidos vemos que nos dan resultados desde tamaño 1 a tamaño 6 ordenados por su rendimiento. El subconjunto con tamaño 6 es la que mejor capacidad predictiva nos ofrece respecto a nuestra variable objetivo de nuestro problema. El peor caso obtenido es el subconjunto de datos de tamaño 1 que es el que nos ofrece la peor capacidad predictiva. Como el objetivo de nuestro algoritmo es encontrar el subconjunto de variables que nos proporcione mayor capacidad predictiva, pero teniendo en cuenta también el tamaño del subconjunto, creemos que el subconjunto óptimo para nuestro problema sería el subconjunto de tamaño 4 dado que, aunque añadamos variables, mejoramos nuestra capacidad es un aumento muy leve.

Los resultados obtenidos en cada experimento, explicando en cada caso lo que se ha conseguido.

* Respecto al otro conjunto de datos que se nos ha dado, el archivo BreastCancerDataset.csv, hemos obtenido los siguientes resultados con esta configuración en la validación cruzada N\_Exp=15, Cv=10, random\_State=123 y test\_size=0’2.

|  |  |  |
| --- | --- | --- |
| MEJOR SOLUCION | TAMAÑO | RENDIMIENTO |
| ['mean concave points', 'worst texture', 'worst radius', 'worst compactness', 'smoothness error', 'radius error', 'worst area'] | 7 | 0.953945 |
| ['mean concave points', 'worst texture', 'worst radius', 'worst compactness', 'smoothness error', 'radius error'] | 6 | 0.94897 |
| ['mean concave points', 'worst texture', 'worst radius', 'worst compactness', 'smoothness error'] | 5 | 0.945817 |
| ['mean concave points', 'worst texture', 'worst radius', 'worst compactness'] | 4 | 0.945289 |
| ['mean concave points', 'worst texture', 'worst radius'] | 3 | 0.941734 |
| ['mean concave points', 'worst texture'] | 2 | 0.914102 |
| ['mean concave points'] | 1 | 0.878079 |

Viendo los resultados al ejecutar nuestro algoritmo, podemos ver que los valores de capacidad predictiva de los diferentes subconjuntos son bastantes parecidos por lo que nos es difícil elegir cuál es el subconjunto que nos ofrezca tanto buena capacidad predictiva como tenga un tamaño razonable. Según nuestras observaciones, creemos que el subconjunto con tamaño 4 puede ser el mejor subconjunto dado que al añadir una variable no obtenemos una ganancia remarcable y si existe un pequeño cambio entre el subconjunto con tamaño 3 al subconjunto elegido de tamaño 4.

# Anexo

Como hemos hablado durante algunos de los anteriores apartados, los métodos de selección de características son procesos capaces de seleccionar los datos más importantes de un conjunto de datos para así evitar trabajar con datos que no son útiles, obtener información relevante sobre el estudio de nuestro problema y mejorar tanto el rendimiento como la fluidez del tratamiento de nuestro conjunto de datos que nos hará conseguir mejores resultados respecto a la variable objetivo de nuestro problema.

Dentro del conjunto de métodos de selección de características se diferencian dos grupos: los no supervisados, que son los procesos que no tienen en cuenta la variable objetivo de nuestro problema para su realización, y los supervisados, que son los procedimientos qué si trabajan teniendo en cuenta esta variable objetivo. En este proyecto nos centraremos en los métodos de selección de variables supervisados. Este tipo de método de selección de variables cuenta con tres tipos diferentes: envoltura, filtrado e integrados.

Los métodos de selección de envoltura son los procedimientos que, usando un algoritmo de aprendizaje automático, van evaluando diversos subconjuntos de características del conjunto de datos inicial y van seleccionado los modelos que vayan dando mayor ganancia de información. La intención de estos métodos es llegar a encontrar la combinación de características que nos proporcione la mayor ganancia de información y mejor rendimiento. Los métodos de envoltura siguen una serie de pasos para su utilización: buscan un subconjunto de características del conjunto de datos inicial y con un modelo de aprendizaje automático va entrenando estos subconjuntos de variables que vayamos obteniendo. Una vez entrenados, evalúan el rendimiento del modelo y repite los pasos citados hasta que cumplamos una condición de parada especificada por nosotros según los resultados que queramos obtener en nuestro método. Una vez llegado este punto, obtendremos el subconjunto de variables que nos proporciona el mejor resultado respecto a nuestra característica objetivo de nuestro problema. [9]

Imagen que contiene tarjeta de presentación, cd

Descripción generada automáticamente

Fig.2 Método de selección de características por envoltura

Dentro de nuestro método de selección de características por envoltura, existen distintos métodos de búsqueda:

* Selección “hacia adelante” (SFS): este método empieza con un conjunto vacío de características solución y a este conjunto le añadimos al inicio la variable que mejor ganancia nos proporcione individualmente. Una vez hecho esto, en las siguientes iteraciones iremos comparando las variables aún no incluidas con las variables del conjunto de variables solución y nos quedaremos con la combinación que mejor resultado nos dé. Este proceso se repite reiteradas veces hasta cumplir la condición de parada.
* Selección “hacia atrás” (SBS): empezaremos con un conjunto con todas las características que posee nuestro problema e iremos eliminando la característica que menos ganancia de información nos ofrezca de nuestro conjunto. Se repetirá hasta cumplir la condición de parada que hemos establecido.
* Selección exhaustiva de funciones: en este tipo de método de búsqueda, calcularemos todos los subconjuntos posibles de nuestro conjunto de datos inicial, entrenamos cada subconjunto y nos quedamos con el mejor subconjunto según su rendimiento. La característica de este tipo de método es que los parámetros que modificaremos según nuestro fin del problema es el número mínimo o máximo de características que deben de tener los subconjuntos solución elegidos.
* Limitaciones de avance/retroceso: los dos primeros métodos explicados presentan un problema y es que, en el caso del método “hacia adelante” (SFS) no estudia si una de las variables que hemos añadido por su buen rendimiento en combinación con las otras en algún punto de nuestro problema puede que ya no nos sea útil y que al eliminarla se mejore o se iguale el rendimiento en ese punto y, en el caso del método “hacia atrás” (SBS), no estudia si una de las variables que hemos eliminado de nuestro conjunto inicial de variables en un punto determinado del problema si nos proporcione ganancia útil para nuestra solución. Estos problemas ocurren ya que, aunque una variable individualmente o una variable en conjunto con otras no sea óptima, no significa que en combinación con otras no sea útil. Para solucionar esto están los métodos de SFFS y SFBS. El método SFFS arregla el error del SFS ya que, a medida que va añadiendo una característica a nuestro conjunto de variables solución, revisa si una de las variables de este conjunto puede omitirse para mejorar nuestro rendimiento. El método SFBS arregla el problema encontrado en el método SBS, ya que mira si una vez eliminada una variable, se puede añadir otra de las variables anteriormente borradas para mejorar nuestro conjunto de variables solución.

Los métodos de selección de filtrado seleccionan las variables de un conjunto de datos basándose en las características de dichas variables, por lo que se realiza el filtrado antes de realizar el aprendizaje. Las características de este tipo de método de selección es que son simples, económicas y ayudan a eliminar características rápidamente que no sean útiles para nuestro problema. Por esto, suelen ser utilizadas en los algoritmos de aprendizaje automático o en cualquier canal de selección de características. [10]



Fig. 10. Método de selección de características por filtrado

Dentro de los métodos de selección de filtrado existen dos tipos: los univariado y multivariado.

Los univariados tratan sólo una variable, es decir, sólo evalúan y clasifican una variable según unos criterios y lo realizan independientemente del conjunto de variables que tenemos. Tras esta evaluación individual nos quedaremos con la que más valor nos proporcione según el criterio establecido. Al tratar cada variable de manera individual sin combinación con las demás, puede darse el caso de que la variable elegida no sea útil para nuestro problema ya que no mira la interacción entre las diferentes variables.

Los métodos de filtro multivariante evalúan todo el conjunto de características por lo que el problema de las univariados estaría resuelto dado que evalúan las diferentes combinaciones de variables evitando así las variables redundantes y poco útiles.

Dentro de los métodos de selección de características por filtrado hay diferentes métodos para emplear a la hora de filtrar las diferentes variables que vamos a tratar. Los métodos son los siguientes:

* Métodos básicos de filtro: estos métodos básicos nos ayudaran a eliminar características constantes que no nos proporcionan información relevante en nuestro modelo dado que muestran valores únicos en todas las observaciones, eliminar características casi constantes y funciones duplicadas.
* Métodos de filtro de correlación: gracias a estos métodos podremos ver cómo de fuerte es la relación lineal entre dos variables. En el caso de que la variable que estamos estudiando presente una alta correlación lineal con la variable objetivo es un indicativo de que esta variable puede ser útil a la hora de obtener una solución, pero, si dos variables tienen una alta correlación lineal entre ellas y ninguna de ellas es la variable objetivo, puede ofrecernos que la información obtenida de estas sea algo redundante, algo que no es satisfactorio para nosotros. Por ello intentaremos buscar variables que cumplan el primer caso. Para medir la correlación de las variables existen varios métodos:

La correlación de Pearson mide la fuerza de relación lineal entre dos variables en valores que oscilan entre el 1 y -1. En el caso de que sea 1 el valor obtenido, significa que la correlación es positiva, es decir, que las dos variables aumentan equitativamente. Si el resultado es -1, significa correlación negativa que diría que una variable disminuye su valor cuando la otra aumenta su valor. Si el valor obtenido es 0, significa que no hay correlación lineal entre las variables de estudio. La fórmula que emplea para el cálculo de la correlación de Pearson es:

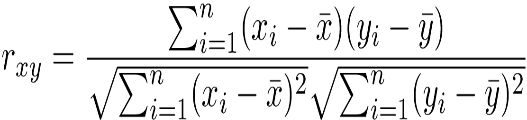


Fig. 11 Correlación de Pearson

El coeficiente de correlación de rango Spearman trata las variables que puedan tener una relación no lineal entre ellas. Para este caso, utilizaríamos esta prueba para medir el grado de asociación entre dos variables con relación creciente o decreciente. Mediríamos la fuerza entre las variables y oscilará entre los valores de 1 y -1, y, cuando se dan estos casos, es cuando la función con relación perfecta entre ellas. Esta sería la fórmula empleada:

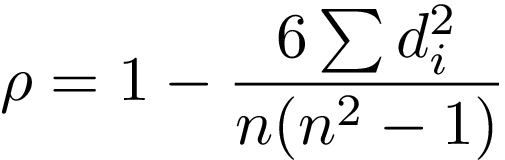


Fig. 12 Fórmula de Spearman

El coeficiente de correlación de rango de Kendall mide calcula la fuerza de asociación ordinal entre dos variables. Sus valores oscilan entre 1 y -1. En el caso de que sea 1 el valor obtenido significa que las variables tienen un rango similar y, en el caso de que sea -1, significa que tienen rango diferente entre ellas. La fórmula utilizada es:

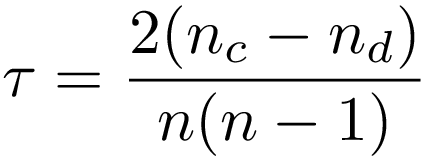


Fig. 13 Fórmula rango de Kendall

* Métodos de filtro estadístico y de clasificación: son procesos que utilizan pruebas estadísticas para evaluar las variables de manera individual para quedarse finalmente con la que mejor valor nos dé. Los criterios que siguen para esta clasificación pueden ser:

Información mutua: parecida a la correlación, pero de manera más general. Mide cuanto se puede saber de una variable según otra variable que estemos observando. Si las dos variables son independientes, su MI es 0. En el caso de que una variable sea determinista de otra, entonces MI es la entropía de la primera. La fórmula es la siguiente:

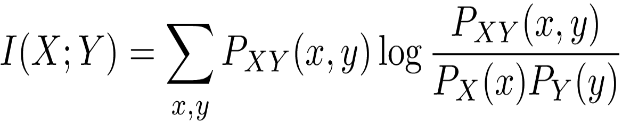


Fig. 14 Fórmula Información Mutua

Puntuación Chi-cuadrado: mira las relaciones entre dos variables categóricas. Por ello, presenta algunas limitaciones como que solo sirve para variables categóricas y objetivos binarios, entre otras.

Los métodos de selección de características integrados son parecidos a los métodos de envoltura sólo que los integrados realizan la selección de la mejor variable dentro del proceso de entrenamiento en vez de separados como vemos en los métodos de envoltura. Por esto, son más rápidos y precisos que los métodos de filtro y con mejor rendimiento que los métodos de envoltura. Los métodos integrados siguen un proceso para su realización: entrenan un modelo automático, obtienen las evaluaciones de los diferentes modelos y luego eliminan las variables que no nos son útiles para nuestro modelo. A continuación, se muestra de manera gráfica como funciona este tipo de método de selección de características: [11]

Imagen que contiene medidor, reloj

Descripción generada automáticamente

Fig. 15 Métodos de selección de características integrados

Para realizar estos métodos de selección de características integrados hay varios métodos que nos permiten hacerlo y son los siguientes: mediante regularización y mediante árboles.

* Regularización: limitan los parámetros de los modelos para conseguir evitar el sobreajuste, quitar el ruido… Hay tres tipos diferentes: regularización L1, regularización L2 y regularización L1/L2.

La regularización L1 pone ciertos parámetros a 0 lo que provoca que estas variables modificadas sean eliminadas dada su nula ganancia de información proporcionada. Esto será usado en la selección de características.

