Algoritmo de búsqueda para selección de características en modelos predictivos

Inteligencia Artificial (IS) 2019/20 – Propuesta de trabajo

Juan Galán Páez

1. Introducción y objetivo

1.1. Selección de características y aprendizaje automático

Una de las ramas más importante del aprendizaje automático (*machine learning*) es el aprendizaje supervisado que proporciona algoritmos para el entrenamiento de modelos predictivos. El aprendizaje supervisado parte de un conjunto de entrenamiento en el que distinguimos una serie de variables predictoras y una variable respuesta. Esta variable respuesta es el objetivo del aprendizaje. El algoritmo de aprendizaje será capaz de encontrar patrones en las variables predictoras que le permitan, dada una nueva instancia de datos para la que estos valores son conocidos, estimar (predecir) el valor de la variable respuesta más adecuado.

El presente trabajo se centra en la selección características[1], es decir, la obtención de un subconjunto de variables predictoras que aporten la mayor cantidad de información posible sobre la variable respuesta. En otras palabras, el objetivo es quedarnos con las variables que más información aportarán al algoritmo de entrenamiento durante el proceso de aprendizaje y descartar aquellas que aporten menos información. ¿Por qué queremos seleccionar variables en vez de usarlas todas? ¿Por qué queremos descartar variables que aportan poca información sobre la variable respuesta, si aportan algo? Existen varias razones por las que la selección de características es importante aunque todas ellas están relacionadas con la complejidad del modelo predictivo resultante. Uno de los factores que más afecta (aunque no el único) a la complejidad de del modelo predictivo obtenido es el numero de variables predictoras con que ha sido entrenado.

- **Recursos computacionales:** El tiempo de entrenamiento y recursos necesitados (procesador, memoria RAM, etc.) serán menores cuanto menos variables tenga nuestro conjunto de datos.
- **Interpretabilidad:** Cuanto más sencillo sea el modelo predictivo obtenido, más facil será interpretar las decisiones (predicciones) que este produce.
- **Sobreajuste y capacidad predictiva:** Cuanto más variables (y menos informativas) se usen en el proceso de aprendizaje, tendremos más riesgo de que el modelo aprenda patrones falsos (ruido), es decir, patrones que existen en el conjunto de entrenamiento pero no en la realidad. Un conjunto de variables reducido pero muy informativas permitirá al algoritmo encontrar pocos patrones pero muy valiosos a la hora de predecir la variable respuesta.

La selección de características es uno de los problemas bajo estudio en la actualidad (y desde hace décadas) más importante dentro del aprendizaje automático. En la actualidad existen numerosas soluciones a este problema, sin embargo, no existe ninguna solución universal. Cada método tiene sus ventajas e inconvenientes y funciona bien sobre ciertos tipos de problemas y algoritmos y mal sobre otros.

1 https://en.wikipedia.org/wiki/Feature_selection

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

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

Nótese que aunque en este trabajo usaremos árboles de decisión, no se usará el método integrado que este proporciona y en su lugar, usaremos los árboles de decisión dentro del método de envoltura.

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

En muchos casos combinaremos los métodos de filtro o los integrados con los de envoltura. En otras palabras, usaremos métodos de filtro o integrados para obtener subconjuntos de características candidatos y los evaluaremos y seleccionaremos mediante el método de envoltura.

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

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

mejores conjuntos obtenidos son los siguientes:

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

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

Como se ha comentado, el presente trabajo se centra en la implementación de un método de selección de características de tipo envoltura. Los métodos de envoltura necesitan un algoritmo de aprendizaje para evaluar cada uno de los subconjuntos candidatos. Para el desarrollo de este trabajo nos centraremos en los árboles de decisión, aunque luego como opción para subir nota, se propondrá la posibilidad de incluir otros algoritmos.

1.2. Árboles de decisión

Los algoritmos para la obtención de árboles de decisión pertenecen al conjunto de técnicas de aprendizaje supervisado. Este tipo de algoritmos permiten obtener un árbol de decisión consistente con un conjunto de ejemplos de entrenamiento. A medida que se van añadiendo nodos al árbol, los ejemplos se van repartiendo entre las ramas (en función del valor del atributo que ha sido elegido como nodo) hasta llegar a las hojas, las cuales tendrán asignada una de las posibles categorías de la variable objetivo (también llamada variable respuesta).

Para la realización de este trabajo usaremos un algoritmo que construye árboles binarios y cuya implementación nos proporciona la librería Scikit-learn de Python (ver enlace[3]). Este algoritmo pertenece a la familia CART (Classification and Regression Trees). La principal diferencia con el algoritmo ID3 visto en clase, es que ID3 está pensado para trabajar con variables categóricas y CART permite trabajar tanto con variables continuas como categóricas.

