Algoritmos de búsqueda para selección de características en modelos predictivos

María Mercedes Bañales Casarone
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
marbancas@alum.us.es
mercibanales@hotmail.com

Ana Gutman Kalichman
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
anagutkal@alum.us.es
ana.gutman2001@gmail.com

***Resumen*—El presente trabajo se focaliza en el desarrollo de algoritmos de búsqueda para la selección de características, es decir, obtener aquel subconjunto de variables predictoras que proporcione mayor información sobre la variable respuesta.**

Mediante el uso de los algoritmos de evaluación robusta, búsqueda secuencial hacia atrás y búsqueda secuencial hacia atrás mixta desarrollados, fue posible identificar un subconjunto de variables predictoras que proporcionasen más información que el conjunto completo en sí. Experimentando con diferentes hiperparámetros para los dos modelos creados -árbol de decisión y kNN-, se optimizó aún más su rendimiento.

***Palabras clave*—Inteligencia Artificial, selección de características, variables, atributos, objetivo, entrenamiento, prueba, validación por retención, validación cruzada, sobreajuste, hiperparámetros, normalización, codificación, evaluación robusta, búsqueda secuencial hacia atrás, búsqueda secuencial hacia atrás mixta, rendimiento, capacidad predictiva, árbol de decisión, kNN.**

I. INTRODUCCIÓN

En el campo de la modelización predictiva, la selección de características juega un papel fundamental para desarrollar modelos eficientes y precisos. Sin embargo, la complejidad combinatoria de la selección de características aumenta exponencialmente con el número de variables disponibles, por lo que en conjuntos de datos de gran tamaño se vuelve inviable la búsqueda exhaustiva de todas las combinaciones de características posibles. Por ende, surgen algoritmos de búsqueda que, mediante diferentes enfoques, exploran subconjuntos de características, con el objetivo de identificar aquellos que maximicen el rendimiento del modelo.

El presente trabajo se centra en la implementación y evaluación de dos variantes de algoritmos de búsqueda de tipo secuencial: hacia atrás y hacia atrás mixta. Estos algoritmos parten de un conjunto inicial con todas las variables predictoras y seleccionan características para eliminar en cada iteración con el fin de mejorar la capacidad predictiva del modelo. En el caso de la búsqueda secuencial mixta, se introduce además la posibilidad de volver a agregar variables a subconjuntos ya reducidos si ello resulta en una mejora del rendimiento del modelo.

El objetivo principal de esta propuesta es desarrollar algoritmos de selección de características generalizables y aplicables a una variedad de conjuntos de datos y problemas de clasificación o regresión. Se implementan tanto la búsqueda secuencial hacia atrás como la búsqueda secuencial mixta, son aplicados en conjuntos de datos reales orientados a la clasificación para evaluar su rendimiento.

En este documento, se describe detalladamente el proceso de implementación de los algoritmos de búsqueda, así como el método de evaluación robusta utilizado para estimar la capacidad predictiva de cada subconjunto de características. En este último se utiliza la tasa de aciertos balanceada como medida principal de rendimiento. Además, evalúan los resultados de los experimentos sobre los conjuntos de datos de

‘titanic.csv’ y ‘BreastCancer.csv’ proporcionados para validar la efectividad de los algoritmos desarrollados.

La estructura del documento se organiza en secciones que abarcan desde la introducción teórica de los algoritmos de búsqueda hasta la presentación y discusión de los resultados experimentales obtenidos.

II. PRELIMINARES

A. Métodos empleados

- Método de envoltura: es un método de selección de características; supone generar diferentes subconjuntos de variables, entrenar un modelo con cada uno y evaluar su capacidad predictiva. Se selecciona aquel con mejor rendimiento. La particularidad de este tipo de métodos es que el algoritmo de entrenamiento sí interviene en dicha selección. Para este trabajo, el modelo de aprendizaje desarrollado fue de clasificación [1] [2].
- Evaluación robusta: se utiliza para determinar el potencial predictivo de cada subconjunto de variables, con el fin de obtener aquella que resulte más prometedora. A grandes rasgos, lleva a cabo una serie de experimentos y promedia sus resultados. Para ello, aplica el método de validación cruzada sobre el modelo y subconjunto de variables en cuestión una determinada cantidad de veces. El promedio de los valores obtenidos representa la capacidad predictiva de dicho subconjunto [1].
- Búsqueda secuencial hacia atrás: es un método de selección de características que comienza con el conjunto completo de las variables y elimina una en cada iteración. Para ello, utiliza el algoritmo de evaluación robusta, seleccionando aquella variable cuya ausencia en el conjunto actual genere el mejor rendimiento. Una vez eliminadas todas, finaliza la búsqueda, retornando la mejor solución hallada en cada paso [1].
- Búsqueda secuencial hacia atrás mixta: basada en la búsqueda secuencial hacia atrás, en adición a eliminar una variable en cada iteración, brinda la posibilidad de agregar una que no se encuentra en el conjunto actual y no fue previamente añadida. Introduce el concepto de dos estrategias: la principal, hacia atrás, busca borrar aquella variable que contribuye menos al rendimiento del modelo; la secundaria, hacia delante, permite agregar una variable eliminada en caso de mejorar la capacidad predictiva del conjunto. Cuando ya no es posible hacerlo ni eliminando ni agregando variables, la solución se estabiliza y la búsqueda se da por finalizada [1].
- Técnicas de clasificación
  - Árboles de decisión: son un tipo de modelo de clasificación cuya estructura de árbol permite tomar decisiones basadas en las características de los datos. Están compuestos por nodos internos, encargados de evaluar las variables acorde a criterios de impureza, y nodos hoja, donde se realiza la predicción final de cada clase. Su fácil interpretabilidad los destaca en el ámbito del aprendizaje supervisado. Entre sus hiperparámetros se encuentran la profundidad máxima, la medida de impureza -gini o entropía- y el número mínimo de muestras necesario para dividir un nodo interno.
  - kNN: es un tipo de algoritmo de clasificación basado en los conceptos de vecinos y clase mayoritaria para realizar predicciones. Clasifica una instancia según la clase mayoritaria entre sus k vecinos más cercanos en el espacio de características. Para determinar dicha cercanía, se utiliza una métrica de distancia, comúnmente la euclídea o manhattan. Además de esta última, otros de sus hiperparámetros incluyen el número de vecinos más cercanos a contemplar, el peso que cada uno tiene y el algoritmo utilizado para computarlos. Se destaca

