# Algoritmos de búsqueda para selección de características en modelos predictivos

Hugo Borrego Angulo
*dpto. Ciencias de la Computación e Inteligencia Artificial*
*Universidad de Sevilla*
Sevilla, España
hugborang@alum.us.es

Rafael Duque Colete
*dpto. Ciencias de la Computación e Inteligencia Artificial*
*Universidad de Sevilla*
Sevilla, España
rafduqcol@alum.us.es

**En este trabajo se presenta una propuesta para la selección de características, basada en la utilización de dos algoritmos de búsqueda: *búsqueda secuencial hacia atrás* y *búsqueda secuencial hacia atrás mixta*. Estos algoritmos se han aplicado a dos conjuntos de datos referentes a: *titanic [1]* y *breast cancer [2]*. Una vez obtenidos los resultados y con ello las características seleccionadas, se han aplicado 2 algoritmos de entrenamiento: *DecisionTreeClassifier [3]* y *kNN*. Con los que podemos evaluar las soluciones que nos proporcionan los algoritmos de búsqueda anteriormente mencionados.**

**Al final podremos ver que las búsquedas mixtas ofrecen mejor rendimiento, pero no tiene porque ser el mejor subconjunto de variables**

***Palabras Claves* - Algoritmo de búsqueda, Algoritmo de entrenamiento, Búsqueda Secuencial, Evaluación Robusta, Inteligencia Artificial, Rendimiento**

## I. Introducción

En la actualidad, la selección de características es una técnica fundamental en el análisis de datos y el aprendizaje automático utilizado en el campo de la Inteligencia Artificial. La selección de características nos ayuda en el preprocesamiento de datos, ya que ayuda a reducir la dimensionalidad de los datos, mejorar la tasa de acierto de los modelos y facilitar la interpretación de los resultados.

Una vez que los datos han sido adecuadamente preprocesados, la selección de características nos ayudará a mejorar la eficiencia y la interpretabilidad de los modelos. Los algoritmos de búsqueda secuencial hacia atrás y búsqueda secuencial hacia atrás mixta son técnicas para la selección de estas características. Estos algoritmos eliminan iterativamente las características menos relevantes, con el objetivo de identificar un subconjunto óptimo de características que maximicen el rendimiento predictivo.

En este trabajo, se aplica un enfoque sistemático para analizar y comparar el rendimiento de dos modelos de clasificación.

Se usarán los algoritmos: *DecisionTreeClassifier* y el modelo *kNN* para entrenar los modelos. En primer lugar sobre todo el conjunto de las variables predictoras para ver la tasa de acierto, la matriz de confusión y sensibilidad del conjunto de pruebas que se obtiene sobre todo el conjunto de variables, para luego hacer un contraste con los resultados obtenidos de los algoritmos de selección de características, y ver si estos ofrecen mejores métricas.

Los resultados obtenidos proporcionarán una relación entre diferentes subconjuntos de variables predictoras y su rendimiento, con el objetivo de encontrar el subconjunto de variables predictoras que guarde un mayor compromiso entre su rendimiento y el número de variables seleccionadas.

La estructura del documento es la siguiente: Primero, se aborda el tratamiento de los datos, que incluye normalización y codificación de variables categóricas. Luego, se describen los algoritmos de selección de características utilizados: *búsqueda secuencial hacia atrás y búsqueda secuencial hacia atrás mixta*. A continuación, se presentan los algoritmos de entrenamiento empleados (*DecisionTreeClassifier* y *kNN*) y el método de evaluación robusta mediante validación cruzada. Posteriormente, se detalla la metodología de implementación y evaluación de los algoritmos, seguido de los resultados obtenidos aplicados a los conjuntos de datos. Finalmente, se discuten los resultados y se aportan conclusiones.

## II. Preliminares

Para comprender el trabajo realizado, es necesario conocer los conceptos básicos de la selección de características y los algoritmos que han sido utilizados en este trabajo.

*A. Tratamiento de datos*

1. *Normalización:* Se ha aplicado la normalización a los datos que era necesario, con el objetivo de que todas las variables tengan la misma escala.
2. *Codificación de variables categóricas:* Se ha aplicado la codificación de variables categóricas a los datos que era necesario, con el objetivo de que todas las variables sean numéricas.