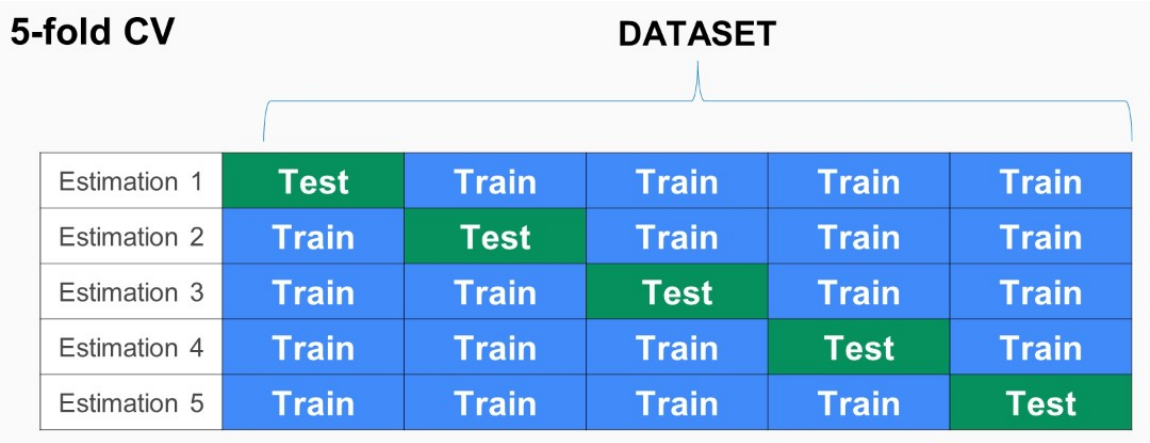
1.3. Validación cruzada

Como se ha comentado, los métodos de selección de características de tipo envoltura, necesitan evaluar cada solución candidata (subconjunto de variables). Para esto, entrenan un algoritmo de aprendizaje (árboles de decisión en nuestro caso) y evalúan su capacidad predictiva. La mayor parte de los algoritmos y procedimientos usados en aprendizaje automático tienen un cierto componente de aleatoriedad. Esto hace que el entrenamiento y posterior evaluación de un algoritmo usando el mismo conjunto de variables, puede resultar e tasas de acierto ligeramente diferentes. De la misma forma, el particionado aleatorio en conjunto de prueba y evaluación también provoca variabilidad en las métricas de evaluación de la capacidad predictiva. La evaluación mediante validación cruzada viene a solucionar este problema.

Veamos una definición de validación cruzada (extraído de Wikipedia)[4]: *En la validación cruzada de*

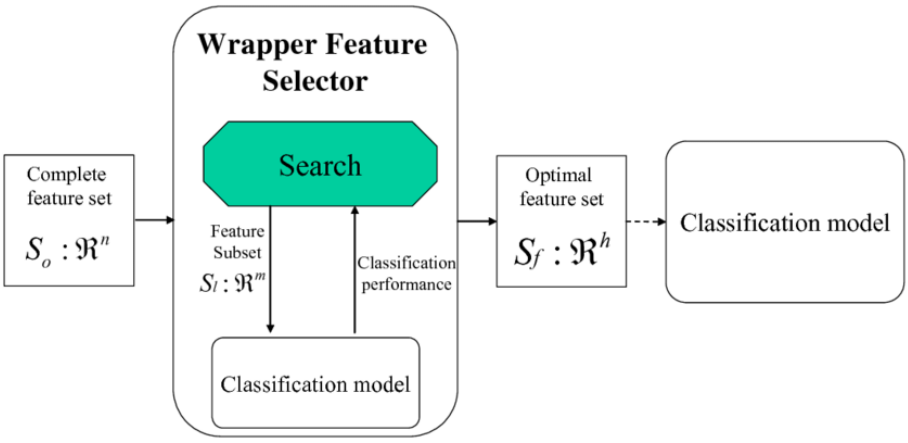
3 https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
4 https://es.wikipedia.org/wiki/Validación_cruzada

K iteraciones o K-fold cross-validation los datos se dividen en K subconjuntos. Uno de los subconjuntos se utiliza como datos de evaluación (datos a predecir y sobre los que evaluar el rendimiento) y el resto (K-1) como datos de entrenamiento. El proceso de validación cruzada es repetido durante k iteraciones, con cada uno de los posibles subconjuntos de datos de prueba. Finalmente se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. Este método es muy preciso puesto que evaluamos a partir de K combinaciones de datos de entrenamiento y de prueba. En la práctica, la elección del número de iteraciones depende de la medida del conjunto de datos. Lo más común es utilizar la validación cruzada de 10 iteraciones (10-fold cross-validation).



