Selección de características en modelos predictivos

Jose Delgado Serrano
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
josdelser@alum.us.es

Antonio Fernández Manzano
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
antferman@alum.us.es

***Resumen*— A través de este trabajo conseguimos representar los resultados obtenidos mediante el proceso de selección de características en modelos predictivos, en concreto, con datos sobre el hundimiento del titanic y datos sobre diagnóstico del cáncer. Para ello hemos desarrollado tres algoritmos utilizando Python que son las herramientas indispensables para conseguir el resultado óptimo.**

Palabras Clave—Validación cruzada, evaluación robusta, SFS, SFFS, iteración.

I. Introducción

La selección de características o de variables es el proceso por el cual elegimos el subconjunto de atributos que mejor representa al conjunto completo de datos, con el fin de predecir el resultado, es decir, reducir la dimensión del problema mediante reconocimiento de patrones*[1]. Para obtener estos resultados, ha sido necesario implementar 3 algoritmos:

- Evaluación robusta: Consiste en la realización de varios experimentos utilizando la técnica de validación cruzada para promediar la tasa de aciertos de las variables analizadas.
- SFS (Sequential forward selection): Algoritmo que nos permite analizar un número de variables predictoras dadas mediante una búsqueda hacia delante. El objetivo de esta búsqueda es obtener el conjunto de variables óptimas que mejor evalúen la capacidad predictiva del conjunto de datos iniciales.
- SFFS (Sequential floating forward selection): Algoritmo que comienza con el conjunto completo de variables y consigue obtener el subconjunto óptimo mediante la eliminación de variables. Con esto conseguimos una gran precisión y una cohesión entre variables. Sin embargo, este algoritmo tiene una gran carga computacional.

Para saber cómo resolver este tipo de problemas, es necesario aprender primero el funcionamiento de la validación cruzada[1], el funcionamiento del algoritmo SFS[2], y finalmente cómo trabaja el algoritmo SFFS[3]. Tras analizar toda la información encontrada en Internet, analizamos el problema dado, y adaptamos la información a nuestras necesidades.

Comenzamos con el desarrollo de la evaluación robusta, en el que trabajamos con árboles de decisión como algoritmo de aprendizaje automático. Una vez terminado, utilizamos dicho método para evaluar las distintas posibles soluciones de los algoritmos SFS y SFFS. Al trabajar con validación cruzada, siempre obtendremos un pequeño margen de aleatoriedad, por lo que las soluciones pueden oscilar con un número bajo de experimentos.

El documento quedará dividido en secciones donde comenzaremos explicando en la sección II el comienzo y las bases del trabajo, continuando en la sección III con la metodología utilizada, y, finalmente, presentando los resultados obtenidos en la sección IV. Por último proporcionaremos las conclusiones obtenidas tras el desarrollo de este proyecto.

II. Preliminares

En esta sección se hace una breve introducción de las técnicas empleadas en el proyecto para llegar a la resolución del problema y como han sido diseñadas para que cumpla su función.

A.Métodos empleados

La técnica de búsqueda usada para resolver los problemas planteados ha sido la selección de características, donde a través del rendimiento íbamos sacando la mejor variable predictora hasta poder llegar al conjunto de variables seleccionadas con mayor rendimiento .

El primer paso de todo es conseguir leer los ficheros con las variables predictoras y sus valores, para esto hemos importado las librerías “pandas”

- **Pandas**[4] : Es una librería para el análisis de datos que cuenta con las estructuras de datos que necesitamos para limpiar los datos en bruto y que sean aptos para el análisis.

Antes de poder realizar los algoritmos SFS Y SFFS necesitábamos promediar esos rendimientos, para ello usamos la evaluación robusta, basada en la validación cruzada, la cual consiste en dos pasos. Primero sacamos el rendimiento de las variables y posteriormente promediamos ese rendimiento entre el número de variable usadas.

Para esto, hemos hecho uso de las librerías "sklearn.model_selection" para utilizar el método "cross_val_score" para la obtención del rendimiento para un conjunto de variables dado y también para la función "tree"

- **Sklearn(scikit-learn)**[5]: Es una librería que cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad. Uno de los algoritmos que hemos utilizado dentro de esta librería es el algoritmo de validación cruzada (cross_val_score)
- **Cross_val_score[**6]: Evalúa mediante la validación cruzada
- **Tree**[7]: Método de aprendizaje supervisado no paramétrico, su objetivo es crear un modelo que prediga el valor de una variable objetivo.

Para el diseño del algoritmo SFS , hemos ido jugando con la iteraciones para poder encontrar el conjunto de variable predictoras de mayor rendimiento. Este algoritmo compara el rendimiento de los diferentes conjuntos para guardar siempre el mayor.

En este algoritmo se ha precisado de las siguientes librerías:"pandas", "time","sklearn" y "copy".

- **Time**[8]: Obtenemos los tiempo de ejecución del algoritmo. Su utilización ayuda para mostrarnos la eficacia del algoritmo en cuanto a tiempo de ejecución,
- **Copy**[9]: Obtenemos una copia de un objeto que no se modifica al cambiar dicho objeto. Con esta librería evitamos los punteros a listas que aumentan la dificultad del código.

En cuanto al algoritmo SFFS, su diseño consta de dos listas para poder trabajar con el conjunto de datos, de esta manera podemos observar de manera más clara las variables eliminadas y las que ha ido añadiendo el algoritmo.

Para este diseño hemos trabajado con las librerías: "pandas", "time","sklearn" y "copy".

B. Trabajo Relacionado

Tras buscar información sobre trabajos y proyectos relacionados encontramos varias tesis de estudiantes sobre el tema como :

- Título: Selección de características. Su aplicación a clasificación de texturas. Autores: María Lucía Violini Director: J

III. METODOLOGÍA

En este punto vamos a describir las funciones implementadas y los pasos seguidos para llegar hasta el diseño óptimo de los algoritmos. Para entender mejor el funcionamiento de cada uno, iremos dividiendo cada uno de los 3 algoritmos en bucles

1) Primero pasaremos a explicar el funcionamiento del algoritmo de **evaluación robusta**:

- **1º While**: La función de este primer bucle es de realizar el número de experimentos que deseamos. Dentro del mismo vamos calculando diferentes valores del rendimiento por cada iteración.
  - La condición de parada es N_exp: número de experimentos a realizar.
- **2ºWhile**: Vamos promediando los resultados obtenidos en el bucle anterior hasta recorrer la lista con los valores del rendimiento
  - La condición de parada es Len(scores): longitud de la lista scores donde se almacena el conjunto de resultados de la evaluación robusta.

Después de tener una clara idea del funcionamiento de los bucles, analizamos el código completo. El algoritmo comienza entrando en el primer bucle donde vamos a ir calculando el rendimiento de las variables predictoras, la salida obtenida es "scores" la cual será la condición de parada del siguiente bucle. Una vez estemos dentro del segundo bucle, iremos promediando estos scores hasta hacerlo con todos.

Finalmente promediamos los resultados con el número de experimentos realizados.