*B. Algoritmo de selección de características*

1. *Búsqueda secuencial hacia atrás:* Este algoritmo elimina iterativamente las características menos relevantes, con el objetivo de identificar un subconjunto óptimo de características que maximicen el rendimiento predictivo.
2. *Búsqueda secuencial hacia atrás mixta:* Este algoritmo elimina iterativamente las características menos relevantes, y añade las características más relevantes si mejoran el rendimiento, con el objetivo de identificar un subconjunto óptimo de características que maximicen el rendimiento predictivo.

*C. Algoritmos de entrenamiento*

1. *DecisionTreeClassifier:* Este algoritmo es un método de aprendizaje supervisado que utilizaremos para ver la calidad de las soluciones y compararlas con el conjunto de variables predictoras completo.

2. *kNN:* Este algoritmo es un método de aprendizaje supervisado que también nos ayudará a comparar los resultados y sacar nuestras propias conclusiones. El modelo kNN define el concepto de similitud entre ejemplos a partir del concepto de cercanía.

*D. Robust Evaluation*

*Método robust evaluation:* Este método devuelve el rendimiento de un subconjunto de variables predictoras haciendo uso del método *cross_val_score* de la librería Scikit-learn.

## III. Metodología

Para el desarrollo de este trabajo se han implementado los algoritmos de búsqueda secuencial hacia atrás y búsqueda secuencial hacia atrás mixta.

Todo esto en el lenguaje de programación Python, y con la ayuda de las librerías de Scikit-learn, Pandas y Numpy.

A. *Tratamiento de datos*

Para que nuestros algoritmos puedan ser aplicados a un conjunto de datos primero será necesario tratar dichos datos.

1. Se dividen los atributos según su tipo:
    a. Atributos discretos
    b. Atributos continuos
2. Se define el objetivo, y se aplican el siguiente tratamiento sobre las diferentes variables si es necesario:
    a. Normalización: Se unifican entre [0, 1] a las variables que sea necesario.
    b. Codificación de variables categóricas: Se codifican numéricamente los atributos discretos cuya información viene expresada en forma de texto.

B. *Evaluación del modelo*

Para ello se hace uso de los siguientes algoritmos de entrenamiento:

1. *DecissionTreeClasiffier*
2. *kNN*

Se dividirán los datos en 2: conjunto de entrenamiento y de pruebas 80% y 20% respectivamente, para luego ver la tasa de acierto, matriz de confusión y sensibilidad del conjunto de pruebas.

Para mejorar el rendimiento de los algoritmos se aplicará una búsqueda en rejilla (*GridSearchCV* [4]) para encontrar los mejores valores para:

- Profundidad máxima
- mínimo de muestras requeridas
- kNN vecinos
- métrica(Manhattan, euclídea)

C. *Método Robust Evaluation*

Para ambos algoritmos de búsqueda se ha usado la función *robustEvaluation* que hace uso del método *cross_val_score* [5] de la librería Scikit-learn, esta función nos permitirá realizar evaluaciones mediante validación cruzada [6].

Respecto a la función *robustEvaluation* está nos permitirá evaluar el rendimiento de los subconjuntos de variables predictoras seleccionadas, siendo necesario para su uso:

- *X*: El conjunto de datos con todas las variables predictoras.
- *y*: El conjunto de datos con la variable objetivo.
- *model*: El modelo de clasificación que se va a evaluar.
- *N_Exp*: El número de repeticiones del experimento por validación cruzada.
- *cV*: El número de pliegues del conjunto de datos para la validación cruzada.

```
1 Entrada: DataFrame X,  Nombre y, RandomForestClassifier model,
        Entero N_Exp,  Entero CV
2 Salida: un valor de rendimiento promedio del modelo
3 ResultadosTotales ← vacía
4 hacer N_Exp veces:
5       resultado ← cross_val_score(model, X, y, cv=CV,
                scoring='balanced_accuracy', n_jobs=-1)
7       ResultadosTotales ← ResultadosTotales + resultado
8 devolver media(ResultadosTotales)
```

Fig. 1. PseudoCódigo RobustEvaluation

Está función devuelve el promedio de las N_Exp evaluaciones realizadas por la función **cross_val_score**, usando como medida de rendimiento la tasa de acierto balanceada [7].

D. *Búsqueda secuencial hacia atrás*

Esta función nos proporcionará un DataFrame con los subconjuntos de variables predictoras ordenadas por rendimiento, su rendimiento, y el número de variables predictoras seleccionadas.