En la imagen podemos ver el esquema de validación cruzada para 5 folds. Como se puede ver, se realizan 5 experimentos que entre todos cubren el conjunto de datos completo. Es decir, es como evaluásemos, sin filtrar información, nuestro algoritmo sobre todo el conjunto de datos completo. En cada uno de los 5 experimentos obtenemos un valor para la métrica de evaluación de la capacidad predictiva sobre el conjunto de evaluación, es decir, el segmento de color verde en cada caso. Finalmente promediamos las 5 estimaciones.

1.4. Algoritmos de búsqueda para selección de características



Como se ha comentado, los métodos de envoltura, son procesos iterativos en los que en cada iteración se selecciona un subconjunto de variables, se entrena un modelo predictivo, y se evalúa el

rendimiento de este. Este proceso se repite para diferentes subconjuntos de características. Al terminar nos quedamos con el subconjunto de características que mejor compromiso entre rendimiento y número de características proporcione.

La búsqueda exhaustiva no es viable:
El enfoque más sencillo e intuitivo, sería generar y evaluar todos los posibles subconjuntos de características.
Por ejemplo, sea un conjunto de 3 variables, el conjunto de todos los subconjuntos posibles sería el siguiente:
('A', 'B', 'C') → [('A',), ('B',), ('C',), ('A', 'B'), ('A', 'C'), ('B', 'C'), ('A', 'B', 'C')]
En total son 7 subconjuntos posibles.

So el conjunto tiene 5 variables:
('A', 'B', 'C', 'D', 'E') → [('A',), ('B',), ('C',), ('D',), ('E',), ('A', 'B'), ('A', 'C'), ('A', 'D'), ('A', 'E'), ('B', 'C'), ('B', 'D'), ('B', 'E'), ('C', 'D'), ('C', 'E'), ('D', 'E'), ('A', 'B', 'C'), ('A', 'B', 'D'), ('A', 'B', 'E'), ('A', 'C', 'D'), ('A', 'C', 'E'), ('A', 'D', 'E'), ('B', 'C', 'D'), ('B', 'C', 'E'), ('B', 'D', 'E'), ('C', 'D', 'E'), ('A', 'B', 'C', 'D'), ('A', 'B', 'C', 'E'), ('A', 'B', 'D', 'E'), ('A', 'C', 'D', 'E'), ('B', 'C', 'D', 'E'), ('A', 'B', 'C', 'D', 'E')]
En total son 31 subconjuntos posibles.

La regla general es que dado un conjunto de N variables, existen $2^{\wedge N}$-1 posibles subconjuntos. Si por ejemplo, tenemos 50 variables, el numero total de subconjuntos existente será de $2^{\wedge 50}$-1, es decir, 1.125.899.906.842.623 subconjuntos posibles.

Esta explosión combinatoria hace inviable el uso de una búsqueda exhaustiva, por lo que es necesario diseñar estrategias de búsqueda que obtengan el mejor resultado posible probando una pequeña parte de los subconjuntos posibles.

Por último, es importante aclarar la razón por la que debemos evaluar subconjuntos de variables. Es decir, ¿por qué no evaluar la capacidad predictiva de cada variable de forma inpendendiente y luego seleccionar aquellas con mayor capacidad predictiva? La respuesta es que existen interacciones no lineales entre variables o dicho de otro modo, dos variables que por separado tienen baja capacidad predictiva, pueden ser buenos predictores si son usados de forma conjunta. De la misma forma dos variables altamente predictivas pueden ser redundantes (ambas aportan la misma información sobre la variable respuesta), y por tanto, podemos prescindid de una de ellas sin perder capacidad predictiva.

Estrategias de búsqueda secuenciales:
Aunque existen enfoques más complejos, como los algoritmos genéricos (y muchas otras técnicas metaheurísticas), dado que estos pertenecen a un tema que no ha sido cubierto antes de proponer este trabajo, nos centraremos en estrategias de búsqueda sencillas, como son las secuenciales. Estas estrategias se descomponen en tres grandes grupos:

- Hacia delante: Partimos de un conjunto vacio y en cada iteración añadimos una nueva variable (la que consideramos mejor a priori), hasta alcanzar el número deseado.
- Hacia atrás: Partimos del conjunto de todas las variables y en cada iteración eliminamos una variable (la que consideramos peor a priori) hasta alcanzar el número deseado.
- Mixtas: Combinan ambas direcciones. Existe una dirección principal y la otra se usa como secundaria para ‘corregir’ soluciones no optimales. Por ejemplo, si tomamos la búsqueda

hacia delante como principal, en cada paso, existirá la posibilidad de eliminar una variable (es decir, dar un paso atrás) si es que eso produce una mejora, en caso contrario, no eliminamos nada.