Algoritmo evaluación robusta

Entrada:

- Conjunto de datos 'D'
- Variables a evaluar 'V'
- Número de experimentos 'n_exp'
- Número de folds 'cv'
- Árbol de decisión 'clf'
- Métrica de evaluación a usar 'scoring'

Salida:

- Tasa de aciertos balanceada promedio obtenida

Algoritmo:

1. Seleccionar la variable objetivo 'obj'
    a. i=0
2. Mientras i < n_exp
    a. Incrementar i en 1
    b. scores = validacionCruzada(clf,V,obj,cv,scoring)
    c. j=0
    d. Mientras j < tamaño(scores)
        i. resultado = suma de los valores de scores
        ii. Incrementar j en 1
    e. promedio = suma de los resultados promediados
    f. resultado = 0
3. Devolver el promedio con el número de experimentos PR

Pseudocódigo 1. Algoritmo de evaluación robusta

2) Ahora pasaremos a explicar el algoritmo de búsqueda **Sequential forward selection (SFS):**

- **1º IF:** Entramos en él en el caso de que no nos pasen la D como parámetro, por lo que la inicializamos a la cantidad de variables predictoras.

- **1ºWhile:** Itera un número de veces igual al número máximo de variables que le pasamos..
    - Condición de parada D. Número máximo de variables a evaluar por subconjunto.

- **2ºWhile:** Recorremos todas las variables predictoras una por una.
    - Condición de parada len (listaVariablesPredictorias). Lista con el conjunto de variables predictoras.

- **2ºIF:** Solo se ejecutará cuando la variable V no se haya tratado anteriormente. Posteriormente llamará a **Evaluación Robusta** , para sacar el rendimiento con V

- **3ºIF:** Comprueba que el rendimiento con V haya mejorado, si es así este valor pasa a ser el máximo y añadimos V a solución temporal,. en caso contrario sigue probando diferentes valores.

Primero comprobamos que le hayamos pasado el número de variables a probar (D), a continuación pasamos al bucle mayor que parará cuando hayamos recorrido D veces.

En el siguiente bucle vamos a ir iterando las variables predictoras, donde en caso de no haber sido evaluada calculamos su rendimiento, y posteriormente, lo vamos comparando con rendimientos anteriores para buscar el máximo. Si este rendimiento es mayor que el anterior pasaría a convertirse en la solución temporal hasta que haya recorrido todas las variables.

Una vez obtenido el máximo rendimiento se convertirá en la solución actual. En el caso de que no lo hubiese superado, se pasaría a la siguiente variable inmediatamente.

Algoritmo SFS

Entrada:

- Conjunto de datos 'D'
- Variables a evaluar 'd'
- Número de experimentos 'n_exp'
- Número de folds 'cv'
- Árbol de decisión 'clf'
- Métrica de evaluación a usar 'scoring'

Salida:

- Datagrama 'res' con los resultados ordenados por rendimiento

Algoritmo:

1. Inicialización de las variables a utilizar
    a. k = 0
    a. rendimientoTemporal = 0
    b. variables = variables predictoras
2. SI d == None
    a. Inicializarla al número de variables totales
3. Mientras k < d
    a. i = 0
    b. Mientras i < tamaño(variables)
        i. v = variables[i]
        ii. SI v no está en soluciónActual

1. solucionTemporal = solucionActual + v
2. rendimiento = evaluacionRobusta (solucionTemporal)
3. SI rendimiento > rendiemtoTemporal
    a. Guardo el mayor rendimiento
    b. Guardo la mejor variable

iii. Incremento i en 1

c. Actualizo soluciónActual con la nueva mejor variable
d. Añado al datagrama la solucionActual
e. Incremento k en 1

4. Devuelvo datagrama 'res' con las soluciones ordenadas por rendimiento

Pseudocódigo 2. Algoritmo SFS

3) Tras implementar el algoritmo anterior, pasamos al desarrollo del **Sequential floating forward selection (SFFS):**

- **1ºWhile:** La cantidad de veces que se va a ejecutar el algoritmo lo indicamos mediante el umbral.
    - Condición de parada Umbral.

- **1ºIF:** Comprobamos que todas las variables han sido añadidas a la lista anadidos, es decir, han sido evaluadas.

- **2ºWhile:** Recorremos todas las variables predictoras una por una.
    - Condición de parada len (listaVariablesPredictorias). Lista con el conjunto de variables predictoras.

- **2ºIF:** Solo se ejecutará cuando la variable V no se haya recorrido ya y no esté en anadidos. Posteriormente llamará a **Evaluación Robusta**, para sacar el rendimiento con V. En caso contrario aumentaría el valor de K (más adelante veremos cuál es su función)

- **3ºIF:** Comprueba que el rendimiento con V haya mejorado, si es así, este valor pasa a ser el máximo y la añado a solución temporal, en caso contrario sigue probando diferentes valores.

- **3ºWhile:** Vamos a iterar la solución actual para comprobar que no hay otro solución más óptima.
    - Condición de parada len(solucionActual). Tamaño de la solucionActual.

- **4º:IF:** Solo se ejecutará cuando la variable R no se haya evaluado ya y la lista de solucionActual contenga más de un elemento.. Posteriormente llamará a **Evaluación Robusta** , para sacar el rendimiento eliminando la variable R.

- **5ºIF:** Comprueba que el rendimiento sin R sea mayor que el rendimientoTemporal. Una vez hayamos evaluado todas las variables de la solucionActual, guardamos la variable con peor rendimiento.

- **6ºIF:** Comprobamos que aumenta el rendimiento tras eliminar la peor variable. Si es así, actualizamos el rendimiento y la soluciónActual.

  Teniendo en cuenta todos los bucles que forman el algoritmo pasamos al funcionamiento interno.

  Todo el método seguirá funcionando mientras no se supere el umbral marcado (K). A continuación comprobamos que no tengamos todas la variables añadidas a la lista anadidos. Si esto se cumple. recorremos todas las variables predictoras comprobando que cada variable no aparezca en la solucionActual ni en anadidos. En este caso vamos calculando el rendimiento (Evaluación Robusta) de cada una.

  Luego comparamos ese rendimiento con el mejor rendimientoTemporal buscando el mejor posible con cada variable V. Una vez hayamos recorrido todas las variables y sacado el máximo rendimiento , vamos iterando todos estos rendimientos.

  Aquí es donde entra la variable R, la cual corresponde a las variables que vamos borrando, para volver a comparar el rendimiento sin R, con el rendimiento con R con el objetivo de encontrar la peor variable.

  Posteriormente en caso de haber aumentado el rendimientoTemporal, este nuevo conjunto pasaría a ser nuestra solucionActual.

  Para finalizar, una vez tratadas todas las variables, volvemos a realizar todo el proceso de eliminación K veces para terminar de refinar el resultado. En caso de que encontrar un rendimiento mejor tras eliminar una variable, K volvería a valer 0 y empezará este proceso de nuevo.