por su simplicidad, tanto para entenderlo como para implementarlo.

B. Trabajo Relacionado

La selección de características es un paso fundamental en las tareas de análisis de datos. Es por ello que el desarrollo de algoritmos capaces de hallar un subconjunto de variables que maximice el rendimiento del modelo de aprendizaje despierta gran curiosidad en la actualidad. Se han publicado trabajos con el mismo objetivo que el presente pero utilizando diferentes técnicas, como lo es el caso de la estudiante Radostina Spasova Dimitrova. En su proyecto de fin de grado, se centró en las técnicas de filtro en lugar de envoltura, con el objetivo de "construir un modelo capaz de predecir cuando un paciente polimedicado sufrirá un evento adverso." Como explica Dimitrova, los métodos de filtro son independientes del algoritmo de entrenamiento, es decir, no forman parte del proceso de selección de características. Trabaja con las búsquedas hacia atrás y hacia adelante, generando diferentes filtros al combinar una serie de parámetros. Resulta interesante la particularidad que le añade a estos métodos para garantizar mejoras en la ejecución de cada uno. Utilizando la librería de *Matlab*, se basa en el conteo de los atributos anteriores -respectivos a la iteración en cuestión- de modo tal que se optimice el proceso de búsqueda en sí [3].

En 2016, Anabel Toledo González publicó un trabajo en torno a la misma problemática. Su finalidad fue analizar diversos "métodos de selección de características basados en teoría de información." Tras una investigación tanto teórica como experimental, seleccionó aquel que tuviese mejor rendimiento. En su documentación, también hace referencia a las búsquedas secuencial hacia atrás y hacia adelante. Las implementó junto con técnicas de discretización para manejar datos numéricos, concluyendo que la combinación de métodos de selección de características y discretización tenía un efecto positivo en el rendimiento del modelo [4].

Inclusive, similar al presente trabajo, el método principal utilizado por Toledo González fue el algoritmo de búsqueda secuencial hacia atrás, complementado con evaluaciones robustas que asegurasen la consistencia y estabilidad de los resultados. Su enfoque le permitió identificar los atributos más informativos junto con aquellos cuya eliminación o inclusión modificaba la precisión del modelo, proporcionando un balance óptimo entre complejidad y rendimiento [4].

Otro trabajo que resulta interesante contemplar es aquel elaborado por Joaquín Pacheco Bonrostro, Silvia Casado Yusta y Laura Núñez Letamendia en 2007, enfocado en el análisis de la selección de características pero con algoritmos meméticos. La idea de estos últimos es combinar técnicas de búsqueda local con operadores de cruce o mutación, y según lo propuesto por los autores, han demostrado ser superiores a los métodos tradicionales como Stepwise, Backward y Forward [5]. Los resultados experimentales mostraron que ofrecen un rendimiento significativamente mejor en comparación con los métodos estadísticos tradicionales. De este modo, proveen una posibilidad de mejora del proyecto actual, pudiendo optimizar la selección de características para un conjunto de datos dado. Es por ello que se vuelve sumamente relevante a la consigna, ofreciendo una metodología robusta y eficiente para abordar la problemática propuesta por la guía del proyecto.

III. DECISIONES DE DISEÑO

Para asegurar el correcto funcionamiento de los algoritmos de búsqueda implementados, fue necesario tomar ciertas decisiones sobre su diseño.

En primer lugar, en cuanto a la búsqueda secuencial hacia atrás, se optó por no evaluar el rendimiento del subconjunto vacío. Es decir, cuando únicamente queda una variable por analizar, la búsqueda finaliza tras calcular su rendimiento, visto que el valor obtenido siempre será superior a aquel del subconjunto vacío. Esto supone una pequeña mejora en la eficiencia del algoritmo. Es por ello que el rango de la estructura del primer *for* en dicho algoritmo es desde el *len(solucionActual)-1* hasta *1* y no *0*. Sólo se contemplan subconjuntos con tamaños mayores a *0*.