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

Sequential forward selection (SFS):

El algoritmo de búsqueda secuencial hacia delante empieza con un conjunto vacío de variables. En cada iteración se selecciona, de entre las variables pendientes, la mejor variable a añadir a este conjunto. Para seleccionar la mejor variable a añadir, en cada iteración se evalúan los posibles candidatos. El algoritmo termina cuando se ha alcanzado el número de variable deseado o cuando se hayan añadido todas las variables (si no se especificó ningún numero máximo de variables). La salida será una tabla que contenga la mejor solución encontrada en cada iteración.

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

Sequential floating forward selection (SFFS):

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

El algoritmo de búsqueda secuencial variable hacia delante es un algoritmo secuencial mixto donde la estrategia principal es hacia delante y la secundaria hacia atrás Es similar al algoritmo SFS visto anteriormente, sin embargo en cada iteración se añade la posibilidad de eliminar una variable previamente añadida, solo si el subconjunto obtenido tras eliminar la variable tiene mejor capacidad predictiva. El algoritmo termina cuando la solución se estabiliza, es decir, el subconjunto de variables actual no es mejorable ni añadiendo ni quitando variables.

La salida será una tabla que contenga la mejor solución encontrada en cada iteración. En estos algoritmos siempre devolvemos una tabla de mejores soluciones y no solo la mejor ya que, como se comentó anteriormente, a veces preferimos elegir una solución que tiene menos rendimiento pero también menos variables.

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

1.5. Objetivo principal

El **objetivo principal** de esta propuesta es desarrollar nuestro propio algoritmo de selección de características. Según lo visto anteriormente, el objetivo es implementar, en primer lugar, el algoritmo Sequential forward selection (SFS) y en segundo lugar el algoritmo Sequential floating forward selection (SFFS). Para ambos algoritmos, será necesario implementar el método de evaluación robusta. Por último, se proporcionan dos conjuntos de datos sobre los que se deben aplicar los algoritmos desarrollados. Este objetivo se descompone en los siguientes objetivos específicos

- Todo el desarrollo realizado debe ser generalizable, es decir, no debe estar vinculado a un conjunto de datos o problema concreto. Para esto, el código debe estar parametrizado según sea necesario.
- Implementar un método de evaluación robusta, que a partir de un conjunto de datos de entrada y una selección de variables del mismo, (y los otros parámetros que fueron definidos en la sección anterior) proporcione una estimación robusta de la capacidad predictiva que tendría un árbol de decisión entrenado sobre dicho subconjunto de datos. La medida de evaluación a de la capacidad predictiva a usar será la tasa de aciertos balanceada (balanced accuracy score).
- Implementar los algoritmos Sequential forward selection (SFS) y Sequential forward floating selection (SFFS) que a partir de un conjunto de datos de entrada, devuelva una tabla con los subconjuntos de variables mas prometedores, es decir, aquellos con mayor capacidad predictiva.
- Se proporcionan dos conjuntos de datos en el que se ha comprobado que es posible obtener subconjuntos de variables que mejoren el rendimiento proporcionada por el conjunto de variables completo. Para esto, es importante, realizar un primer experimento en el que se evalúe la capacidad predictiva del conjunto de variables completo. Esta tasa de aciertos servirá de referencia para evaluar la calidad de las soluciones que nuestro algoritmo de búsqueda proporciona.
- Documentar el trabajo en un fichero con formato de artículo científico, explicando con precisión las decisiones de diseño en la implementación de los algoritmos. Se debe mostrar que se ha entendido el problema de la selección de características en general y el funcionamiento de los algoritmos desarrollados en particular. De la misma forma se interpretarán los resultados obtenidos en los diferentes experimentos. Por último, de forma razonada se elegirá el mejor subconjunto de variables para cada conjunto de datos.
- Realizar una presentación de los resultados obtenidos en la defensa del trabajo.