- *X*: El conjunto de datos con todas las variables predictoras y la variable objetivo.
- *objective*: El nombre de la variable objetivo.
- *model*: El modelo de clasificación que se va a evaluar.
- *N_Exp*: El número de repeticiones del experimento por validación cruzada.
- cV: El número de pliegues del conjunto de datos para la validación cruzada.

```
Entrada: Data Frame data, Nombre objective,
RandomForest Classifier,model, Entero N_Exp, Entero cV
Salida: un DataFrame con las combinaciones obtenidas en cada iteración,
 su tamaño y su rendimiento.
1  Variables ← data.columns()
2  Variables ← Variables - objective
3  Solución Actual ← Variables
4  y ← data[objective]
5  Results ← vacío
6  K ← tamaño de Solución Inicial
7  cada k desde K hasta 1 hacer
8       MejorRendimiento ← -inf
9       PeorVariable← vacía
10      cada V en SoluciónActual hacer
11              SoluciónTemporal ← SoluciónActual - V
12              X Temp ← data[SoluciónTemporal]
13              si SoluciónTemporal no tiene elementos entonces
14                      seguir con el siguiente V
15              Rendimiento ← evaluaciónRobusta(X Temp, y, model,
                                                N_Exp, Cv)
16              si Rendimiento > MejorRendimiento entonces
17                      MejorRendimiento ← Rendimiento
18                      PeorVariable ← V
19      si Peor Variable está vacía entonces
20              terminar bucle
21      SoluciónActual ← SoluciónActual - PeorVariable
22      Results ← agregar(Results, SoluciónActual, tamaño(SoluciónActual),
                          MejorRendimiento)
23 RendimientoTodasVariables ← evaluaciónRobusta(data[Variables], y,
                                model, N_Exp, Cv)
24 Results ← agregar(Results, Variables, tamaño(Variables),
                     RendimientoTodasVariables)
25 DataFrame ← convertDataFrame(Results)
26 DataFrame ← ordenarPorRendimiento(DataFrame)
27 devolver DataFrame
```

Fig. 2. PseudoCódigo búsqueda secuencial hacia atrás

Para este algoritmo en primer lugar, se inicializa el conjunto de variables predictoras con todas las variables predictoras, y otro con la variable respuesta. Posteriormente se elimina de las variables la variable predictora, además se crea una lista vacía que contendrá el resultado, y una solución actual que será el conjunto de las variables, que se irá actualizando en cada iteración.

Se realizará el siguiente proceso k veces, siendo k el número de variables predictoras:

1. Se eliminan las variables de la solución actual una por una.
2. Se aplicará la función robustEvaluation sobre el nuevo conjunto de variables predictoras..
3. Se guardará el mejor rendimiento de entre todos los subconjuntos de variables predictoras, además de la peor variable.
4. Se elimina de la solución actual la peor variable.
5. Se añade la solución actual al resultado.
6. Se vuelve al inicio del bucle si este no ha sido ejecutado k veces.

Por último añadimos al resultado el rendimiento del conjunto de variables predictoras completo y devolvemos un *DataFrame* con el resultado ordenado por el rendimiento de las soluciones.

E. *Búsqueda secuencial hacia atrás mixta*

Esta función nos proporcionará un *DataFrame* con los subconjuntos el de variables predictoras, su rendimiento, y número de variables predictoras seleccionada, para su uso es necesario:

- *X*: El conjunto de datos con todas las variables predictoras y la variable objetivo.
- *objective*: El nombre de la variable objetivo.
- model: El modelo de clasificación que se va a evaluar.
- *N_Exp*: El número de repeticiones del experimento por validación cruzada.
- *cV*: El número de pliegues del conjunto de datos para la validación cruzada.
- *M*: Número de iteraciones una vez se hayan eliminado todas las variables predictoras, y además no se añade ninguna variable predictora.

```
Entrada: Data Frame data, Nombre objective,
RandomForestClassifier,model, Entero N Exp, Entero cV, Entero M
Salida: un DataFrame con las combinaciones obtenidas en cada iteración,
        su tamaño y su rendimiento.
1  Variables ← data.columns()
2  Variables ← Variables - objective
3  y ← data[objective]
4  SoluciónActual ← vacía
5  Eliminados ← vacía
6  Añadidos ← vacía
7  PeorVariable ← vacía
8  contador ← 0
9  mientras len(Eliminados) distinto len(Variables) o contador < M hacer
10       si longitud de Eliminados es igual a la de Variables entonces
11              contador ← contador + 1
12       Mejor Rendimiento ← -inf
13       Mejor Solución Temporal ← vacía
14       cada V en Solución Actual hacer
15              si V no está en Eliminados entonces
16                      Solución Temporal ← Solución Actual - V
17                      Xtemp ← data[Solución Temporal]
18                      Rendimiento ← evaluacionRobusta(Xttemp, y,
                                model, N_Exp, Cv, M)
19                      si Rendimiento > Mejor Rendimiento entonces
20                              Mejor Rendimiento ← Rendimiento
21                              Peor Variable ← V
22       si PeorVariable no está vacía entonces
23              SolucionActual ← SolucionActual - PeorVariable
24              Eliminados ← Eliminados + PeorVariable
25       MejorVariable ← vacía
28       cada V en Variables hacer
29              si V no está en SoluciónActual ni en Añadidos entonces
30                      SoluciónTemporal ← SoluciónActual + V
30                      Xtemp ← data[SoluciónTemporal]
31                      Rendimiento ← evaluacionRobusta(Xtemp, y, model,
                                N_Exp, Cv, M)
32                      si Rendimiento > MejorRendimiento entonces
                                MejorVariable ← V
33                              MejorRendimiento ← Rendimiento
35       si mejorVariable no está vacía entonces
              SolucionActual ← SolucionActual + MejorVariable
37            Añadidos ← Añadidos + MejorVariable
38       si MejorRendimiento distinto a -inf entonces
40              Results← agregar(Results← , MejorSoluciónTemporal,
                                tamaño(MejorSoluciónTemporal))
41       DataFrame ← convertirADataFrame(Results)
42       DataFrame ← ordenarPorRendimiento(DataFrame)
43       devolver DataFrame
```

Fig. 3. PseudoCódigo búsqueda secuencial hacia atrás mixta

En primer lugar, se inicializa el conjunto de variables con todas las variables predictoras, y otro con la variable respuesta. Se elimina de las variables la variable respuesta además se crea una lista vacía que contendrá el resultado, y una solución actual que contendrá las variables predictoras, que se irá actualizando cuando proceda. También se crea una lista de variables añadidas y otra de variables eliminadas.

Luego mientras se cumpla la condición de parada, la cuál será que todas la variables predictoras están en la lista eliminadas, cuando esto se cumpla el bucle debe realizar al menos M iteraciones, siempre y cuando no se añada ninguna variable predictora, si se llega a añadir una variable predictora M volverá a 0, teniendo que llegar otra vez a M iteraciones.

Mientras la condición de parada no se cumpla:

1. Se eliminan variables de la solución actual una por una.
2. Se aplicará la función *robustEvaluation* sobre el conjunto de variables predictoras actualizado.
3. Se guardará el mejor rendimiento de entre todos los subconjuntos de variables predictoras, además de la peor variable.
4. Se elimina de la solución actual la peor variable y se añade a la lista de eliminadas.
5. Se añaden variables de entre todas las variables a la solución actual una por una, siempre que estas no estén ni en la lista de eliminadas ni en la solución actual.
6. Se aplicará la función *robustEvaluation* sobre el conjunto de variables predictoras actualizado.
7. Se guardará el mejor rendimiento de entre todos los subconjuntos de variables predictoras si este supera el rendimiento ya calculado anteriormente, además de la mejor variable.
8. Por último se añade la solución actual al conjunto de variables que proporcionó el mejor rendimiento al resultado.
9. Devolvemos un *DataFrame* con el resultado, ordenado por el rendimiento.

F. *Algoritmos de entrenamiento*

1. *DecisionTreeClasiffier [8]:* Es un algoritmo de aprendizaje supervisado. Este algoritmo funciona construyendo un árbol de decisiones a partir de los datos de entrenamiento. Cada nodo del árbol representa una característica de las variables predictoras, y cada bifurcación representa una regla de decisión basada en dicha característica.
2. *KNN [9]:* Es un método de aprendizaje supervisado, clasifica un punto de datos según la mayoría de las etiquetas de sus vecinos más cercanos. El "K" en KNN representa el número de vecinos más cercanos que se consideran para la clasificación.

Una vez tenemos los subconjuntos de variables predictoras, usaremos estos algoritmos para poder ver la calidad que tienen los subconjuntos proporcionados por los algoritmos de búsqueda, y ver si la tasa de acierto, la matriz de confusión y la sensibilidad del conjunto de pruebas de estos subconjuntos son valores similares y ver que proporcionan unos buenos valores, pero usando menos variables por lo que el coste computacional será mucho menor al eliminar variables que aportan poca o ninguna información.