Con respecto al método de búsqueda secuencial hacia atrás mixta, la adición de un umbral trajo consigo

ciertas complicaciones. El objetivo de este es limitar la cantidad de iteraciones que transcurren sin agregar variables una vez que todas fueron eliminadas; define la condición de parada del método. En particular, resultó desafiante lograr que, llegado el punto en que no se pueden eliminar más variables, el algoritmo no elija agregar nuevas en cada iteración. De hacerlo, no se incrementaría el contador hasta acabadas las variables para agregar y el umbral se volvería redundante. Introduciendo la variable *mejorRendimientoIteración* además de *mejorRendimiento*, fue posible garantizar que, tras haber eliminado todas las variables, sólo se añada una nueva si su rendimiento es mayor al de la iteración anterior. En caso contrario, no se añade ninguna y el contador que mantiene un registro de las iteraciones que transcurren sin añadir variables aumenta.

Otra decisión fundamental que fue necesario tomar supuso la elección de los valores para las variables *n_exp, k* y *M*. En un principio, para poder probar los algoritmos implementados con mayor facilidad y siguiendo las recomendaciones brindadas en la guía del proyecto, las variables fueron inicializadas con los valores 1, 3 y 3, respectivamente. Una vez asegurada su ejecución sin fallos y la obtención de resultados coherentes, se eligió un valor para *n_exp* entre 5 y 20, para *k* entre 5 y 10, y a *M* le asignamos el valor de 10 como se ejemplificaba en la guía. Particularmente, para *n_exp* y *k* se seleccionaron valores que permitiesen alcanzar un equilibrio entre el nivel de robustez y el de complejidad. Es decir, que fuesen lo suficientemente robustos sin requerir un tiempo de cómputo excesivo para el procesamiento de los algoritmos de búsqueda. De este modo, se definió que *n_exp* valdría 10, *k* valdría 7 y *M* valdría 10.

En adición a esto, también fue importante definir la rejilla de hiperparámetros con los cuales llevar a cabo los diferentes experimentos del algoritmo de búsqueda secuencial mixta hacia atrás. Los dos modelos de clasificación utilizados en el proyecto fueron árboles de decisión y kNN.

Para los árboles de decisión, se generaron 6 tripletas de hiperparámetros con las cuáles construir un árbol distinto. Dichas tripletas variaron en tres aspectos: criterio de impureza -gini o entropía-, máxima profundidad -3 o 5- y mínimo número de muestras necesario para dividir un nodo interno -2 o 4. Para los últimos dos, se eligieron valores relativamente bajos con el fin de evitar el sobreajuste del modelo a los datos de entrenamiento. En cuanto a las medidas de impureza, se probaron las dos que se usan con frecuencia en los árboles de decisión.

Para el algoritmo de kNN, también fueron utilizadas 6 tripletas de hiperparámetros. En esta ocasión, se fueron alterando tres características del modelo en cuestión: número de vecinos más cercanos a tener en cuenta -3 o 10-, el peso que cada uno tiene -*uniform* o *distance*- y el algoritmo utilizado para computarlos -*auto* o *ball_tree*. Para el número de vecinos más cercanos, nuevamente fue importante elegir valores que no resulten en la generación de un modelo sobreajustado a los datos de entrenamiento.

Se concluye que 6 experimentos por modelo de clasificación permiten un análisis adecuado de los resultados generados por parte de la búsqueda secuencial mixta hacia atrás. Si bien hubiese sido aún más exhaustivo generar un mayor número de combinaciones con más valores de hiperparámetros, el tiempo de cómputo y procesamiento que exigía era muy alto, por lo que se considera que se logró un buen balance.

Por último, para parametrizar el código desarrollado de tal modo que fuese genérico para cualquier conjunto de datos, se asumió que se recibiría el nombre del archivo csv de donde leer los datos, así como el nombre de la variable objetivo dentro de dicho archivo. Por ejemplo, para el problema de Titanic, los parámetros que recibe la función *ExperimentosConHiperparametrosDeAlgoritmo*, encargada de llevar a cabo los experimentos con los diferentes hiperparametros, son “titanic.csv” y “Survived”. A partir de estos dos parámetros, se preprocesan los datos, se lleva a cabo la búsqueda secuencial hacia atrás mixta y se evalúa el rendimiento de los diferentes subconjuntos de variables. De este modo, fue posible generalizar todos los algoritmos y asegurarse de que no dependieran de ningún conjunto de datos o problema concreto.

IV. METODOLOGÍA

En esta sección, se describe el método implementado en el trabajo para desarrollar y evaluar los algoritmos de búsqueda para selección de características en modelos predictivos.

Método de Evaluación Robusta:

Se diseñó un método de evaluación robusta para estimar la capacidad predictiva de cada subconjunto de características. Se realiza la evaluación mediante experimentos de validación cruzada, promediando el rendimiento obtenido en múltiples repeticiones.

EvaluacionRobusta(a, D, O, V, N_Exp, k):

Entrada: algoritmo de entrenamiento *a*, conjunto de datos completo *D*, objetivo *O,* subconjunto de variables *V*, número de experimentos *N_Exp* y número de pliegues para validación cruzada *k*.