Para que el trabajo pueda ser evaluado, se deben satisfacer TODOS los objetivos específicos al completo: el trabajo debe ser original, estar correctamente implementado y funcionar perfectamente, los experimentos se deben haber llevado a cabo y analizados razonadamente, el documento debe ser completo y contener entre 6 y 12 páginas, y se debe realizar la defensa con una presentación de los resultados obtenidos.

2. Descripción del trabajo

A continuación se proporcionan una guía para el correcto desarrollo del trabajo.

Se pueden usar funciones ya implementadas para el cálculo de métricas, entrenamiento de arboles de decisión o evaluación mediante validación cruzada. No se permite el uso de funciones de alto nivel que realicen el procedimiento de búsqueda.

Antes de comentar a implementar el trabajo, se recomienda que el alumno se familiarice con:

- La librería para manipulación de conjuntos de datos Pandas (usando los llamados DataFrames).
- El framework de aprendizaje automático Scikit-learn.
- Los conjuntos de datos proporcionados.
- La problemática de la selección de características en general.
- Los algoritmos de busqueda a implementar. Se recomienda entenderlos bien antes de empezar a escribir código.
- La práctica de aprendizaje automático.

2.1. Evaluación de soluciones:

Una de las piezas fundamentales del procedimiento de búsqueda es la función de evaluación de las soluciones candidatas que permitirá elegir la solución más prometedora en cada paso. En esta sección detallamos como sería un procedimiento de evaluación robusto. Para obtener una evaluación robusta, debemos realizar varios experimentos y promediar la tasa de aciertos obtenida. Es decir, aunque la propia validacion cruzada ya nos proporciona cierta robustez en la estimación, debido a la aleatoriedad existente en los procesos a usa, además vamos a promediar varios experimentos de validación cruzada.

Para esto, el framework de python Scikit-learn nos proporciona una función[5] para realizar evaluaciones mediante validación cruzada, por lo que no será necesario implementarla. Esta función nos devolverá el valor para la métrica seleccionada (que indica la capacidad predictiva) de nuestro algoritmo. **Importante:** Existen decenas de métricas[6] para evaluar la capacidad predictiva de un algoritmo, donde la más sencilla es la tasa de aciertos (predicciones_acertadas/total_predicciones) o accuracy en inglés. Para este trabajo se desea utilizar la tasa de aciertos balanceada[7], en inglés balanced_accuracy_score. Para usarla, bastará con especificar el valor ‘balanced_accuracy’ en el parámetro ‘scoring’ de la función de validación cruzada (cross_val_score).

Se propone el siguiente método de evaluación robusta de cada conjunto de variables quedaría como sigue:

Entrada:

- Datos: Conjunto de datos completo. Contiene todas las filas y todas las columnas (variables predictoras y variable respuesta)
- Variables: Subconjunto de variables a evaluar. Es un subconjunto de las columnas contenidas en Datos. Es importante recordar, que la variable respuesta nunca podrá ser

5 https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
6 https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter
7 https://scikit-learn.org/stable/modules/model_evaluation.html#balanced-accuracy-score

incluida en el subconjunto de variables a evaluar.

- N_Exp: número de repeticiones del experimento por validación cruzada. Valores recomendados entre 10 y 20.
- CV: Número de folds a considerar en la validación cruzada (10 por defecto).

Salida:

- Tasa de aciertos balanceada promedio obtenida.

Ejecución:

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

Consideraciones:

- *Antes de comentar con la implementación, se recomienda que el alumno haga pruebas de entrenamiento y predicción con árboles de decisión de forma individual. De la misma forma debe familirizarse con el método de validación cruzada . También se recomienda entender la métrica usada.*
- *Mientras desarrollamos nuestros algoritmos, se recomienda usar N_Exp=1 y CV=3 para que los experimentos sean más rápidos. Una vez que todo esté funcionando correctamente pondremos los valores adecuados.*
- *La función cross_val_score realizará un experimento de validación cruzada y proporcionará una estimación de la capacidad predictiva o tasa de aciertos de un algoritmo entrenado sobre nuestros datos. Esto implica que en la función cross_val_score debemos especificar el algoritmo de aprendizaje que vamos a usar. Tal y como se comentó al principio, el algoritmo a usar será la implementación de árbol de decisión que nos proporciona Scikit-learn.*

2.2. Sequential forward selection (SFS):

A continuación se proporciona una descripción más detallada del algoritmo de búsqueda secuencial hacia delante:

Entrada:

- Conjunto de datos con N variables predictoras y una variable respuesta (sobre la que evaluar la capacidad predictiva).
- Número de variables D máximo a probar. Si D no se proporciona, D=N.

