Selección de características

Minería de datos

Master Universitario en Tecnologías de Análisis de Datos Masivos
Escola Técnica Superior de Enxeñaría (ETSE)
Universidade de Santiago de Compostela

Contenidos de la presentación

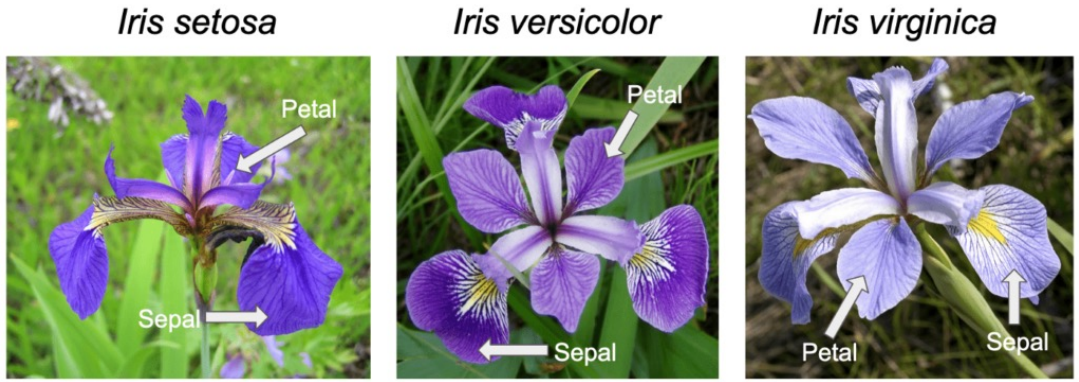
- Introducción
  - Ejemplo usando la base de datos IRIS
  - La maldición de la dimensionalidad
- Selección de características
- Evaluación de un subconjunto de atributos
  - Medidas de relevancia
  - Clasificadores de características (Rankers)
  - Métodos basados en envolturas (Wrappers)
- Conclusiones

Introducción

- Los datos disponibles para el análisis son cada vez más abundantes y complejos.
- Sin embargo, la complejidad de la mayoría de los algoritmos de aprendizaje depende de la dimensión del conjunto de entrada (número de características) y del número de instancias.
- El objetivo de la reducción de la dimensionalidad es por lo tanto:
  - Obtener una representación reducida del conjunto de datos
  - Preservar la información relevante contenida en los datos originales
- Si se consigue este objetivo, los beneficios son directos:
  - Modelos más reducidos, claros e interpretables
  - Reducción del sobreajuste
  - Identificación de atributos de interés

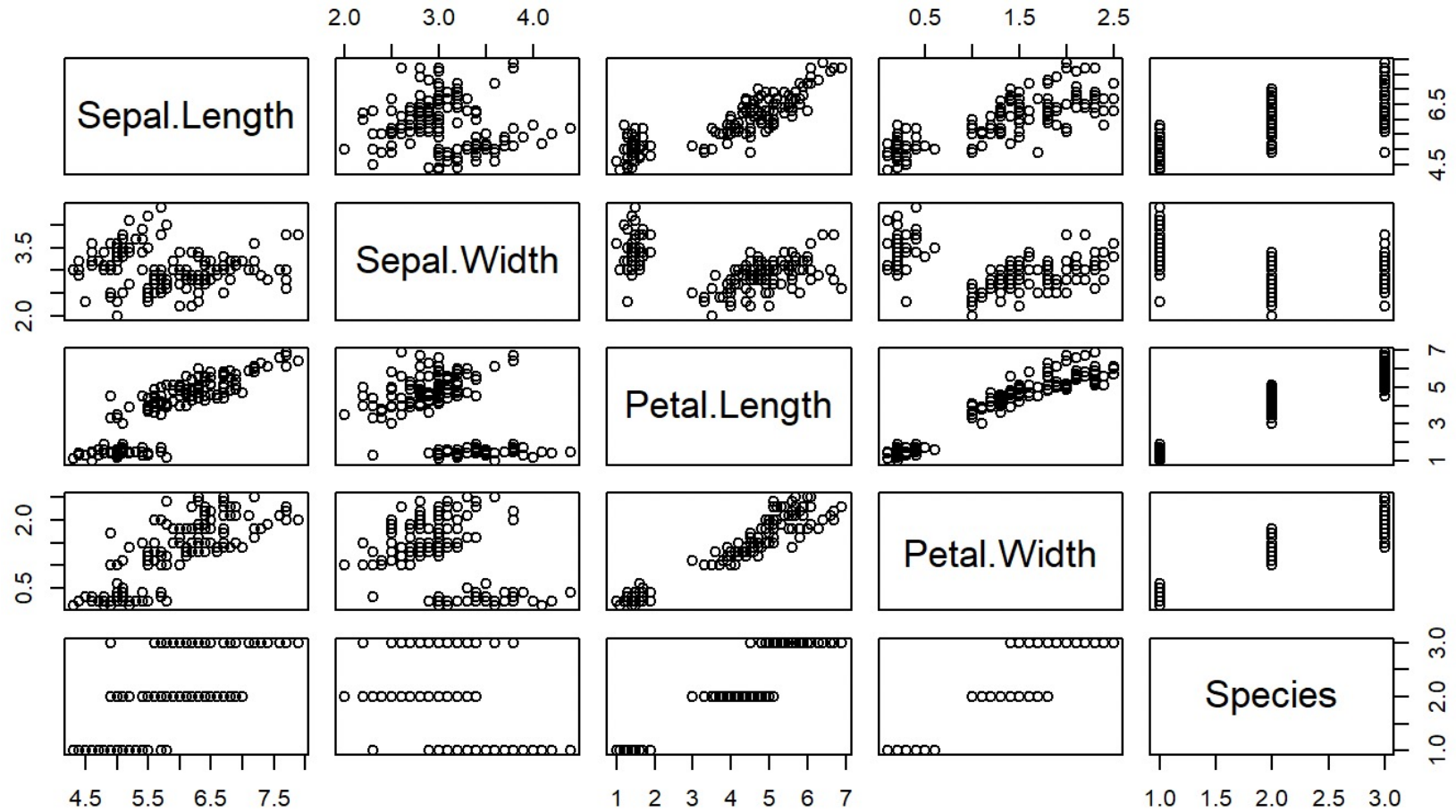
Ejemplo: veamos el efecto de las variables en una base de datos "pequeña" como IRIS.