Algoritmo SFFS

Entrada:

- Conjunto de datos ‘D
- Condición de parada (umbral)
- Número de experimentos ‘n_exp’
- Número de folds ‘cv’
- Árbol de decisión ‘clf’
- Métrica de evaluación a usar ‘scoring’

Salida:

- Datagrama ‘res’ con los resultados ordenados por rendimiento

Algoritmo:

1. Inicialización de las variables a utilizar
2. Mientras k < umbral
    a. k = 0
    b. rendimientoTemporal = 0
    c. rendimientoTemporal2 = 0
    d. variables = variables predictoras
    e. SI anadidos no contiene todas las variables
        i. i=0
        ii. rendimientoTemporal=0
        iii. Mientras i < tamaño(variables)
            1. v = variables[i]
            2. SI v no está en soluciónActual y no está en anadidos
                a. solucionTemporal = solucionActual + v
                b. rendimiento = evaluacionRobusta (solucionTemporal)
                c. SI rendimiento > rendiemtoTemporal
                    i. Guardo el mayor rendimiento
                    ii. Guardo la mejor variable
            3. Incremento i en 1
        iv. Actualizo soluciónActual con la nueva mejor variable
        v. Añado a anadidos la nueva variable
        vi. rendimientoTemporal2 = rendimientoTemporal
    f. SINO
        i. Incremento k en 1
    g. j = 0
    h. Mientras j < tamaño solucionActual
        i. R = solucionActual[j]
        ii. SI R no está en eliminados y hay más de 1 elemento en la soluciónActual
            1. solucionTemporal=solucionActual - R
            2. rendimiento2 = evaluacionRobusta(solucionTemporal)
            3. SI rendimiento2>rendimientoTemporal2
                a. Guardo en eliminadosTemporal la peor variable
                b. Guardo el mejor rendimiento
                c. k=0
            4. Incremento j en 1
    i. SI rendimientoTemporal2 > rendiemientoTemporal
        i. añado a eliminados eliminadosTemporal
        ii. elimino de solucionActual la variable eliminadosTemporal
        iii. rendimientoTemporal =rendimientoTemporal2
        iv. k=0
        v. añado el resultado con rendimientoTemporal2
    j. SINO
        i. añado el resultado con rendimientoTemporal
3. Devuelvo datagrama ‘res’ con las soluciones ordenadas por rendimiento

Pseudocódigo 3. Algoritmo SFFS

IV. RESULTADOS

Una vez terminada la implementación de todo el código requerido, nos disponemos a exponer los resultados obtenidos. Para ello hemos realizado experimentos independientes para cada uno de los conjunto de datos de prueba para obtener la combinación perfecta que nos devuelva el resultado óptimo.

- En primer lugar comenzamos con los datos del titanic. Evaluamos en primer lugar el algoritmo SFS. Para el punto de partida hemos utilizado 10 experimentos, cv de valor 10 y todas las variables predictoras.

Figura 1.

Tiempo de ejecución total = 87.43955898284912 segundos

	solution	score	size
4	[Initial, SibSp, Deck, Fare_cat, Title]	0.818232	5
5	[Initial, SibSp, Deck, Fare_cat, Title, Sex]	0.818232	6
6	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.818232	7
7	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.813820	8
3	[Initial, SibSp, Deck, Fare_cat]	0.811655	4
2	[Initial, SibSp, Deck]	0.809191	3
8	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.807603	9
1	[Initial, SibSp]	0.805218	2
11	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.804462	12
9	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.804362	10
10	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.801810	11
12	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.796186	13
0	[Initial]	0.783354	1
13	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.773089	14
14	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.757589	15

Resultados titanic SFS

Vemos en un primer resultado, que el tiempo de ejecución ha sido de 87 segundos, un tiempo un poco elevado pero aceptable, y el conjunto de variables predictoras óptimas han sido: [Initial, Sibsp, Deck, Fare_car, Title].

Partiendo de esta base, hemos ido modificando valores hasta obtener el resultado óptimo. Sabemos con esos valores, que el umbral mínimo que podemos poner es de 6, ya que el número mínimo de variables predictoras que nos da el mejor resultado es de 5, igualado con 6.

Por lo tanto nuestro siguiente experimento fue con los valores: 7 variables a evaluar, 10 experimentos y cv de 6. Dejamos el mismo número de experimentos para partir de la misma base anterior.

Tiempo de ejecución total = 39.93417716026306 segundos

	solution	score	size
4	[Initial, SibSp, Deck, Fare_cat, Title]	0.820749	5
5	[Initial, SibSp, Deck, Fare_cat, Title, Sex]	0.820749	6
6	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.820749	7
3	[Initial, SibSp, Deck, Fare_cat]	0.818371	4
2	[Initial, SibSp, Deck]	0.815107	3
1	[Initial, SibSp]	0.805416	2
0	[Initial]	0.783611	1

Figura 2. Mejores resultados titanic SFS

Observamos que el rendimiento del conjunto óptimo ha mejorado, por lo que deducimos tras más pruebas, que el cv óptimo para este conjunto de datos debía ser 6. Además, con la reducción del umbral a 7, es decir, evaluar como máximo conjuntos de 7 variables predictoras, nos ofrece una mejora bastante importante del tiempo de ejecución.

Finalmente realizamos una prueba con 1 solo experimento en la que obtuvimos el mismo resultado anterior pero con un tiempo de ejecución de 4 segundos (véase la imagen inferior). Obviamente esto en la práctica es algo impensable, ya que 1 solo experimento no nos ofrece una visión clara del problema.

Tiempo de ejecución total = 4.163998365402222 segundos

	solution	score	size
4	[Initial, SibSp, Deck, Fare_cat, Title]	0.820749	5
5	[Initial, SibSp, Deck, Fare_cat, Title, Sex]	0.820749	6
6	[Initial, SibSp, Deck, Fare_cat, Title, Sex, I...	0.820749	7
3	[Initial, SibSp, Deck, Fare_cat]	0.818371	4
2	[Initial, SibSp, Deck]	0.815107	3
1	[Initial, SibSp]	0.805416	2
0	[Initial]	0.783611	1

Figura 3. Resultados titanic SFS con 1 experimento

Por lo tanto concluimos que para el algoritmo Sequential forward selection, el conjunto de variables óptimas han sido: Initial, Sibsp, Deck, Fare_Cat, Title. El rendimiento ofrecido es de 0,820749, con los parámetros:

- Variables a evaluar=7
- Número de experimentos =10
- Cv=6

Vemos aquí el resultado de la gráfica para esta solución.