Salida:

- Tabla con cada una de las combinaciones obtenidas en cada iteración, su tamaño y su rendimiento. El objetivo de proporcionar esta tabla es que es posible que la solución más conveniente no sea una de tamaño K sino una de menor tamaño.

Inicialización:

- *SolucionActual*: Almacena el mejor conjunto de variables obtenido en cada iteración. Inicialmente está vacío.
- K=0. K es el contador de iteraciones o de variables seleccionadas en cada iteración.

Ejecución:

Mientras que K<D:

1. Seleccionar la mejor variable para añadir a *SolucionActual*. Por cada variable V del conjunto original de variables que no se encuentre en *SolucionActual*:

1.1. *SolucionTemporal* = *SolucionActual* + V
1.2. Evaluar *SolucionTemporal* y guardar su rendimiento.

2. Seleccionar la mejor *SoluciónTemporal* (la que proporcione mayor rendimiento) y hacer *SolucionActual* = *MejorSolucionTemporal* y K = K+1

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

Nota: El punto 1.2 se refiere el procedimiento de evaluación robusta anteriormente descrito.

2.3. Sequential floating forward selection (SFFS):

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

Entrada:

- Conjunto de datos con N variables predictoras y una variable respuesta (sobre la que evaluar la capacidad predictiva).

Salida:

- Tabla con cada una de las combinaciones obtenidas en cada iteración, su tamaño y su rendimiento. El objetivo de proporcionar esta tabla es que es posible que la solución más conveniente no sea la mejor solución seleccionada.

Inicialización:

- *SolucionActual*: Almacena el mejor conjunto de variables obtenido en cada iteración. Inicialmente está vacío.
- *Añadidos*: Almacena las variables que ya han sido añadidas.
- *Eliminados*: Almacena las variables que han sido elimnadas.
- K=0. K es el contador de iteraciones o de variables seleccionadas en cada iteración.

Ejecución:

Mientras que no se cumpla la CondicionDeParada:

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

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

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

4. Seleccionar la **peor** variable para eliminar de *SolucionActual*. Por cada variable V de *SolucionActual* que no se encuentre en *Eliminados*:
   4.1. *SolucionTemporal* = *SolucionActual* – V
   4.2. Evaluar *SolucionTemporal* y guardar su rendimiento.
5. Seleccionar la mejor *SoluciónTemporal* (la que proporcione mayor rendimiento). Solo si el rendimiento de la mejor Solución temporal es superior al rendimiento de la mejor solucion

obtenida en el punto 2, entonces: *SoluciónActual = MejorSolucionTemporal*. En este caso, actualizar *Eliminados* añadiendo la variable eliminada.

6. Evaluar condición de parada (se detalla a continuación).

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

Consideraciones:

- *En este algoritmo no se proporciona número máximo de variables deseado ya que será el algoritmo el que determine el numero óptimo. Aunque se supone que la mejor solución obtenida tras cumplirse el criterio de parada es optimal, vamos a devolver la tabla completa para comparar resultados con el primer algoritmo.*
- *<u>Condición de parada:</u> La condición de para solo puede ser cierta una vez que todas las variables han sido procesadas por el proceso de adición, es decir, Añadidos contiene todas las variables originales. Llegados a este punto, usaremos un contador para contar las iteraciones que transcurren sin que se eliminen variables (recordemos que ya no se podrán añadir más variables). Si en alguna iteración una nueva variable es eliminada, entonces reseteamos el contador a cero. Si el contador alcanza un cierto valor umbral (por ejemplo 10 iteraciones) se considera que se cumple la condición de parada y se detiene el algoritmo. Este criterio de parada más complejo (en vez de parar cuando ya todas las variables hayan sido añadidas) nos permitirá obtener soluciones más robustas, ya que el objetivo del umbral de 10 iteraciones (o el número que se elija) es conseguir una solución estable. O dicho de otra forma, este criterio de parada nos garantiza que se han eliminado todas las variables que no son imprescindibles. La razón de este periodo de 10 iteraciones es la misma por la que usamos un procedimiento de evaluación robusta, es decir, la aleatoriedad existente en cada uno de los experimentos realizados.*
- *El objetivo final de este trabajo es la implementación de este algoritmo (SFFS). Se solicita también la implementación del algoritmo SFS por dos motivos. 1) Al ser SFS similar a SFFS pero más sencillo, nos proporciona un acercamiento gradual al problema, de forma que una vez que SFS ha sido implementado, nos será más fácil abordar la implementación del algoritmo SFFS. 2) En caso de no conseguir resultados satisfactorios en la implementación del algoritmo SFFS, dispondremos del algoritmo SFS para realizar experimentos y presentar resultados.*