Objetivo: clasificar las flores en base a sus características



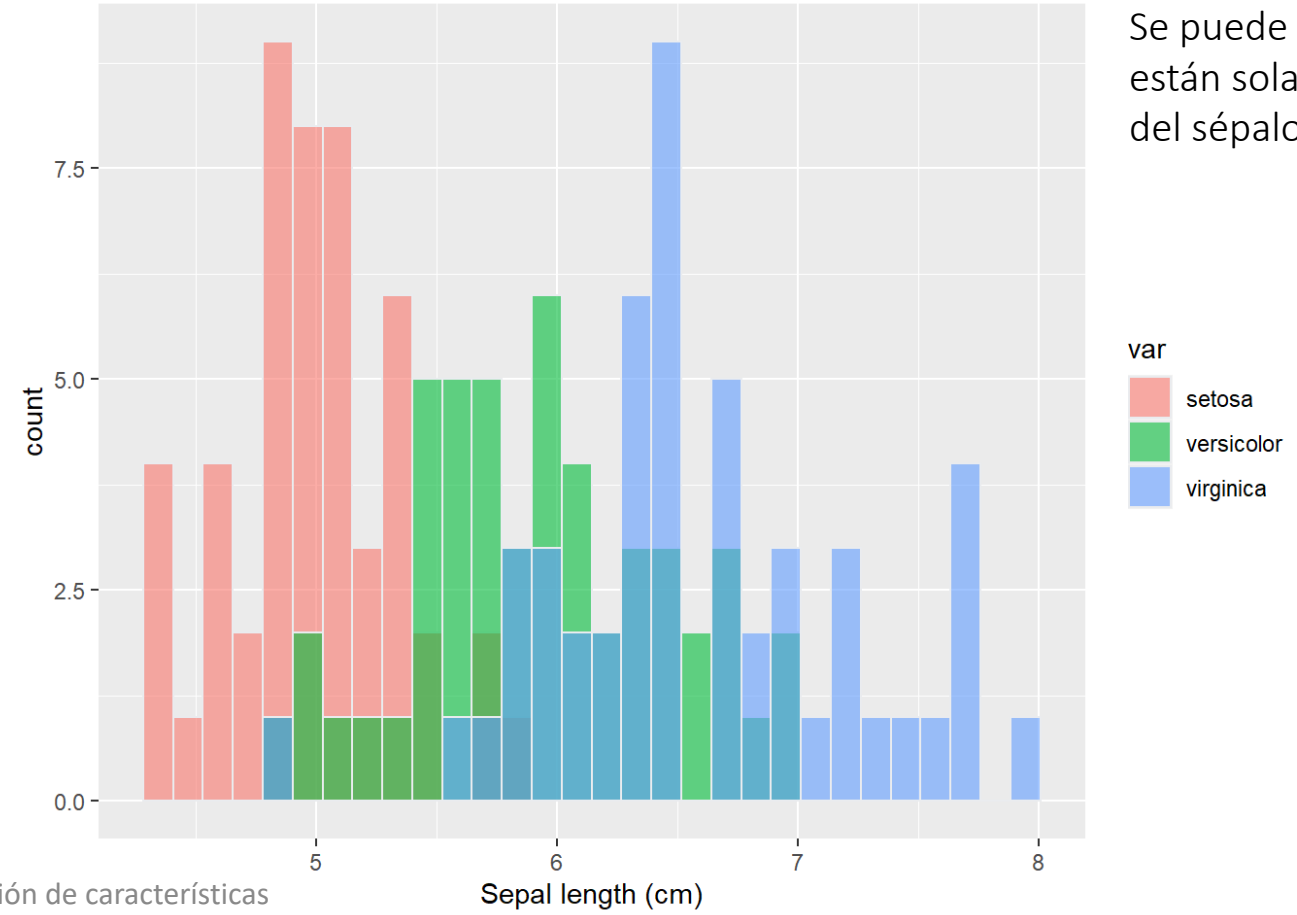
Relación entre las variables

A simple vista, puede observarse cierta correlación entre algunas de las variables:



Clasificación con una variable

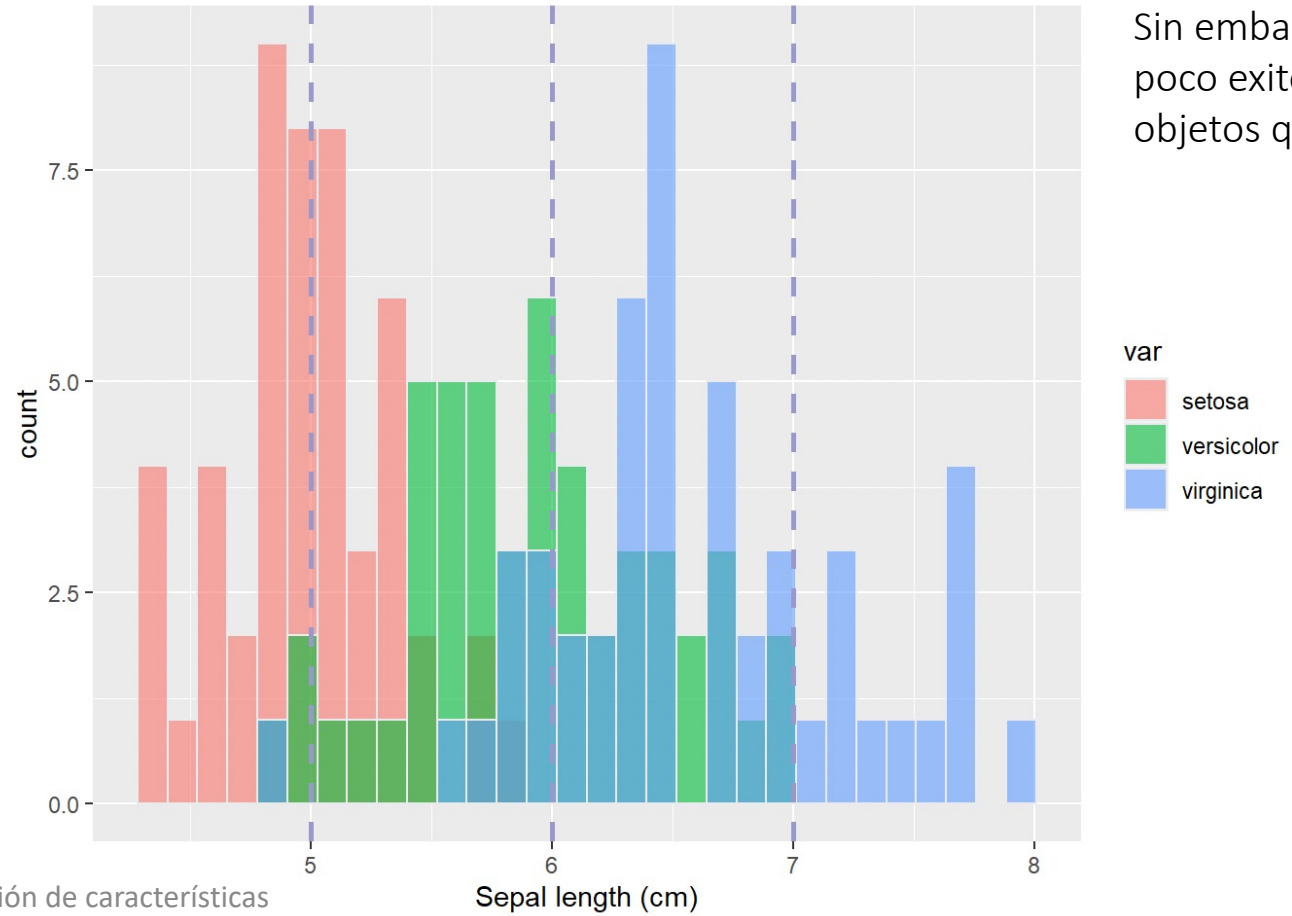
Representemos el histograma de cada clase (variable Species) en base a la longitud del sépalo (variable Sepal.Length):