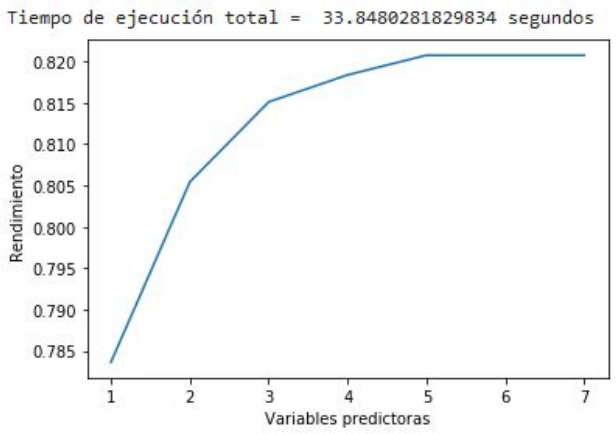

Figura 4. Gráfica titanic del mejor resultado

Seguimos a continuación con el algoritmo SFFS. Comenzamos con unos valores iniciales de:

- Condición de parada (umbral) = 10
- Número de experimentos = 10
- Cv = 6

Tiempo de ejecución total = 118.84426164627075 segundos

	solution	score	size
6	[SibSp, Deck, Fare_cat, Title, Sex]	0.821481	5
12	[SibSp, Deck, Fare_cat, Title, Sex]	0.821390	5
5	[SibSp, Deck, Fare_cat, Title, Sex]	0.821298	5
8	[SibSp, Deck, Fare_cat, Title, Sex]	0.821298	5
14	[SibSp, Deck, Fare_cat, Title, Sex]	0.821207	5
23	[SibSp, Deck, Fare_cat, Title, Sex]	0.821207	5
22	[SibSp, Deck, Fare_cat, Title, Sex]	0.821207	5
21	[SibSp, Deck, Fare_cat, Title, Sex]	0.821207	5
20	[SibSp, Deck, Fare_cat, Title, Sex]	0.821207	5
19	[SibSp, Deck, Fare_cat, Title, Sex]	0.821207	5
18	[SibSp, Deck, Fare_cat, Title, Sex]	0.821207	5
17	[SibSp, Deck, Fare_cat, Title, Sex]	0.821207	5
16	[SibSp, Deck, Fare_cat, Title, Sex]	0.821207	5
15	[SibSp, Deck, Fare_cat, Title, Sex]	0.821207	5

Figura 5. Resultados SFFS titanic iniciales

Observamos un tiempo de ejecución cercano a los 2 minutos, y un resultado parecido al algoritmo SFS. Partiendo de esta base, seguimos modificando parámetros.

Para esta segunda prueba modificamos el número de experimentos a 15 para comprobar que el resultado obtenido no es tan aleatorio.

Tiempo de ejecución total = 215.68495893478394 segundos

	solution	score	size
33	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
28	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
23	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
24	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
25	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
27	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
26	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
29	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
30	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
31	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
32	[SibSp, Deck, Fare_cat, Title, Sex]	0.821542	5
10	[SibSp, Deck, Fare_cat, Title, Sex, Is_Married]	0.821420	6

Figura 6. Resultados SFFS titanic aumentando experimentos

Observamos que el tiempo de ejecución ha aumentado hasta el doble, con una mejora del rendimiento de 0,0001 unidades, pero manteniéndose el conjunto de tamaño 5 como resultado óptimo.

Esta mejora del rendimiento nos llevó a pensar en que un aumento del número de experimentos podría llevarnos a un mejor resultado. Quedaba aún una incógnita que resolver, la condición de parada. Decidimos aumentarla hasta un valor de 30, lo que devolvió el siguiente resultado:

6]: Tiempo de ejecución total = 328.30650758743286 segundos

	solution	score	size
7	[SibSp, Deck, Fare_cat, Title, Sex, Is_Married]	0.821344	6
12	[SibSp, Deck, Fare_cat, Title, Sex, Is_Married]	0.821344	6
13	[SibSp, Deck, Fare_cat, Title, Sex, Is_Married]	0.821298	6
6	[SibSp, Deck, Fare_cat, Title, Sex, Is_Married]	0.821252	6
9	[SibSp, Deck, Fare_cat, Title, Sex, Is_Married]	0.821252	6
37	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
28	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
29	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
30	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
31	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
32	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
33	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
34	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
35	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
36	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
23	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
38	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5
26	[SibSp, Deck, Fare_cat, Title, Sex]	0.821252	5

Figura 7. Resultados SFFS titanic aumentando el umbral

Obtuvimos un aumento considerable del tiempo de ejecución, con una disminución del rendimiento.

Con todos estos experimentos, deducimos que los valores óptimos para el algoritmo SFFS con el conjunto de datos del titanic, son los siguientes:

- Condición de parada (umbral) = 10
- Número de experimentos = 10-15. A mayor número de experimentos mejor resultado pero peor tiempo de ejecución.
- Cv = 6

En cuanto al rendimiento óptimo tendríamos un rango entre 0,821481 y 0,821542.

La gráfica obtenida es la siguiente:

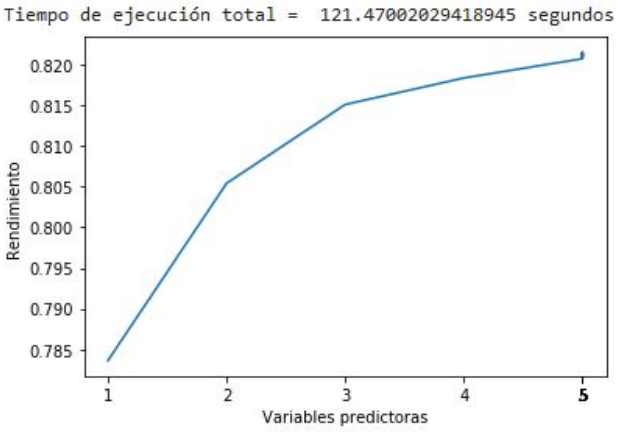

Figura 8. Gráfica SFFS titanic mejor resultado

- A continuación tenemos el conjunto de prueba de datos del cáncer. Este conjunto de datos es mucho más extenso que el anterior, por lo que los resultados no son tan precisos si queremos que sea eficiente a la vez.

Comenzamos probando con los valores estándar del algoritmo SFS propuestos por los desarrolladores: 10 experimentos, cv de valor 10 y todas las variables predictoras.

Tiempo de ejecución total = 375.68080139160156 segundos

	solution	score	size
12	[worst concave points, worst area, mean textur...	0.961623	13
10	[worst concave points, worst area, mean textur...	0.961282	11
11	[worst concave points, worst area, mean textur...	0.961099	12
9	[worst concave points, worst area, mean textur...	0.960200	10
8	[worst concave points, worst area, mean textur...	0.959971	9
13	[worst concave points, worst area, mean textur...	0.959911	14
14	[worst concave points, worst area, mean textur...	0.958958	15
7	[worst concave points, worst area, mean textur...	0.958025	8
17	[worst concave points, worst area, mean textur...	0.957705	18
6	[worst concave points, worst area, mean textur...	0.957376	7
15	[worst concave points, worst area, mean textur...	0.956617	16
5	[worst concave points, worst area, mean textur...	0.955987	6
16	[worst concave points, worst area, mean textur...	0.955732	17
4	[worst concave points, worst area, mean textur...	0.955626	5
18	[worst concave points, worst area, mean textur...	0.955449	19
19	[worst concave points, worst area, mean textur...	0.954140	20
20	[worst concave points, worst area, mean textur...	0.953391	21
21	[worst concave points, worst area, mean textur...	0.951387	22
22	[worst concave points, worst area, mean textur...	0.950759	23
3	[worst concave points, worst area, mean textur...	0.949773	4

Figura 9. Resultados SFS cáncer iniciales

Observamos un tiempo de ejecución de 375 segundos, cerca de 6 minutos y medio. Obtenemos un rendimiento muy alto de 0.961623. A partir de este conjunto comenzamos a modificar los parámetros para obtener alguna mejora.