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

La imagen muestra un ejemplo de salida esperada para cualquiera de los dos algoritmos de búsqueda: Cada fila se corresponde con cada una de las MejorSolucionTemporal generadas por el algoritmo. La columna solution contiene una lista con los nombres de las variables que contiene esa solución, score contiene una medida de la capacidad predictiva del algoritmo entrenado con esa solución y size contiene el numero de variables de la solución.
Importante: La salida no es necesario que sea una tabla ni que sea igual a la de la imagen. Se puede usar la estructura de datos que se considere más oportuna (listas de listas, diccionarios, etc.). Lo importante es que la salida contenga los elementos solicitados y el alumno es libre de presentarlos por pantalla como considere apropiado. Idealmente los resultados se presentarán en orden descendente, es decir, las mejores soluciones al principio y las peores al final.

Las clases o funciones desarrolladas deben proporcionarse en un módulo python (.py) de forma que puedan importarse y ser usadas en un notebook.

2.4. Conjuntos de datos

Se proporcionan dos conjuntos de datos en el que se ha comprobado que es posible obtener subconjuntos de variables que mejoren el rendimiento proporcionada por el conjunto de variables completo:

- Conjunto de datos sobre el Titanic:[8] Contiene información de pasajeros que viajaban el Titanic. El objetivo es predecir, a partir de las características de los pasajeros, quién sobrevive y quién no. El nombre de la columna que contiene la variable respuesta es s*urvived*.
- Conjunto de datos sobre cancer de mama:[9] Contiene información sobre carácterísticas físicas de diferentes masas tumorales. El objetivo es predecir, a partir de estas caracteristicas, si la masa representa un cancer benigno o maligno. El nombre de la columna que contiene la variable respuesta es *diagnosis*.

Importante: los ficheros que se proporcionan han sido modificados, por lo que es necesario trabajar con estos y no con los que se pueden obtener en internet. Los conjuntos de datos han sido tratados para que el alumno no tenga que realizar ningún tipo de preprocesado o limpieza en los datos. Estos están listos para ser usados con algoritmos de aprendizaje de Scikit-learn.

2.5. Presentación del código

El trabajo debe presentarse en forma de notebooks de jupyter que pueden ser desarrollados tanto con jupyter-notebook como jupyter-lab, ambos disponibles en el paquete Anaconda. Se proporcionarán las implementaciones solicitadas así como su aplicación sobre los conjuntos de datos proporcionados. El código debe estar debidamente documentado, las decisiones tomadas debidamente justificadas y los resultados obtenidos deben ser interpretados.

2.6. Documentación

8 https://www.kaggle.com/c/titanic/data
9 https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

En el fichero `plantilla-trabajo.doc` se muestra una sugerencia de estructura y formato de estilo artículo científico correspondiente a la documentación del trabajo. Este formato es el del *IEEE conference proceedings*, cuyo sitio web *guía para autores* [6] ofrece información más detallada y plantillas para Word y Latex.

El artículo deberá tener una extensión **entre 6 y 12 páginas**, y la estructura general del documento debe ser como sigue: en primer lugar realizar una **introducción** al trabajo explicando el objetivo fundamental, incluyendo un breve repaso de antecedentes en relación con la temática del trabajo. A continuación, describir la **estructura** del trabajo, las **decisiones de diseño** que se hayan tomado a lo largo de la elaboración del mismo, y la **metodología** seguida al implementarlo (nunca poner código, pero sí pseudocódigo), y seguidamente detallar los **experimentos** llevados a cabo, **analizando los resultados** obtenidos en cada uno de ellos. Por último, el documento debe incluir una sección de **conclusiones**, y una **bibliografía** donde aparezcan no sólo las referencias citadas en la sección de introducción, sino cualquier documento consultado durante la realización del trabajo (incluidas las referencias web a páginas o repositorios).

2.7. Mejoras

Aunque no sea obligatorio, sí se tendrán en cuenta en la calificación los siguientes objetivos adicionales:

- La función de evaluación permitirá usar otros algoritmos de aprendizaje automático en vez del arbol de decisión. Para esto, se pasaría el estimador deseado como argumento de los algoritmos de búsqueda.
- La función de evaluación permitirá modificar la métrica de evaluación a usar (actualmente ‘balanced_accuracy_score’.
- Explorar otros conjuntos de datos y evaluar como se comporta el algoritmo desarrollado. Nótese que los conjuntos de datos proporcionados se corresponden con problemas de clasificación binaria, sin embargo, la metodología propuesta debería funcionar sin cambios con conjuntos de datos con respuesta multi-clase. Sin embargo, no funcionará con conjuntos de datos con respuesta continua.
- Nuestro algoritmo entrenará múltiples árboles al mismo tiempo. Estas tareas son totalmente independientes y se beneficiarían de una implementación en paralelo.
- Representar gráficamente (usando por ejemplo la librería matplotlib) la evolución de la capacidad predictiva con respecto al número de variables consideradas, para cada algoritmo de búsqueda y cada conjunto de datos.
- Paralelizar los algoritmos de búsqueda. En concreto, los puntos 1.2 y 4.2 deberían ser fácilmente paralelizables.