Se puede observar que las distintas clases están solapadas a lo largo de la longitud del sépalo.

Clasificación con una variable

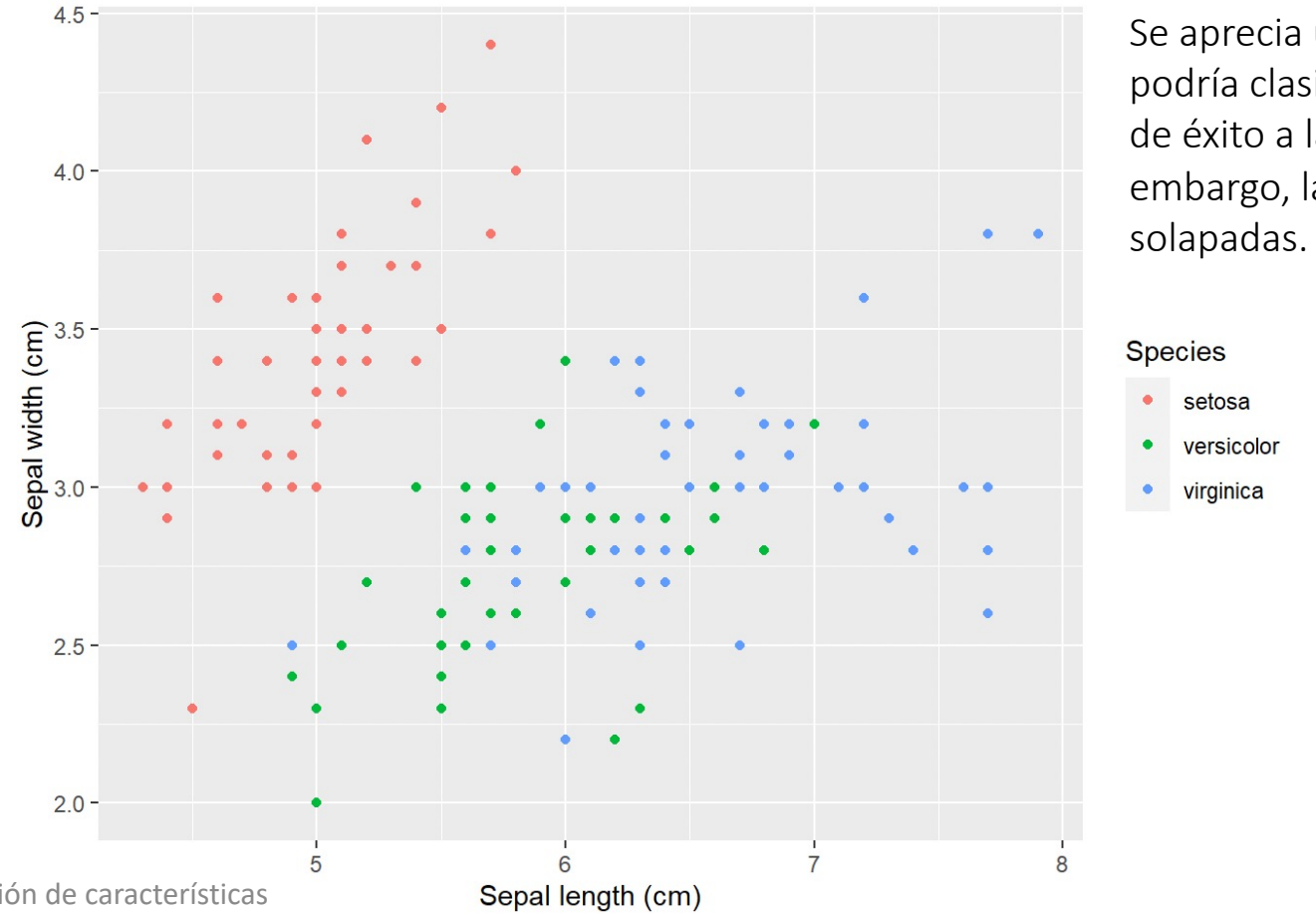
Se podría clasificar particionando por la longitud del sépalo y tomando la clase mayoritaria de cada partición:



Sin embargo, la clasificacion sería poco exitosa, ya que muchos objetos quedarían mal clasificados.