En primer lugar aumentamos el valor de cv a 20.

Tiempo de ejecución total = 825.2820827960968 segundos

	solution	score	size
9	[worst concave points, worst area, mean textur...	0.964798	10
10	[worst concave points, worst area, mean textur...	0.963035	11
8	[worst concave points, worst area, mean textur...	0.962392	9
11	[worst concave points, worst area, mean textur...	0.962275	12
12	[worst concave points, worst area, mean textur...	0.961097	13
7	[worst concave points, worst area, mean textur...	0.960750	8
6	[worst concave points, worst area, mean textur...	0.959593	7
15	[worst concave points, worst area, mean textur...	0.959294	16
5	[worst concave points, worst area, mean textur...	0.959144	6
13	[worst concave points, worst area, mean textur...	0.958647	14
14	[worst concave points, worst area, mean textur...	0.958330	15
17	[worst concave points, worst area, mean textur...	0.958032	18
4	[worst concave points, worst area, mean textur...	0.957899	5
16	[worst concave points, worst area, mean textur...	0.957736	17
18	[worst concave points, worst area, mean textur...	0.956520	19
19	[worst concave points, worst area, mean textur...	0.955640	20
21	[worst concave points, worst area, mean textur...	0.954318	22
20	[worst concave points, worst area, mean textur...	0.953656	21
22	[worst concave points, worst area, mean textur...	0.952120	23
3	[worst concave points, worst area, mean textur...	0.951541	4
23	[worst concave points, worst area, mean textur...	0.950843	24
24	[worst concave points, worst area, mean textur...	0.949723	25

Figura 10. Resultados SFFS cáncer tras subir el cv

El tiempo de ejecución se ha duplicado y el rendimiento ha mejorado en 0,003 unidades. Pensamos que este aumento del tiempo de ejecución que trae consigo un aumento del rendimiento, no es eficiente puesto que la mejora no es demasiado abultada.

Puesto que aumentando cv ha mejorado, probamos un valor de 15

Tiempo de ejecución total = 569.9223654270172 segundos

	solution	score	size
14	[worst area, worst smoothness, mean texture, m...	0.965827	15
11	[worst area, worst smoothness, mean texture, m...	0.964495	12
8	[worst area, worst smoothness, mean texture, m...	0.964356	9
15	[worst area, worst smoothness, mean texture, m...	0.964350	16
10	[worst area, worst smoothness, mean texture, m...	0.963846	11
9	[worst area, worst smoothness, mean texture, m...	0.963846	10
7	[worst area, worst smoothness, mean texture, m...	0.963337	8
13	[worst area, worst smoothness, mean texture, m...	0.963088	14
6	[worst area, worst smoothness, mean texture, m...	0.962967	7
12	[worst area, worst smoothness, mean texture, m...	0.962782	13
5	[worst area, worst smoothness, mean texture, m...	0.961948	6
16	[worst area, worst smoothness, mean texture, m...	0.961790	17
17	[worst area, worst smoothness, mean texture, m...	0.960044	18
4	[worst area, worst smoothness, mean texture, m...	0.959825	5
18	[worst area, worst smoothness, mean texture, m...	0.957179	19
19	[worst area, worst smoothness, mean texture, m...	0.955006	20
3	[worst area, worst smoothness, mean texture, m...	0.954493	4
20	[worst area, worst smoothness, mean texture, m...	0.953373	21
24	[worst area, worst smoothness, mean texture, m...	0.952724	25
22	[worst area, worst smoothness, mean texture, m...	0.952605	23
23	[worst area, worst smoothness, mean texture, m...	0.952219	24

Figura 11. Resultados SFS cáncer tras disminuir cv a 15

El tiempo de ejecución ahora es de aproximadamente 10 minutos y el rendimiento obtenido es hasta ahora el máximo. Por consiguiente decidimos fijar cv a un valor de 12. Ahora solo nos faltaría el número de experimentos y a partir de ahí fijar un valor para el número de variables a evaluar.

Sin embargo, aumentando el número de experimentos hace que vuelva a aumentar de forma considerable el tiempo de ejecución tal y como vemos en la siguiente imagen:

Tiempo de ejecución total = 843.3797755241394 segundos

	solution	score	size
9	[worst area, worst smoothness, mean texture, m...	0.965903	10
15	[worst area, worst smoothness, mean texture, m...	0.965812	16
13	[worst area, worst smoothness, mean texture, m...	0.965010	14
10	[worst area, worst smoothness, mean texture, m...	0.964603	11
14	[worst area, worst smoothness, mean texture, m...	0.964565	15
11	[worst area, worst smoothness, mean texture, m...	0.964374	12
8	[worst area, worst smoothness, mean texture, m...	0.963946	9
12	[worst area, worst smoothness, mean texture, m...	0.963837	13
6	[worst area, worst smoothness, mean texture, m...	0.963723	7
16	[worst area, worst smoothness, mean texture, m...	0.963293	17
7	[worst area, worst smoothness, mean texture, m...	0.963172	8
5	[worst area, worst smoothness, mean texture, m...	0.962242	6
17	[worst area, worst smoothness, mean texture, m...	0.961093	18
4	[worst area, worst smoothness, mean texture, m...	0.960057	5
18	[worst area, worst smoothness, mean texture, m...	0.958373	19
20	[worst area, worst smoothness, mean texture, m...	0.955976	21
19	[worst area, worst smoothness, mean texture, m...	0.954330	20
3	[worst area, worst smoothness, mean texture, m...	0.953058	4
23	[worst area, worst smoothness, mean texture, m...	0.951472	24
24	[worst area, worst smoothness, mean texture, m...	0.951101	25
21	[worst area, worst smoothness, mean texture, m...	0.950299	22
26	[worst area, worst smoothness, mean texture, m...	0.950056	27
22	[worst area, worst smoothness, mean texture, m...	0.949758	23
25	[worst area, worst smoothness, mean texture, m...	0.948989	26
27	[worst area, worst smoothness, mean texture, m...	0.947005	28
2	[worst area, worst smoothness, mean texture]	0.941280	3

Figura 12. Resultados SFS cáncer tras aumentar el número de experimentos

El rendimiento apenas mejora, por lo que creemos más eficiente fijar el número de experimentos a 10.