Salida: el promedio de los resultados obtenidos en cada iteración.

```
1  resultados ← 0
2  datos ← D[V]
3  para i desde 0 a N_Exp-1
4      resultado_validación ← cross_val_score(a, D,
       O, k)
5      resultados ← resultados +
       resultado_validacion
6  devolver resultado/N_Exp
```

Algoritmos de Búsqueda Secuencial:

Se implementaron dos variantes de los mismos:

a) Búsqueda Secuencial Hacia Atrás: itera sobre cada variable, evaluando el rendimiento del modelo al eliminarla. Selecciona la variable cuya eliminación maximiza el rendimiento del modelo.

BusquedaSecuencialHaciaAtras(A, O, a, N_Exp, k)

Entrada: DataFrame con el conjunto de atributos *A,* Serie con el objetivo *O,* algoritmo de entrenamiento *a*, número de experimentos *N_Exp* y número de pliegues para validación cruzada *k*.

Salida: DataFrame *retorno* con el vector de variables de las mejores soluciones temporales, su rendimiento y tamaño.

```
1  solucionActual ← vector con los nombres de las
   variables de A
2  vectorMejoresSolucionesTemporales ← []
3  para i desde len(solucionActual)-1 a 1
4      mejorRendimiento ← -1
5      mejorSolucionTemporal ← []
6      para v desde 0 a len(solucionActual)-1
7          solucionTemporal ← copia de
           solucionActual
8          solucionTemporal ← quitar la variable
           en la posición v
9          rendimiento ←EvaluacionRobusta(a,
           A, O, solucionTemporal, N_Exp, k)
10         si rendimiento > mejorRendimiento
           entonces
11             mejorRendimiento ←
               rendimiento
12             mejorSolucionTemporal ←
               solucionTemporal
13     solucionActual ← mejorSolucionTemporal
14     vectorMejoresSolucionesTemporales ←
       agregar solucionActual, rendimiento y tamaño
15 retorno ← definir el formato de la salida
16 devolver retorno
```

b) Búsqueda Secuencial Hacia Atrás Mixta: similar al anterior, pero con la posibilidad de agregar variables previamente eliminadas. Itera sobre cada variable para eliminarla y evalúa la mejora al agregar variables previamente eliminadas. Selecciona la mejor combinación de eliminación y adición de variables que maximiza el rendimiento.

Asimismo, se implementó un control de variables visitadas para evitar bucles infinitos. Se registran las variables añadidas y eliminadas en cada iteración. Se elimina la peor variable y se añade una nueva variable si mejora el rendimiento, siguiendo un criterio de parada basado en la estabilización de la solución.

BusquedaSecuencialHaciaAtrasMixta(A, O, a, N_Exp, k, M):

Entrada: DataFrame con el conjunto de atributos *A,* Serie con el objetivo *O,* algoritmo de entrenamiento *a*, número de experimentos *N_Exp*, número de pliegues para validación cruzada *k* y número de iteraciones *M*.

Salida: DataFrame *retorno* con el vector de variables de las mejores soluciones temporales, su rendimiento y tamaño.

```
1  variablesDataset ← vector con los nombres de las
   variables de A
2  solucionActual ← copia de variablesDataset
3  añadidos ← []
4  eliminados ← []
5  vectorMejoresSolucionesTemporales ← []
6  contadorIteracionesSinAñadir ← 0
7  mejorRendimientoIteracion ← -1
8  mientras contadorIteracionesSinAñadir < M
9      mejorRendimiento ← -1
10     mejorSolucionTemporal ← []
11     EliminarVariable(solucionActual, eliminados,
       a, A, O, N_Exp, k, mejorRendimiento,
       mejorRendimientoIteracion)
12     AñadirVariable(variablesDataset,
       solucionActual, añadidos, eliminados, a, A, O,
       N_Exp, k, mejorRendimiento,
       mejorSolucionTemporal,
       contadorIteracionesSinAñadir)
13     mejorRendimientoIteracion ←
       mejorRendimiento
14     si solucionActual no se encuentra en
       vectoresMejoresSolucionesTemporales
       entonces
15         vectoresMejoresSolucionesTemporales
           ← agregar mejorSolucionTemporal,
           rendimiento y tamaño
16 retorno ← definir el formato de salida
17 devolver retorno
```

EliminarVariables(V, E, a, A, O, N_Exp, k, m, mt)

Entrada: vector con el subconjunto de variables actual *V*, vector con las variables eliminadas *E*, algoritmo de entrenamiento *a*, DataFrame con el conjunto de atributos *A*, Serie con el objetivo *O*, número de experimentos *N_Exp*, número de pliegues para validación cruzada *k*, el mejor rendimiento inicializado en -1 *m* y el mejor rendimiento hasta el momento *mt*.