## IV. Resultados

Una vez entendemos el proceso para la selección de características y conocemos las soluciones proporcionadas por ambos algoritmos de búsqueda, podemos ver la calidad de nuestras soluciones, comparando las métricas que ofrecen el conjunto de variables completo y las métricas que ofrecen los subconjuntos de variables.

Además de la tasa de acierto balanceada, se han realizado diferentes experimentos con el conjunto completo de variables, para luego comparar esas métricas con los subconjuntos de variables que ofrecen los algoritmos de búsqueda, estas métricas son:

- Tasa de acierto
- Matriz de confusión
- Sensibilidad del conjunto de pruebas

Recordemos que el mejor subconjunto de características no tiene porque ser aquel que tenga mayor rendimiento sino el que consiga un equilibrio entre un bajo número de características y el mayor rendimiento posible.

En las siguientes figuras (4, 5, 6, 7) se muestran los subconjuntos, su tamaño y el rendimiento asociado a ese subconjunto.

### A. *Titanic*

Soluciones proporcionadas por el algoritmo de búsqueda secuencial hacia atrás.

| | variables | size | score |
|---|---|---|---|
| 2 | [Initial, Pclass, Fare] | 3 | 0.819838 |
| 3 | [Initial, Is_Married, Pclass, Fare] | 4 | 0.819430 |
| 4 | [Initial, Is_Married, Pclass, Embarked, Fare] | 5 | 0.817972 |
| 5 | [Initial, Is_Married, Pclass, Embarked, Alone,... | 6 | 0.815888 |
| 6 | [Initial, SibSp, Is_Married, Pclass, Embarked,... | 7 | 0.813601 |
| 7 | [Initial, SibSp, Is_Married, Pclass, Parch, Em... | 8 | 0.811819 |
| 9 | [Initial, SibSp, Deck, Is_Married, Pclass, Par... | 10 | 0.810012 |
| 10 | [Initial, SibSp, Deck, Is_Married, Pclass, Par... | 11 | 0.809098 |
| 1 | [Initial, Fare] | 2 | 0.808618 |
| 8 | [Initial, SibSp, Deck, Is_Married, Pclass, Par... | 9 | 0.808402 |
| 11 | [Initial, SibSp, Deck, Fare_cat, Is_Married, P... | 12 | 0.808332 |
| 12 | [Initial, SibSp, Deck, Fare_cat, Sex, Is_Marri... | 13 | 0.805844 |
| 14 | [Initial, SibSp, Deck, Fare_cat, Title, Sex, I... | 14 | 0.804816 |
| 13 | [Initial, SibSp, Deck, Fare_cat, Title, Sex, I... | 15 | 0.799331 |
| 0 | [Initial] | 1 | 0.783354 |

Fig. 4. Solución Titanic búsqueda secuencial hacia atrás

Se puede ver que al reducir el número de variables se puede obtener un mejor rendimiento, en este caso el subconjunto de tamaño 3 reduce considerablemente el número de variables predictoras, traduciéndose en un menor coste computacional.

| | variables | size | score |
|---|---|---|---|
| **14** | [Initial, Embarked, Is_Married, Title, SibSp, ... | 9 | 0.820649 |
| **9** | [SibSp, Deck, Fare_cat, Pclass, Age_band, Init... | 9 | 0.820248 |
| **8** | [SibSp, Deck, Fare_cat, Sex, Pclass, Age_band,... | 9 | 0.819778 |
| **12** | [Fare_cat, Pclass, Initial, Embarked, Is_Marri... | 9 | 0.819305 |
| **11** | [Fare_cat, Pclass, Age_band, Initial, Embarked... | 9 | 0.819123 |
| **13** | [Fare_cat, Initial, Embarked, Is_Married, Titl... | 9 | 0.819093 |
| **15** | [Initial, Embarked, Is_Married, Title, SibSp, ... | 10 | 0.818819 |
| **10** | [Deck, Fare_cat, Pclass, Age_band, Initial, Em... | 9 | 0.818590 |
| **7** | [SibSp, Deck, Fare_cat, Title, Sex, Pclass, Ag... | 10 | 0.818501 |
| **6** | [SibSp, Deck, Fare_cat, Title, Sex, Pclass, Ag... | 10 | 0.817196 |
| **16** | [Initial, Embarked, Is_Married, Title, SibSp, ... | 11 | 0.816012 |
| **5** | [SibSp, Deck, Fare_cat, Title, Sex, Pclass, Ag... | 10 | 0.808210 |
| **4** | [SibSp, Deck, Fare_cat, Title, Sex, Pclass, Ag... | 11 | 0.807162 |
| **2** | [Initial, SibSp, Deck, Fare_cat, Title, Sex, P... | 12 | 0.807076 |
| **3** | [Initial, SibSp, Deck, Fare_cat, Title, Sex, P... | 11 | 0.806501 |
| **17** | [Initial, Embarked, Is_Married, Title, SibSp, ... | 12 | 0.806409 |
| **1** | [Initial, SibSp, Deck, Fare_cat, Title, Sex, I... | 13 | 0.803794 |
| **18** | [Initial, Embarked, Is_Married, Title, SibSp, ... | 13 | 0.801950 |
| **0** | [Initial, SibSp, Deck, Fare_cat, Title, Sex, I... | 14 | 0.799777 |
| **19** | [Initial, Embarked, Is_Married, Title, SibSp, ... | 14 | 0.798650 |
| **20** | [Initial, Embarked, Is_Married, Title, SibSp, ... | 15 | 0.793762 |