Por último al observar todos los resultados obtenidos, decidimos fijar el número de variables a evaluar a 17, obteniendo este resultado final:

Tiempo de ejecución total = 398.26311683654785 segundos

	solution	score	size
12	[worst area, worst smoothness, mean texture, m...	0.967263	13
11	[worst area, worst smoothness, mean texture, m...	0.966211	12
13	[worst area, worst smoothness, mean texture, m...	0.965397	14
10	[worst area, worst smoothness, mean texture, m...	0.964629	11
9	[worst area, worst smoothness, mean texture, m...	0.964172	10
14	[worst area, worst smoothness, mean texture, m...	0.964134	15
8	[worst area, worst smoothness, mean texture, m...	0.963728	9
7	[worst area, worst smoothness, mean texture, m...	0.963469	8
6	[worst area, worst smoothness, mean texture, m...	0.962862	7
15	[worst area, worst smoothness, mean texture, m...	0.962388	16
5	[worst area, worst smoothness, mean texture, m...	0.961717	6
16	[worst area, worst smoothness, mean texture, m...	0.961277	17
4	[worst area, worst smoothness, mean texture, m...	0.959737	5
3	[worst area, worst smoothness, mean texture, m...	0.952456	4
2	[worst area, worst smoothness, mean texture]	0.941898	3
1	[worst area, worst smoothness]	0.928942	2
0	[worst area]	0.873896	1

Figura 13. Mejores resultados SFS cáncer

Obtenemos un tiempo de ejecución bastante correcto en relación a este problema y observamos el mejor rendimiento posible hasta ahora. Ese rendimiento es ofrecido por las variables:

- worst area,
- worst smoothness
- mean texture
- mean concave points
- mean symmetry
- area error
- worst radius
- mean perimeter
- fractal dimension error
- concave points error
- concavity error
- mean smoothness
- smoothness error

Observamos en la siguiente gráfica la relación entre el rendimiento y las variables predictoras en el algoritmo.

Tiempo de ejecución total = 390.78312277793884 segundos

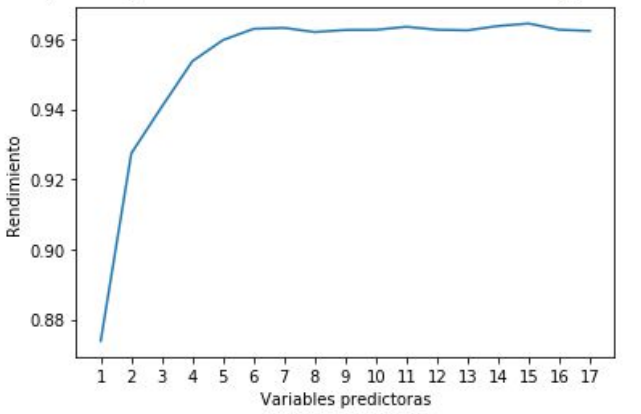

Figura 14. Gráfica mejores resultados cáncer

En definitiva, para el algoritmo SFS y el conjunto de datos de prueba del cáncer, los parámetros que mejores resultados han ofrecido son:

- Número de experimentos = 10
- Variables a evaluar = 17
- cv = 12

Finalmente pasamos a probar el algoritmo SFFS con los datos de prueba del cáncer. Comenzamos con un umbral de 10, un número de experimentos igual a 10 y cv de 15

Tiempo de ejecución total = 1561.3180284500122 segundos

	solution	score	size
13	[worst concave points, worst area, mean textur...	0.961760	12
22	[worst concave points, worst area, mean textur...	0.961058	14
17	[worst concave points, worst area, mean textur...	0.960947	14
19	[worst concave points, worst area, mean textur...	0.960915	14
20	[worst concave points, worst area, mean textur...	0.960915	14
39	[worst concave points, worst area, mean textur...	0.960834	11
40	[worst concave points, worst area, mean textur...	0.960834	11
41	[worst concave points, worst area, mean textur...	0.960834	11
38	[worst concave points, worst area, mean textur...	0.960834	11
42	[worst concave points, worst area, mean textur...	0.960834	11
47	[worst concave points, worst area, mean textur...	0.960834	11
44	[worst concave points, worst area, mean textur...	0.960834	11

Figura 15. Resultados SFFS cáncer iniciales

Vemos un buen rendimiento pero a costa de un tiempo de ejecución demasiado elevado. Por eso disminuimos el número de cv a 6 para la siguiente prueba.

Tiempo de ejecución total = 449.0450322628021 segundos

	solution	score	size
22	[worst radius, mean texture, mean concave poin...	0.964949	12
25	[worst radius, mean texture, mean concave poin...	0.963831	12
43	[worst radius, mean texture, mean concave poin...	0.963747	8
42	[worst radius, mean texture, mean concave poin...	0.963747	8
41	[worst radius, mean texture, mean concave poin...	0.963747	8
40	[worst radius, mean texture, mean concave poin...	0.963747	8
39	[worst radius, mean texture, mean concave poin...	0.963747	8
38	[worst radius, mean texture, mean concave poin...	0.963747	8
37	[worst radius, mean texture, mean concave poin...	0.963747	8
36	[worst radius, mean texture, mean concave poin...	0.963747	8
35	[worst radius, mean texture, mean concave poin...	0.963747	8
34	[worst radius, mean texture, mean concave poin...	0.963747	8
44	[worst radius, mean texture, mean concave poin...	0.963747	8
33	[worst radius, mean texture, mean concave poin...	0.963550	9
32	[worst radius, mean texture, mean concave poin...	0.963550	9
31	[worst radius, mean texture, mean concave poin...	0.963425	10
28	[worst radius, mean texture, mean concave poin...	0.962733	12
30	[worst radius, mean texture, mean concave poin...	0.962669	11

Figura 16. Mejores resultados SFFS cáncer disminuyendo cv

En comparación con la anterior, el tiempo de ejecución disminuyó de los 26 minutos a 7 minutos y medio, además el rendimiento obtenido es mucho mayor. Por lo tanto fijamos el número de cv a 6. A continuación pasamos a modificar el número de experimentos, el cual vamos a aumentar a 15:

Tiempo de ejecución total = 680.0246567726135 segundo

	solution	score	size
14	[worst concave points, worst radius, mean text...	0.961523	10
27	[worst concave points, worst radius, mean text...	0.960485	11
40	[worst concave points, worst radius, mean text...	0.960316	9
35	[worst concave points, worst radius, mean text...	0.960316	9
31	[worst concave points, worst radius, mean text...	0.960316	9
39	[worst concave points, worst radius, mean text...	0.960316	9
32	[worst concave points, worst radius, mean text...	0.960316	9
33	[worst concave points, worst radius, mean text...	0.960316	9
34	[worst concave points, worst radius, mean text...	0.960316	9
41	[worst concave points, worst radius, mean text...	0.960316	9
36	[worst concave points, worst radius, mean text...	0.960316	9
37	[worst concave points, worst radius, mean text...	0.960316	9
38	[worst concave points, worst radius, mean text...	0.960316	9
18	[worst concave points, worst radius, mean text...	0.960214	11

Figura 17. Resultados SFFS cáncer aumentando experimentos

El tiempo de ejecución como era de esperar ha aumentado hasta los 11 minutos, y el rendimiento ha disminuido, por lo tanto, como queremos obtener un resultado preciso pero sin demasiado tiempo de espera, decidimos que el número de experimentos óptimos será entre 10 y 15.