Salida: no tiene retorno

```
1  peorVariableAEliminar = ''
2  para v desde 0 a len(V)-1
3      si V[v] no está en E
4          solucionTemporal ← copia de V
5          solucionTemporal ← quitar la variable
           en la posición v
6          rendimiento ← EvaluacionRobusta(a,
           A, O, solucionTemporal, N_Exp, k)
7          si rendimiento > m
           entonces
8              m ←
               rendimiento
9              mejorSolucionTemporal ←
               solucionTemporal
10             peorVariableAEliminar ←
               V[v]
11 si peorVariableAEliminar no es '' entonces
12     V ← mejorSolucionTemporal
13     eliminados ← agregar peorVariableAEliminar
14 si no entonces
15     m ← mt
```

AñadirVariables(V, S, AN, E, a, A, O, N_Exp, k, m, MST, c)

Entrada: vector con el conjunto de variables del dataset *V*, vector con el subconjunto de variables actual *S,* vector con las variables añadidas *AN*, vector con las variables eliminadas *E*, algoritmo de entrenamiento *a*, DataFrame con el conjunto de atributos *A*, Serie con el objetivo *O*, número de experimentos *N_Exp*, número de pliegues para validación cruzada *k*, el rendimiento de la mejor solución temporal *m*, un vector con la mejor solución temporal hasta el momento *MST* y un contador de iteraciones sin añadir *c*.

Salida: no tiene retorno

```
1  mejorVariableAñadir ← ''
2  para v desde 0 a len(V)-1
3      si V[v] no está en S ni en AN entonces
```

```
4               solucionTemporal ← copia de S
5               solucionTemporal ← agregar la
                variable V[v]
6               rendimiento ← EvaluacionRobusta(a,
                A, O, solucionTemporal, N_Exp, k)
7               si rendimiento > m
                entonces
8                       m ← rendimiento
9                       MST ← solucionTemporal
10                      mejorVariableAñadir ← V[v]
11  si mejorVariableAñadir no es ‘’ entonces
12      S ← MST
13      añadidos ← agregar mejorVariableAñadir
14      c ← 0
15  si no entonces
16      si ya fueron eliminadas todas las variables
        entonces
17              c ← c + 1
```

Conjuntos de Datos Utilizados:

Se aplicaron los algoritmos desarrollados en dos conjuntos de datos de clasificación indicados: ‘titanic.csv’ y ‘BreastCancer.csv’.

Tecnologías y Librerías Utilizadas

Las implementaciones se realizaron en Python utilizando bibliotecas como scikit-learn para la construcción de modelos y pandas para el manejo de datos.

V. RESULTADOS

En esta sección se detallan tanto los experimentos realizados como los resultados obtenidos.

Para evaluar la capacidad predictiva y la eficiencia de los algoritmos de búsqueda secuencial implementados, se llevaron a cabo múltiples experimentos utilizando como algoritmo a los árboles de decisión y a KNN. Los parámetros de configuración para estos experimentos fueron los siguientes:

- Número de experimentos (n_exp): 10
- Número de pliegues para cross validation (k): 7
- Umbral para estabilización (M): 10

El objetivo de los experimentos fue determinar cuál combinación de hiperparámetros produce el mejor rendimiento en términos de capacidad predictiva. Para ello primero se evaluó dicha capacidad para el conjunto completo de variables, y luego se aplicó el algoritmo de búsqueda. Una vez encontrados los hiperparámetros y el subconjunto de variables que proporcionen el mejor rendimiento, se utilizaron para construir el algoritmo de clasificación y entrenarlo con los datos de entrenamiento. Luego se evaluó su capacidad de generalización sobre los datos de prueba a partir de la matriz de confusión y la métrica Recall.

Hiperparámetros evaluados:

Árboles de decisión

- criterion: 'gini', 'entropy'
- max_depth: 3, 5, None
- min_samples_split: 2, 4

KNN:

- n_neighbors: 3, 10
- weights: uniform, distance
- algorithm: auto, ball_tree

En la *Figura 1* a continuación se presenta una tabla que contiene el rendimiento bajo la evaluación robusta de usar el conjunto completo de las 15 variables en titanic.csv junto con el mejor subconjunto encontrado por el algoritmo de búsqueda hacia atrás mixta, su rendimiento y tamaño. Esta es seguida por la *Figura 2*, en la cual se presentan los mismos resultados pero para el set de datos de 30 variables en BreastCancer.csv. De este modo, ambas tablas permiten evaluar si efectivamente es favorable al rendimiento de las predicciones optar por elegir un subconjunto de las variables disponibles mediante nuestro algoritmo de búsqueda secuencial.