Fig. 5. Solución Titanic búsqueda secuencial hacia atrás mixta.

Podemos ver que respecto a la búsqueda secuencial hacia atrás este algoritmo proporciona soluciones con mejor rendimiento, ya que en cada iteración hace una corrección si al añadir una nueva variable que no ha sido añadida antes ni esté en el subconjunto de variables mejora el rendimiento, sin embargo si el rendimiento mejora al añadir una variable el tamaño del subconjunto aumenta.

B. *Breast Cancer*

| | variables | size | score |
|---|---|---|---|
| 14 | [mean perimeter, mean area, mean fractal dimen... | 15 | 0.966157 |
| 12 | [mean perimeter, radius error, texture error, ... | 13 | 0.964752 |
| 15 | [mean perimeter, mean area, mean fractal dimen... | 16 | 0.963340 |
| 19 | [mean texture, mean perimeter, mean area, mean... | 20 | 0.962844 |
| 7 | [compactness error, concave points error, frac... | 8 | 0.962828 |
| 23 | [mean texture, mean perimeter, mean area, mean... | 24 | 0.962341 |
| 6 | [concave points error, fractal dimension error... | 7 | 0.961874 |
| 22 | [mean texture, mean perimeter, mean area, mean... | 23 | 0.960936 |
| 11 | [mean perimeter, radius error, compactness err... | 12 | 0.960494 |
| 21 | [mean texture, mean perimeter, mean area, mean... | 22 | 0.960051 |
| 24 | [mean texture, mean perimeter, mean area, mean... | 25 | 0.960051 |
| 18 | [mean texture, mean perimeter, mean area, mean... | 19 | 0.960043 |
| 16 | [mean perimeter, mean area, mean symmetry, mea... | 17 | 0.959593 |
| 20 | [mean texture, mean perimeter, mean area, mean... | 21 | 0.959089 |
| 13 | [mean perimeter, mean area, radius error, text... | 14 | 0.959066 |
| 25 | [mean texture, mean perimeter, mean area, mean... | 26 | 0.958646 |
| 27 | [mean radius, mean texture, mean perimeter, me... | 28 | 0.958639 |
| 17 | [mean perimeter, mean area, mean compactness, ... | 18 | 0.958631 |
| 5 | [concave points error, fractal dimension error... | 6 | 0.958562 |
| 26 | [mean texture, mean perimeter, mean area, mean... | 27 | 0.958127 |
| 10 | [mean perimeter, compactness error, concave po... | 11 | 0.957685 |
| 9 | [mean perimeter, compactness error, concave po... | 10 | 0.955837 |
| 29 | [mean radius, mean texture, mean perimeter, me... | 29 | 0.954868 |
| 8 | [mean perimeter, compactness error, concave po... | 9 | 0.954349 |
| 4 | [fractal dimension error, worst texture, worst... | 5 | 0.951990 |
| 2 | [worst texture, worst area, worst smoothness] | 3 | 0.951540 |
| 28 | [mean radius, mean texture, mean perimeter, me... | 30 | 0.950174 |
| 3 | [fractal dimension error, worst texture, worst... | 4 | 0.949181 |
| 1 | [worst area, worst smoothness] | 2 | 0.942609 |
| 0 | [worst area] | 1 | 0.853361 |