Por último modificamos la condición de parada, manteniendo el número de experimentos a 10 y el cv a 6. El resultado obtenido fue el siguiente:

Tiempo de ejecución total = 554.3464980125427 segundos

	solution	score	size
13	[worst concave points, worst radius, mean text...	0.961563	10
55	[worst concave points, worst radius, mean text...	0.961059	8
50	[worst concave points, worst radius, mean text...	0.961059	8
43	[worst concave points, worst radius, mean text...	0.961059	8
42	[worst concave points, worst radius, mean text...	0.961059	8
41	[worst concave points, worst radius, mean text...	0.961059	8
40	[worst concave points, worst radius, mean text...	0.961059	8
39	[worst concave points, worst radius, mean text...	0.961059	8
38	[worst concave points, worst radius, mean text...	0.961059	8
37	[worst concave points, worst radius, mean text...	0.961059	8
36	[worst concave points, worst radius, mean text...	0.961059	8
35	[worst concave points, worst radius, mean text...	0.961059	8
46	[worst concave points, worst radius, mean text...	0.961059	8
47	[worst concave points, worst radius, mean text...	0.961059	8

Figura 18. Resultados SFFS cáncer modificando condición de parada

Vemos un aumento en el tiempo de ejecución, y ningún cambio en el rendimiento, por lo que fijamos la condición de parada al valor anterior, 10.

Observamos que al tener tantos datos en el conjunto de entrada, se hace muy complicado obtener un resultado que no oscile demasiado con un algoritmo eficiente en el tiempo. Es por eso que el conjunto de variables óptimas, oscilan entre un tamaño de 10 y un tamaño de 12. Tomando como el mejor rendimiento, el obtenido en la figura 16, estas serían:

- worst area,
- worst smoothness
- mean texture
- mean concave points
- mean symmetry
- area error
- worst radius
- mean perimeter
- fractal dimension error
- concave points error
- concavity error
- mean smoothness

Vemos en la gráfica de la mejor solución cómo ha ido cambiando el rendimiento respecto al número de variables obtenidas.

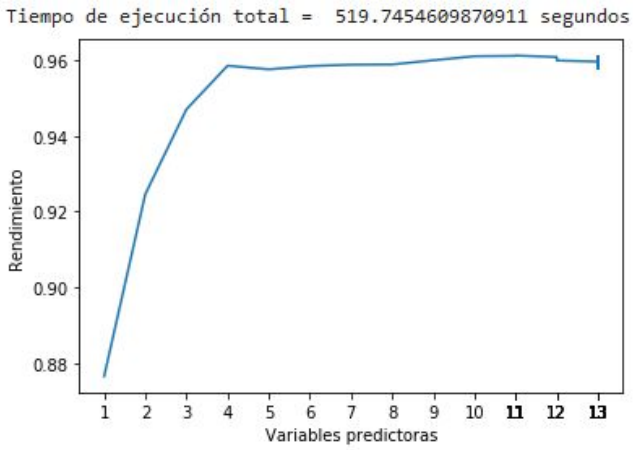

Figura 19. Gráfica mejores resultados SFFS cáncer

Para este algoritmo con estos datos, los parámetros óptimos son:

- Condición de parada (umbral) = 10
- Número de experimentos = 10-15
- cv = 6

- Análisis de los resultados, haciendo comparativas y obteniendo conclusiones.

Tras testear ambos algoritmos con ambos conjuntos de datos, sacamos las siguientes conclusiones:

1. El algoritmo SFS es mucho más eficiente en el tiempo pero nos devuelve unos resultados menos precisos, por lo que podemos estar obviando alguna variable predictora que nos mejore el resultado. Sin embargo, su rápida ejecución nos sirve para obtener una vista previa del problema a evaluar.

   En cuanto a los parámetros óptimos serían los siguientes:

   - Número de experimentos = 10
   - Cv= Rango entre 6 y 12 dependiendo del tamaño de los datos
   - Número de variables a evaluar = Dependiendo de resultados previos

2. El algoritmo SFFS nos ofrece resultados más exactos pero menos eficiente en cuanto a tiempo. Mediante este algoritmo, podremos obtener la mejor combinación de variables gracias a que realiza todas las pruebas posibles con cada variable. Esto hace que para obtener unos resultados estables, necesitemos muchos experimentos, lo que eleva demasiado el tiempo de ejecución.

   Por tanto, los valores recomendados para este algoritmo son los siguientes:

   - Número de experimentos = 10
   - Cv= 6
   - Condición de parada (umbral) = 10

V. CONCLUSIONES

Como hemos ido documentando, para llegar al diseño final del algoritmo hemos tenido que comprender diferentes métodos iterativos y tener en cuenta las diferentes posibilidades hasta poder llegar a implementar de manera correcta el algoritmo.

Nos ha resultado de gran ayuda importar algunas librerías para poder hacer operaciones rápidamente y de esta manera ahorrarnos métodos demasiado complejos, como por ejemplo, la librería copy, que hemos utilizado para la copia de listas, para evitar el siempre complicado uso de punteros. También las librerías para dibujar las gráficas y las tablas con los resultados, nos acortaron bastante trabajo.

Conforme a los experimentos y conclusiones obtenidas, nos dimos cuenta que al aumentar el número de experimentos obtuvimos resultados más precisos, pero el tiempo de ejecución aumentaba drásticamente, por lo que tuvimos que buscar el equilibrio para conseguir un algoritmo completo.

Una vez comprendido el algoritmo, observamos que una de las maneras de mejorar los tiempos es paralelizar la ejecución del rendimiento con el guardado de datos en variables, también paralelizar el entrenamiento de los árboles de decisión mejorará comprensiblemente el tiempo de ejecución del algoritmo. Esta parte quedaría para una posible mejora futura con vistas a un algoritmo más eficiente.

Finalmente, las gráficas ayudan a visualizar mejor el rendimiento máximo y el punto óptimo del algoritmo, un hecho que mejora la comprensión de la selección de características en modelos predictivos.

Vista la importancia que la selección de características tiene en el mundo de machine learning vamos a realizar un pequeño estudio sobre algunos tipos.

La selección de características es un punto fundamental, ya que nos permite poder eliminar características que entorpecen el rendimiento y nos producen sobreajustes. Procederemos a explicar y comparar 3 tipos de métodos que son muy interesantes a la hora de realizar un proyecto de mayor potencial.

El primer método que vamos a considerar es el **Filtrado** [11]

Este tipo de método de preprocesamiento de datos que selecciona las características es independiente para cualquier tipo de algoritmo de machine learning . Esto es debido a las metodologías seguidas para filtrar las características. Existen dos tipos:

- **Univariante**: Evalúan cada característica de forma individual, según un criterio dado, sin tener en cuenta la relación que existen entre ellas. Esto produce un problema ya que puede seleccionar una característica redundante.[10]

- **Multivariante**: Tienen en cuenta la relación entre las características por lo cual no habría problema de redundancia.[10]

A partir de estas metodologías se han ido creando diferentes métodos basados en univariante y multivariante:

- **Filtros Básicos:** Métodos sencillos que ayudan bastante a filtrar características que siempre tienen el mismo valor o las que están duplicadas, ya que no aportan información útil al modelo de machine learning.