			Mejor subconjunto de variables obtenido por la búsqueda mixta hacia atrás		
	TITANIC	Rendimiento con las 15 variables	Rendimiento	Subconjunto	Tamaño
ÁRBOLES DE DECISIÓN: (criterion, max_depth, min_samples_split)	(gini, 3, 2)	0.8073397928236637	0.818967	[Pclass, Fare, Initial, Family_Size]	4
	(gini, 5, 4)	0.8037881984886592	0.811454	[Pclass, Sex, Age, SibSp, Parch, Fare, Embarked, Initial, Age_band, Fare_cat, Deck, Is_Married, A	13
	(gini, none, 2)	0.7517563849361084	0.796154	[Pclass, SibSp, Fare, Embarked, Initial, Title, Family_Size, Alone, Fare_cat]	9
	(entropy, 3,2)	0.802699988183859	0.822157	[Pclass, Fare, Initial]	3
	(entropy, 5,4)	0.7977369276908446	0.819671	[Pclass, Sex, Age, SibSp, Fare, Initial, Age_band, Is_Married]	8
	(entropy, none,2)	0.749403706892186	0.785400	[Pclass, Age, Fare, Initial, Age_band, Family_Size, Alone, Fare_cat, Title]	9
KNN: (n_neighbors, weights, algorithm)	(3, uniform, auto)	0.7912284847768716	0.825512	[Pclass, Sex, Age, SibSp, Fare, Initial, Age_band, Family_Size, Alone, Fare_cat, Deck, Is_Married]	12
	(10, uniform, auto)	0.7749601908588082	0.805555	[Pclass, Sex, Age, SibSp, Parch, Fare, Embarked, Initial, Age_band, Family_Size, Alone,	12
	(3, distance, auto)	0.7786653950248421	0.810878	[Pclass, Sex, Age, SibSp, Initial, Age_band, Fare_cat, Deck, Is_Married]	9
	(10, distance, auto)	0.7816855443583552	0.811773	[Pclass, Sex, SibSp, Parch, Embarked, Age_band, Family_Size, Fare_cat, Deck, Title]	10
	(3, uniform, ball_tree)	0.7912284847768716	0.825512	[Pclass, Sex, Age, SibSp, Fare, Initial, Age_band, Family_Size, Alone, Fare_cat, Deck, Is_Married]	12
	(10, uniform, ball_tree)	0.7749601908588082	0.805555	[Pclass, Sex, Age, SibSp, Parch, Fare, Embarked, Initial, Age_band, Family_Size, Alone,	12

Figura 1: Tabla comparativa del rendimiento del conjunto completo de variables en titanic.csv con el rendimiento del mejor subconjunto de variables encontrado por el algoritmo de búsqueda secuencial hacia atrás

	BREAST CANCER	Rendimiento con las 30 variables	Rendimiento	Subconjunto	Tamaño
ÁRBOLES DE DECISIÓN: (criterion, max_depth, min_samples_split)	(gini, 3, 2)	0.9301117886178863	0.966854	[mean texture, mean concave points, area error, compactness error, concavity error, worst radius, worst texture, worst symmetry, mean area, mean compactness, symmetry error]	11
	(gini, 5, 4)	0.9386146922183508	0.959813	[mean texture, mean perimeter, mean area, mean concave points, mean fractal dimension, symmetry error, fractal dimension error, worst texture, worst area, worst smoothness, worst concave points, worst fractal dimension, worst perimeter, mean symmetry, mean compactness, worst symmetry, mean concavity, worst compactness, texture error]	19
	(gini, none, 2)	0.938927700348432	0.962872	[mean radius, mean texture, mean perimeter, mean area, mean concavity, mean fractal dimension, area error, symmetry error, worst texture, worst area, worst smoothness, worst compactness, worst concave points, texture error, mean symmetry, mean compactness, mean smoothness, worst symmetry]	18
	(entropy, 3,2)	0.9335307781649245	0.970234	[mean texture, mean smoothness, mean compactness, mean concavity, mean concave points, texture error, perimeter error, area error, smoothness error, symmetry error, worst texture, worst perimeter, worst area, worst compactness, mean fractal dimension, mean perimeter]	16
	(entropy, 5,4)	0.9352700348432055	0.970087	[mean texture, mean area, mean concavity, mean concave points, perimeter error, area error, symmetry error, worst texture, worst area, worst compactness, mean compactness, texture error, worst symmetry, worst concavity, mean smoothness, worst smoothness]	16
	(entropy, none,2)	0.936824912891986	0.968095	[mean symmetry, worst symmetry, worst smoothness, compactness error, mean compactness, smoothness error, area error, worst radius, worst compactness, mean fractal dimension, worst concavity, mean smoothness, mean area, worst area, mean concavity, mean texture, mean concave points, perimeter error, worst texture, symmetry error, concave points error, texture error]	22
KNN: (n_neighbors, weights, algorithm)	(3, uniform, auto)	0.9572415795586527	0.979016	[mean texture, mean perimeter, mean area, mean smoothness, mean concavity, mean concave points, mean fractal dimension, radius error, texture error, perimeter error, area error, smoothness error, compactness error, concavity error, concave points error, symmetry error, worst radius, worst texture, worst perimeter, worst area, worst compactness, worst concavity, worst symmetry]	23
	(10, uniform, auto)	0.9742973286875726	0.978897	[mean radius, mean texture, mean perimeter, mean area, mean smoothness, mean compactness, mean concavity, mean concave points, texture error, smoothness error, compactness error, symmetry error, worst radius, worst texture, worst perimeter, worst area, worst smoothness, worst compactness, worst concavity, worst concave points, worst symmetry, radius error] (**)	22
	(3, distance, auto)	0.9572415795586527	0.979016	[mean texture, mean perimeter, mean area, mean smoothness, mean concavity, mean concave points, mean fractal dimension, radius error, texture error, perimeter error, area error, smoothness error, compactness error, concavity error, concave points error, symmetry error, worst radius, worst texture, worst perimeter, worst area, worst compactness, worst concavity, worst symmetry]	23
	(10, distance, auto)	0.9672299651567944	0.981873	[mean texture, mean perimeter, mean smoothness, mean concave points, mean symmetry, texture error, smoothness error, compactness error, concave points error, fractal dimension error, worst radius, worst texture, worst perimeter, worst area, worst concavity, worst concave points, worst fractal dimension]	17
	(3, uniform, ball_tree)	0.9572415795586527	0.979016	[mean texture, mean perimeter, mean area, mean smoothness, mean concavity, mean concave points, mean fractal dimension, radius error, texture error, perimeter error, area error, smoothness error, compactness error, concavity error, concave points error, symmetry error, worst radius, worst texture, worst perimeter, worst area, worst compactness, worst concavity, worst symmetry]	23
	(10, uniform, ball_tree)	0.9742973286875726	0.978897	[mean radius, mean texture, mean perimeter, mean area, mean smoothness, mean compactness, mean concavity, mean concave points, texture error, smoothness error, compactness error, symmetry error, worst radius, worst texture, worst perimeter, worst area, worst smoothness, worst compactness, worst concavity, worst concave points, worst symmetry, radius error] (**)	22