2.8. Presentación y defensa

El día de la defensa se deberá realizar una pequeña presentación (PDF, PowerPoint o similar) de 10 minutos en la que participarán activamente todos los miembros del grupo que ha desarrollado el trabajo. Esta presentación seguirá a grandes rasgos la misma estructura que el documento, pero se deberá hacer especial mención a los resultados obtenidos y al análisis crítico de los mismos. Se podrá usar un portátil (personal del alumno), diapositivas y/o pizarra. En los siguientes 10 minutos de la defensa, el profesor procederá a realizar preguntas sobre el trabajo, que podrán ser tanto del documento como del código fuente. Adicionalmente, el profesor podrá proporcionar un nuevo conjunto de datos con el que probar el algoritmo implementado.

3. Criterios de evaluación

Para que el trabajo pueda ser evaluado, se deberá satisfacer los objetivos concretos descritos en el apartado 1 (todos y cada uno de ellos, si no, el trabajo obtendrá una nota de suspenso). Uno de los alumnos del equipo deberá subir a través del formulario disponible en la página de la asignatura un fichero comprimido .zip, que contenga:

- **Una carpeta con el código fuente**. **S**e proporcionarán instrucciones sobre como reproducir la experimentación realizada. El código fuente se entregará en forma de notebook de Jupyter.
- **El documento – artículo en formato PDF**. Deberá tener una extensión mínima de 6 páginas, y máxima de 12. Deberá incluir toda la bibliografía consultada (libros, artículos, technical reports, páginas web, códigos fuente, diapositivas, etc.) en el apartado de referencias, y mencionarlas a lo largo del documento.

Para la evaluación se tendrá en cuenta el siguiente criterio de valoración, considerando una nota máxima de 3 en total para el trabajo:

- **El código fuente (1.5 punto) y resultados obtenidos**: se valorará la claridad y buen estilo de programación, corrección, eficiencia y usabilidad de la implementación, y calidad de los comentarios. La claridad del fichero README.txt también se valorará. En ningún caso se evaluará un trabajo con código copiado directamente de Internet o de otros compañeros.
  En este trabajo está permitido utilizar en parte librerías de Python existentes, aunque en este apartado se valorará exclusivamente el código original desarrollado por los alumnos.
  Con respecto a la experimentación, se valorará la calidad y completitud de los experimentos realizados. Además se tendrá en cuenta el rendimiento del algoritmo y los resultados obtenidos. Por último, no se tendrán en cuenta aquellos resultados experimentales que no sean reproducibles.
- **El documento – artículo científico (0.75 punto)**:
  - Se valorará el uso adecuado del lenguaje y el estilo general del documento (por ejemplo, el uso de la plantilla sugerida).
  - Se valorará en general la claridad de las explicaciones, el razonamiento de las decisiones, y especialmente el análisis y presentación de resultados en las secciones de experimentación y conclusiones.
  - Igualmente, no se evaluará el trabajo si se detecta cualquier copia del contenido del documento. La sección de referencias deberá incluir menciones a todas las pertinentes fuentes consultadas.
- **La presentación y defensa (0,75 puntos)**: se valorará la claridad de la presentación y la buena explicación de los contenidos del trabajo, así como, especialmente, las respuestas a las preguntas realizadas por el profesor.
- **Mejoras**: Se valorarán hasta con 0.5 puntos extra sin superar el máximo de 3 puntos totales del trabajo.

IMPORTANTE: Cualquier **plagio**, **compartición de código** o uso de material que no sea original y del que no se cite convenientemente la fuente, significará automáticamente la **calificación de cero** en la asignatura para **todos los alumnos involucrados**. Por tanto, a estos alumnos **no se les conserva**, ni para la actual ni para futuras convocatorias, **ninguna nota** que hubiesen obtenido hasta el momento. Todo ello sin perjuicio de las correspondientes **medidas disciplinarias** que se pudieran

tomar.