- **Filtros de correlación :** Este filtro trata de eliminar características que están altamente correlacionadas. Ahora bien, al existir esta unión podemos prescindir de la que sea menos influyente, haciendo de esta manera una aproximación de las dos en una sola característica.

  Para obtener esta correlación y elegir qué elemento filtrar tenemos varias formas:

  - Coeficiente de correlación de rango de Kendall,
  - Coeficiente de correlación de rango de Spearman
  - Coeficiente de correlación de Pearson.

- **Filtros estadístico y de clasificación:** Esto filtros básicamente evalúan las características de forma individual según un criterio concreto para posteriormente elegir las que mejor resultado devuelven. Para ello tenemos diferentes posibilidades: Información Mutua, Puntuación Chi-cuadrado, Prueba univariante ANOVA y ROC-AUC / RMSE univariante.

Como conclusión obtenemos que este método funciona realmente bien para filtrar características duplicadas, que no aportan valor alguno o que están tan fuertemente ligadas que se puede prescindir de algunas.

Para entender esto mejor vamos a proceder a enumerar sus ventajas y desventajas:

- **Ventajas:** Su alta sencillez por lo que tiene bajo coste computacional y no requieren del uso de un algoritmo para calcular la capacidad de las características.

- **Desventajas:** Poseen menor rendimiento debido a que no contemplan las dependencias entre las diferentes características.

El segundo método que vamos a tratar es el método de **Envoltura.**

Para el método de envoltura, se necesita un algoritmo de Machine Learning y utiliza su rendimiento como criterio de evaluación. En la imagen inferior podemos ver una vista previa de cómo trabaja este tipo de método.

Figura 20. Método de envoltura

Este método busca el mejor rendimiento mediante la adición de características. Algunos ejemplos de métodos de envoltura son:

- **Selección hacia adelante:** Este método es el algoritmo SFS implementado en este trabajo.
- **Eliminación hacia atrás:** Este método es el antagónico del anterior, es decir, parte de un conjunto con todas las variables y las elimina según el rendimiento óptimo.

- **Eliminación de características recursivas:** Crea repetidamente modelos dejando de lado las características que necesite para, posteriormente, clasificarlas según su orden de eliminación.
- **Selección de características exhaustiva:** Prueba todas las combinaciones de características posibles.
- **Selección bidireccional:** Realiza tanto la selección hacia adelante como la eliminación hacia atrás de formas simultánea. Este método es el equivalente al algoritmo SFFS implementado en este trabajo.

Generalmente hablando, este tipo de métodos siguen un proceso de trabajo similar, que consiste en el siguiente proceso:

1. Seleccionar un conjunto de características
2. Utilizar un algoritmo de Machine Learning para entrenarlo usando el conjunto seleccionado
3. Obtener el rendimiento del conjunto tras evaluarlo con el algoritmo
4. Repetir hasta la condición de parada

Los métodos de envoltura tienen la utilidad de poder evaluar características en conjunto, es decir, puede haber una variable con un rendimiento óptimo de forma individual pero puede ser inútil si se combina con otras características. Es por esto que este tipo de métodos son bastante útiles puesto que nos ayuda a tener una visión más real del problema.

Analizando este método, podemos ver las siguientes ventajas:

- Mejora la selección de las características.
- Aporta visión sobre las relaciones entre las características
- Obtiene el conjunto óptimo de características.

En cuanto a las desventajas, diferenciamos las siguientes:

- Son métodos costosos para problemas con una alta dimensionalidad.
- Cambios dinámicos en el conjunto de variables carga el algoritmo y altera demasiado los resultados.

Podemos concluir que los métodos de envoltura son bastante eficaces pero siempre después de eliminar algunas características para reducir la dimensionalidad del conjunto porque este tipo de métodos tiene una gran carga computacional.

El último tipo de método es el **Integrado**

Estos tipos de métodos combinan cualidades de ambos métodos descritos anteriormente. Los métodos integrados tienen sus propios métodos de selección de características incorporados.

Algunos ejemplos de este tipo de métodos son:

- **Regresión Lasso:** ***"****En la regularización Lasso, también llamada L1, la complejidad C se mide como la media del valor absoluto de los coeficientes del modelo."*[13].
- **Regresión Ridge:** *"En la regularización Lasso, también llamada L1, la complejidad C se mide como la media del valor absoluto de los coeficientes del modelo. "* [13]
- **Redes Elásticas:** Combina ambos tipos de regresiones.

El proceso de trabajo de este tipo de método se puede definir de forma general de la siguiente manera:

1. Entrena un modelo de machine learning.
2. Extrae la importancia de cada característica
3. Elimina las características no importantes utilizando el paso 2.

Los métodos integrados ofrecen las siguientes ventajas:

- Rapidez, precisión y mantiene las relaciones entre las distintas características.
- Ofrece una buena cobertura para grandes escenarios de datos.

Sin embargo también tiene la desventaja de que depende del clasificador, normalmente de los árboles de decisión.

Una vez analizados los 3 métodos necesitamos ver cuales son las diferencias entre ellos. Como vemos, el único que no usa machine learning es el filtrado, por lo que no podrá decidir si una característica es esencial.

Aunque filtrado e integrado no posean mejores tiempos de ejecución, sí cuenta con un coste computacional menor al del tipo envoltura por lo que para grandes conjuntos de datos este método no es muy aconsejable.

Por otra parte, el filtrado es el único que no va a poder mostrar con seguridad el mejor conjunto cuando no hay suficientes datos para obtener una buena correlación de las características. Otro dato a tener en cuenta es que la envoltura puede llevar a un sobreajuste,, lo cual puede producir fallos importantes a la hora de obtener los resultados finales.

Referencias

[1] Funcionamiento validación cruzada. http://rstudio-pubs-static.s3.amazonaws.com/405322_6d94d05e54b24ba99438f49a6f8662a9.html

[2] Algoritmo SFS https://es.wikipedia.org/wiki/Selecci%C3%B3n_de_variable

[3] Algoritmo SFFS https://www.researchgate.net/figure/Sequential-Forward-Floating-Selection-Algorithm_fig1_221907632

[4] Pandas :Introducción a pandas. Programación en Castellano.

[5] Sklner: Installing scikit-learn — scikit-learn 0.23.1 documentation

[6] Cross_val_escore:sklearn.model_selection.cross_val_score — scikit-learn 0.23.1 documentation

[7] Tree: 1.10. Decision Trees — scikit-learn 0.23.1 documentation

[8] Time:Python time time() Method

[9] Copy: 3.15 copy -- Operaciones de copia superficial o profunda

[10] Filtrado: Hands-on with Feature Selection Techniques: Filter Methods

[11] Filtrado: Métodos de Selección de Características

[12] Filtrado y selección de subconjuntos de atributos basada en su relevancia descriptiva para la clase. https://www.researchgate.net/publication/308141950_Comenzando_con_Weka_Filtrado_y_seleccion_de_subconjuntos_de_atributos_basada_en_su_relevancia_descriptiva_para_la_clase [

[13] Regularización Lasso L1, Ridge L2 y ElasticNet https://iartificial.net/regularizacion-lasso-l1-ridge-l2-y-elasticnet/