Figura 2: Tabla comparativa del rendimiento del conjunto completo de variables en BreastCancer.csv con el rendimiento del mejor subconjunto de variables encontrado por el algoritmo de búsqueda secuencial hacia atrás

El mejor subconjunto de características es aquel que proporcione un balance entre minimizar la cantidad de características y maximizar el rendimiento. Con esto en consideración y para dar prioridad a los resultados de rendimiento obtenidos a partir del método de evaluación robusta desarrollado para este trabajo, se consideró como el mejor subconjunto a aquel con el mayor rendimiento de todos. En el caso de que dicho valor coincida para más de un subconjunto de variables, se escogió aquel con menor tamaño. Luego, existieron casos en los que varios subconjuntos de variables distintos pero de igual tamaño demostraron el mismo rendimiento, los cuales fueron marcados en las tablas con dos asteriscos al final (**). Esto indica que tanto el subconjunto mostrado en la tabla cómo los otros que cumplen las condiciones mencionadas son considerados indistintamente como los mejores subconjuntos para ese set de hiperparámetros.

La primera conclusión saliente de los resultados de los experimentos es que para todos los sets de hiperparámetros probados en cada algoritmo de clasificación, se obtuvo un mayor rendimiento usando el subconjunto de variables obtenido por la búsqueda secuencial mixta hacia atrás que usando todas las variables disponibles. Asimismo, todos los subconjuntos de variables obtenidos son de menor tamaño que el conjunto completo original. Por ende, se comprueba la efectividad del método para estos conjuntos de datos.

Para identificar qué hiperparámetros fueron los más eficientes se podrían evaluar muchas condiciones. Por ejemplo, la diferencia entre el rendimiento de considerar todas las variables con el rendimiento del set generado en la búsqueda secuencial, o el tamaño de dicho set. Sin embargo, se mantuvo el criterio de considerar mejor a aquel que generó el máximo rendimiento.

En el caso de los árboles de decisión podemos ver que para el conjunto de datos de titanic.csv, el máximo rendimiento obtenido fue de 0.822157 con los hiperparámetros: criterion=entropy, max_depth=3 y min_samples_split=2, en un subconjunto de 3 variables. La medida de recall para el árbol construido con dichos hiperparametros y entrenado con los datos de entrenamiento correspondientes a dichas variables fue 0.7101449275362319. La matriz de confusión obtenida se observa en la *Figura 3*.

y \ $\hat{y}$	0	1
0	97	13
1	20	49

Figura 3: Matriz de confusión para árbol de decisión de titanic.csv

Luego, en BreastCancer.csv se obtuvo un rendimiento máximo de 0.970234 para el mismo conjunto de hiperparámetros que en los datos de titanic y un subconjunto de 16 variables. El valor de recall correspondiente fue de 0.9861111111111112 y la matriz de confusión se presenta en la siguiente figura:

y \ $\hat{y}$	0	1
0	37	5
1	1	71

Figura 4: Matriz de confusión para árbol de decisión de BreastCancer.csv

A partir de los resultados obtenidos, se evidencia que el valor de recall para los datos de titanic.csv es significativamente menor que para los de BreastCancer.csv. Cabe destacar que dicho valor juega un rol más importante en el caso del diagnóstico de cáncer de mamá, visto que trata con temas de salud personal. Es preferible que el modelo diagnostique un caso de cáncer de mama que en realidad no lo sea -False Positive-, a que no diagnostique a un paciente que sí lo padezca -False Negative-. Esto también se visualiza en la *Figura 4*, donde el número de False Positives (5) fue mayor que el de False Negatives (1),

mientras que en la *Figura 3* sucede lo contrario -el valor de False Negative (20) fue mayor que el de False Positive (13). Por ende, consideramos que los resultados son satisfactorios.