La regularización L2 es igual que L1 solo que los coeficientes no son 0, sino que son valores que se acercan a 0.

La regularización L1/L2: combinan las dos anteriores regularizaciones.

* Árboles: procedimiento que trabaja con árboles de decisión, que consiste en crear nodos a este árbol según una condición de una característica. Gracias a esto iremos dividiendo nuestro conjunto en dos conjuntos diferentes según la condición establecida, estando las variables similares en el mismo conjunto.

Una vez explicados los diferentes tipos de métodos de selección de características procederemos a compararlos entre ellos para ver las principales diferencias que poseen y nos aclarará cual debemos de usar en cada situación.

La diferencia entre los métodos de envoltura y los métodos de filtro son los siguientes: [12]

* Evalúan la importancia de unas características de manera diferente, mientras los métodos de envoltura calculan la importancia de un subconjunto de variables entrenando el modelo procedente de este subconjunto, los métodos de filtrado calculan el valor de una característica mediante la correlación lineal entre la característica tratada con la característica objetivo.
* Miden diferente la evaluación dado que los métodos por filtrado usan métodos estadísticos y los métodos de envoltura usan validación cruzada.
* Mientras los métodos de envoltura tienen un coste computacional elevado, los métodos de filtrado son muy rápidos.
* Los métodos de envoltura son más fiables a la hora de encontrar el mejor subconjunto de variables debido a su método de evaluación ya que los métodos de filtrado fallan a veces en este aspecto.

La diferencia entre los métodos de envoltura y los métodos integrados son:

* Mientras que los métodos de envolturas primero entrenan los modelos y luego los evalúa, los métodos integrados lo hacen a la vez que están entrenando el modelo.
* Debido a la diferencia anteriormente nombrada, los métodos integrados tienen un mejor rendimiento.

La diferencia entre los métodos de filtrado y los métodos integrados son:

* Los métodos integrados usan tanto los métodos de envoltura y de filtrado, por lo que esto hace que sea más eficiente que los métodos de filtrado.
* Los métodos integrados evalúan los modelos usando el mismo proceso que los métodos de envoltura, usa validación cruzada y los métodos de filtrado usan métodos estadísticos.

# Conclusiones

Este trabajo se desarrolla con la finalidad de obtener las mejores características o propiedades que sean capaces de clasificar con un mayor rendimiento nuestro conjunto de datos. Para ello, hemos desarrollado tanto un algoritmo de validación cruzada para evaluar el rendimiento basándonos en un número de experimentos y en un número de divisiones (folds:Cv), un algoritmo de Sequential Forward Selection (SFS) que lo emplearemos para introducirnos en el contexto del algoritmo objetivo del proyecto. Este algoritmo es una búsqueda secuencial que sólo añadía, en cada iteración, la mejor variable al conjunto de variables solución. Al no revisar si una variable añadida, que en su momento nos ofrecía una ganancia útil, en alguna iteración deja de ser necesaria en nuestro conjunto de variables solución. Para solucionar esto, hemos implementado el algoritmo objetivo del proyecto, el algoritmo Sequential Floating Forward Selection (SFFS), que partiendo de la idea de añadir variables como el anterior, también revisa si las variables dejan de ofrecernos una ganancia y empeora el rendimiento y, en este caso, la elimina, siendo así un algoritmo mucho más eficiente y práctico. Para la realización de estos algoritmos, hemos desarrollado varios métodos auxiliares para calcular la mejor variable, la peor variable del conjunto actual y ordenar los resultados obtenidos.

En conclusión, ha sido un proyecto interesante y ha sido todo un reto para nosotros, dado que es la primera vez que nos enfrentábamos a desarrollar un algoritmo de estas características y, aunque ha sido un proceso largo y lleno de incertidumbres, ha sido reconfortante ver que después de todo esto hemos conseguido desarrollar el algoritmo pedido. En cuanto a la técnica usada para conseguir el algoritmo nos ha hecho aplicar distintos conocimientos proporcionados por esta asignatura y hemos podido profundizar en ellos ya que lo hemos aplicado a un ejemplo real.

En cuanto a la mejora del trabajo, es difícil dar unas pautas de mejora dado que este proyecto ha sido desarrollado en unos momentos excepcionales y por tanto no hemos tenido prácticas ni clases. Nos habría gustado que, dado que nos hemos visto en una situación, se hubiese proporcionado más información respecto a Python dado que es un lenguaje que no hemos dado anteriormente y, aunque es parecido a Java, ha sido lioso en ciertos momentos. Quitando esto, estamos satisfechos con el proyecto.

##### Referencias