Fig. 6. Solución BreastCancer búsqueda secuencial hacia atrás

En esta tabla, al igual que en las anteriores, se proporciona el rendimiento de los diferentes subconjuntos. Al analizar los datos presentados, se puede observar que existen ciertos subconjuntos que logran mejorar el rendimiento en comparación con otros. Esta información es determinante para identificar cuáles combinaciones específicas de elementos contribuyen de manera más significativa a los resultados generales.

| | variables | size | score |
|---|---|---|---|
| 34 | [worst compactness, texture error, worst smoot... | 16 | 0.969474 |
| 33 | [worst compactness, texture error, worst smoot... | 15 | 0.969422 |
| 22 | [mean texture, mean area, mean concave points,... | 11 | 0.969055 |
| 9 | [mean texture, mean area, mean smoothness, mea... | 20 | 0.968638 |
| 19 | [mean texture, mean area, mean concave points,... | 12 | 0.968423 |
| 31 | [worst compactness, texture error, worst smoot... | 13 | 0.968355 |
| 20 | [mean texture, mean area, mean concave points,... | 11 | 0.968344 |
| 36 | [worst compactness, texture error, worst smoot... | 18 | 0.968320 |
| 38 | [worst compactness, texture error, worst smoot... | 20 | 0.968224 |
| 18 | [mean texture, mean area, mean concave points,... | 12 | 0.968222 |
| 37 | [worst compactness, texture error, worst smoot... | 19 | 0.968164 |
| 12 | [mean texture, mean area, mean smoothness, mea... | 18 | 0.968059 |
| 15 | [mean texture, mean area, mean smoothness, mea... | 15 | 0.968008 |
| 26 | [mean texture, worst texture, worst area, wors... | 11 | 0.967944 |
| 13 | [mean texture, mean area, mean smoothness, mea... | 17 | 0.967913 |
| 32 | [worst compactness, texture error, worst smoot... | 14 | 0.967878 |
| 10 | [mean texture, mean area, mean smoothness, mea... | 19 | 0.967862 |
| 35 | [worst compactness, texture error, worst smoot... | 17 | 0.967779 |
| 14 | [mean texture, mean area, mean smoothness, mea... | 16 | 0.967635 |
| 11 | [mean texture, mean area, mean smoothness, mea... | 18 | 0.967635 |
| 21 | [mean texture, mean area, mean concave points,... | 11 | 0.967605 |
| 27 | [worst texture, worst area, worst compactness,... | 11 | 0.967583 |
| 30 | [worst compactness, texture error, worst smoot... | 12 | 0.967550 |
| 39 | [worst compactness, texture error, worst smoot... | 21 | 0.967547 |
| 29 | [worst compactness, texture error, worst smoot... | 11 | 0.967516 |
| 28 | [worst area, worst compactness, texture error,... | 11 | 0.967367 |
| 17 | [mean texture, mean area, mean concave points,... | 13 | 0.967285 |
| 25 | [mean texture, worst texture, worst area, wors... | 11 | 0.967193 |
| 16 | [mean texture, mean area, mean concave points,... | 14 | 0.967157 |
| 8 | [mean texture, mean area, mean smoothness, mea... | 21 | 0.966873 |
| 24 | [mean texture, mean concave points, worst text... | 11 | 0.966872 |
| 40 | [worst compactness, texture error, worst smoot... | 22 | 0.966621 |
| 23 | [mean texture, mean area, mean concave points,... | 11 | 0.966558 |
| 7 | [mean texture, mean area, mean smoothness, mea... | 22 | 0.966261 |
| 5 | [mean texture, mean area, mean smoothness, mea... | 24 | 0.965805 |
| 6 | [mean texture, mean area, mean smoothness, mea... | 23 | 0.965686 |
| 42 | [worst compactness, texture error, worst smoot... | 24 | 0.965129 |
| 41 | [worst compactness, texture error, worst smoot... | 23 | 0.965061 |
| 43 | [worst compactness, texture error, worst smoot... | 25 | 0.964887 |
| 3 | [mean texture, mean area, mean smoothness, mea... | 26 | 0.964234 |
| 4 | [mean texture, mean area, mean smoothness, mea... | 25 | 0.963915 |
| 45 | [worst compactness, texture error, worst smoot... | 27 | 0.963901 |
| 2 | [mean texture, mean area, mean smoothness, mea... | 27 | 0.963420 |
| 44 | [worst compactness, texture error, worst smoot... | 26 | 0.962783 |
| 1 | [mean texture, mean perimeter, mean area, mean... | 28 | 0.962665 |
| 46 | [worst compactness, texture error, worst smoot... | 28 | 0.962201 |
| 0 | [mean radius, mean texture, mean perimeter, me... | 29 | 0.961194 |
| 47 | [worst compactness, texture error, worst smoot... | 29 | 0.961069 |
| 48 | [worst compactness, texture error, worst smoot... | 30 | 0.958259 |

Fig. 7. Solución BreastCancer búsqueda secuencial hacia atrás mixta.