Para el algoritmo de kNN, para el conjunto de datos de titanic.csv, el máximo rendimiento obtenido fue de 0.825512 para dos subconjuntos de hiperparametros. Con la función *obtener_mejor_solucion(dataframe),* se selecciona el primero: n_neighbors=3, weights=uniform y algorithm=auto. Su tamaño es de 12 variables. La medida de recall para el modelo construido con dichos hiperparámetros y entrenado con los datos de entrenamiento correspondientes a dichas variables fue 0.7391304347826086. La matriz de confusión obtenida se observa a continuación:

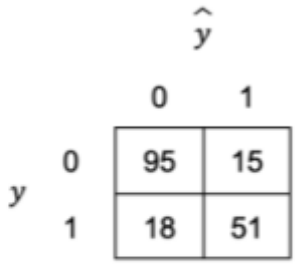

Figura 5: Matriz de confusión para kNN de titanic.csv

Finalmente, en el caso de BreastCancer.csv se obtuvo un rendimiento máximo de 0.981873 para un subconjunto de 17 variables con los siguientes hiperparámetros: n_neighbors=10, weight=distance y algorithm=auto. El recall score del knn correspondiente fue de 0.9861111111111112 y su matriz de confusión se presenta en la siguiente *Figura 6*:

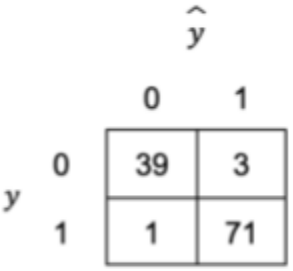

Figura 6: Matriz de confusión para kNN de BreastCancer.csv

Podemos notar que para ambos conjuntos de datos, los resultados de recall fueron mayores en los algoritmos kNN que en los árboles de decisión. Asimismo, se repite que dicha métrica es más favorable para los datos de breast cancer que para los de titanic. El análisis realizado para el caso de los árboles de decisión es análogo.

VI. CONCLUSIONES

A modo de cierre, para abordar la problemática de la selección de características en modelos predictivos, el presente trabajo se centró en el desarrollo de dos algoritmos de búsqueda reconocidos en el ámbito de la Inteligencia Artificial: búsqueda secuencial hacia atrás y búsqueda secuencial hacia atrás mixta. De este modo, fue posible obtener un subconjunto de variables que proporcionasen un mejor rendimiento que el conjunto completo en sí, utilizando dos archivos de datos -titanic.csv y BreastCancer.csv- para ello. En adición a esto, se llevaron a cabo una serie de experimentos variando los hiperparámetros de los modelos de clasificación con el fin de seguir optimizando su capacidad predictiva. Una vez elegidos los mejores hiperparámetros y el mejor subconjunto de variables, fueron utilizados para entrenar los respectivos modelos -árboles de decisión y kNN- con los datos de entrenamiento previamente procesados. Por último, se evaluó la capacidad de generalización de dichos modelos con los datos de prueba preprocesados con la métrica de recall y la matriz de confusión.

Tras un análisis exhaustivo de los resultados obtenidos, es posible concluir que, a través de un método de evaluación robusta basado en validación cruzada para obtener medidas de rendimiento, se hallaron numerosos subconjuntos de variables con una mejor capacidad predictiva que el conjunto completo. También se comprobó el efecto que los hiperparámetros elegidos tienen sobre dicho rendimiento, generando resultados variados acorde al nivel de sobreajuste que imponen sobre los datos de entrenamiento.

La selección de características trae consigo diversas ventajas, entre ellas destacándose la simplicidad, interpretabilidad y eficiencia computacional que añaden a un modelo de aprendizaje.

Los algoritmos propuestos y evaluados en el proyecto en cuestión evidencian los beneficios de trabajar con un número reducido de variables en lugar de todo el conjunto de datos.

Como fue contemplado en la sección II. PRELIMINARES, futuras investigaciones podrían explorar la combinación de los algoritmos de búsqueda con otras técnicas de selección de características y su aplicación en diferentes tipos de modelos y dominios de datos.

REFERENCIAS

[1] J. Galán Páez. "Algoritmos de búsqueda para selección de características en modelos predictivos." Abril, 2024.

[2] "Métodos de envoltura en la selección de características | Conceptos de Aprendizaje Automático." Marzo, 2024.
Extraído de:
https://www.toolify.ai/es/ai-news-es/mtodos-de-envoltura-en-la-seleccin-de-caractersticas-conceptos-de-aprendizaje-automtico-2457389

[3] R. S. Dimitrova. "Desarrollo y evaluación de métodos de selección de características para la predicción de eventos adversos en pacientes polimedicados." Mayo, 2017.
Extraído de:
https://academica-e.unavarra.es/xmlui/bitstream/handle/2454/24594/memoriaTFG_spasova78342.pdf?sequence=1&isAllowed=y

[4] A. Toledo González. "Métodos de selección de atributos para clasificación supervisada basados en teoría de información." Junio, 2016.
Extraído de:
https://www.researchgate.net/publication/331155838_Metodos_de_seleccion_de_atributos_para_clasificacion_supervisada_basados_en_teoria_de_informacion

[5] J. Pacheco Bonrostro, S. Casado Yusta y L. Núñez Letamendia. "Algoritmos meméticos para selección de variables en el análisis discriminante." 2007.