Al comparar los resultados obtenidos con los diferentes subconjuntos de variables proporcionados por los algoritmos de búsqueda, se puede ver que hay una mejora en el rendimiento en comparación con la búsqueda hacia atrás, aunque esta mejora no es significativa.

Además se realizan diferentes experimentos para ver la calidad de las soluciones proporcionadas por los algoritmos. Se dividirán los datos en entrenamiento y pruebas, para luego usar los siguientes algoritmos de entrenamiento.

- DecissionTreeClassifier
- kNN

En primer lugar se hace usando los algoritmos de entrenamiento de *DecissionTreeClassifier* y *kNN*, para ver la tasa de acierto, su matriz de confusión y la sensibilidad sobre el conjunto de pruebas, además se usa la búsqueda en rejilla para optimizar los hiperparámetros y mejorar el rendimiento de los algoritmos.

Por último, se realizará el mismo procedimiento con cada subconjunto de variables que nos proporcionan los algoritmos de búsqueda, con el fin de evaluar la calidad de las soluciones propuestas. Se puede observar la relación entre los mejores subconjuntos, determinados según el rendimiento proporcionado por los algoritmos de búsqueda, y las métricas calculadas gracias a los algoritmos de entrenamiento.

Para ello, se llevará a cabo lo siguiente:

1. *Ajuste del Modelo:*Buscar los mejores hiperparametros para cada subconjunto.
2. *Evaluación de Métricas*: Calcular las métricas de rendimiento para cada subconjunto de variables, incluyendo la tasa de acierto en el conjunto de entrenamiento, la matriz de confusión, la sensibilidad y la tasa de acierto balanceada en el conjunto de prueba.
3. *Comparación de Resultados*: Comparar las métricas obtenidas para cada subconjunto de variables, identificando aquellas que proporcionan un mejor rendimiento.

Al final, se obtendrá una visión clara de la calidad de cada subconjunto de variables, permitiendo identificar cuáles de ellos son los más efectivos según las métricas de evaluación seleccionadas. Esto proporcionará una base sólida para la toma de decisiones en cuanto a la selección de variables óptimas para el modelo.

## V. Conclusiones

Gracias a este trabajo podemos comprender mejor en que nos ayuda la selección de características [10], concretamente mediante el método de envoltura, que utilizan medidas estadísticas para evaluar la importancia de cada característica y seleccionar un subconjunto de las más relevantes.

Además de conocer 2 algoritmos de búsqueda como lo son el algoritmo de búsqueda secuencial hacia atrás y el algoritmo de búsqueda secuencial hacia atrás mixta, y ver como estos algoritmos obtienen diferentes subconjuntos de variables predictoras y aprender a evaluar estas soluciones para ver si guardan un buen compromiso entre variables seleccionadas y su tasa de acierto.

Es importante recalcar que el mejor subconjunto de variables no es el que proporciona mejor rendimiento, si no el que guarda un mejor compromiso entre el número de variables del subconjunto y rendimiento.

Para mejorar y extender este trabajo, se podrían explorar otros algoritmos de selección de características además de los ya utilizados. Algoritmos como la búsqueda secuencial hacia adelante u otros algoritmos además de usar modelos como el de Naive Bayes [11] podrían proporcionar diferentes perspectivas y resultados. La comparación entre estos métodos y los actuales podría ofrecer una visión más completa de la efectividad de cada enfoque en diferentes tipos de conjuntos de datos.

## Referencias


[1] https://www.kaggle.com/c/titanic
[2] https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data
[3] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[4] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[5] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
[6] https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada
[7] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html
[8] https://es.wikipedia.org/wiki/Aprendizaje_basado_en_%C3%A1rboles_de_decisi%C3%B3n
[9] https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos
[10] https://en.wikipedia.org/wiki/Feature_selection#Main_principle
[11] https://es.wikipedia.org/wiki/Clasificador_bayesiano_ingenuo
[12] Práctica 1 - Curso Inteligencia Artificial - Universidad de Sevilla