Clasificación con dos variables

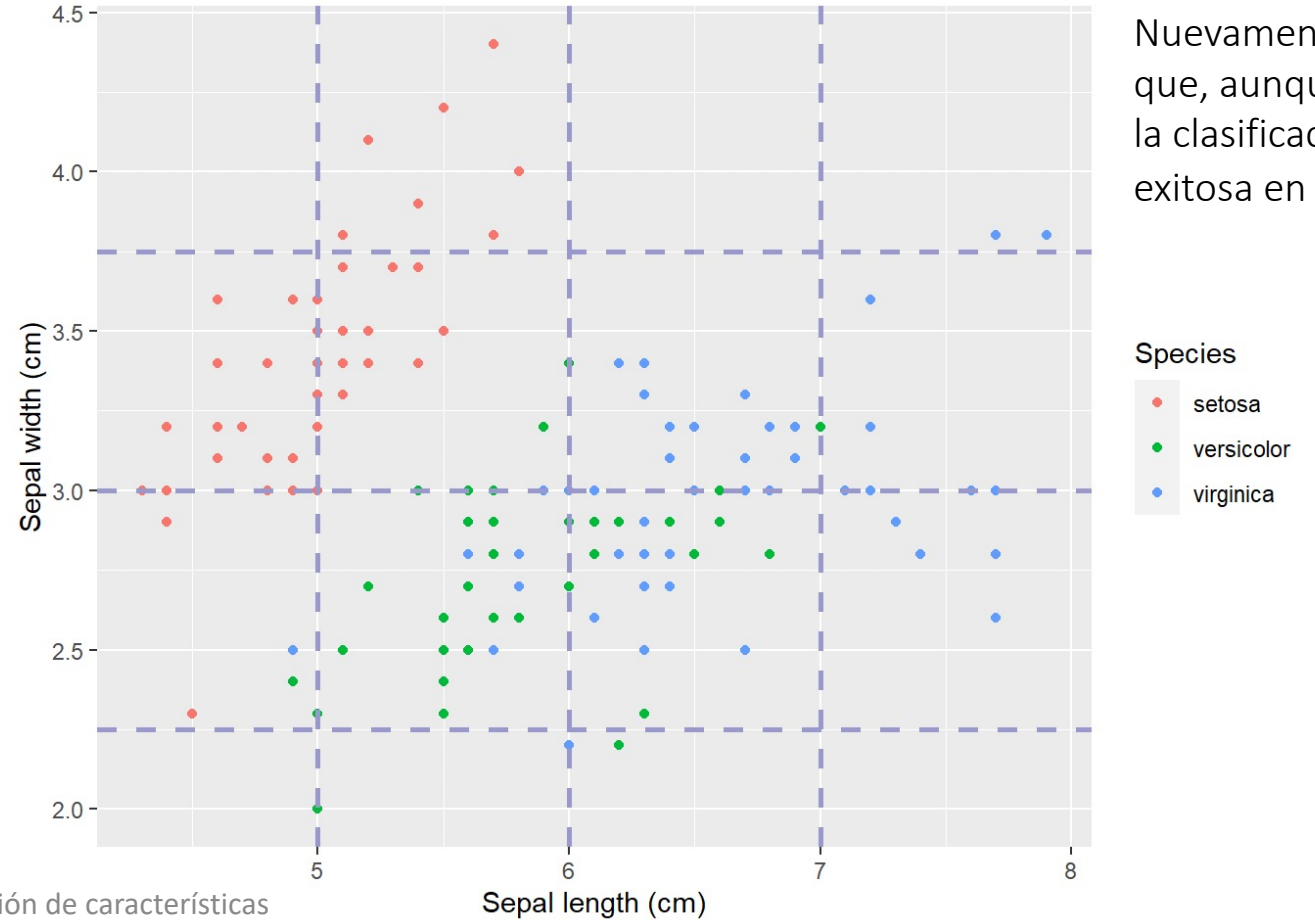
Si se añade el ancho del sépalo (variable Sepal.Width) para intentar mejorar la clasificación:



Se aprecia una clara mejora. Se podría clasificar con un alto grado de éxito a la clase Setosa. Sin embargo, las otras dos clases siguen solapadas.

Clasificación con dos variables

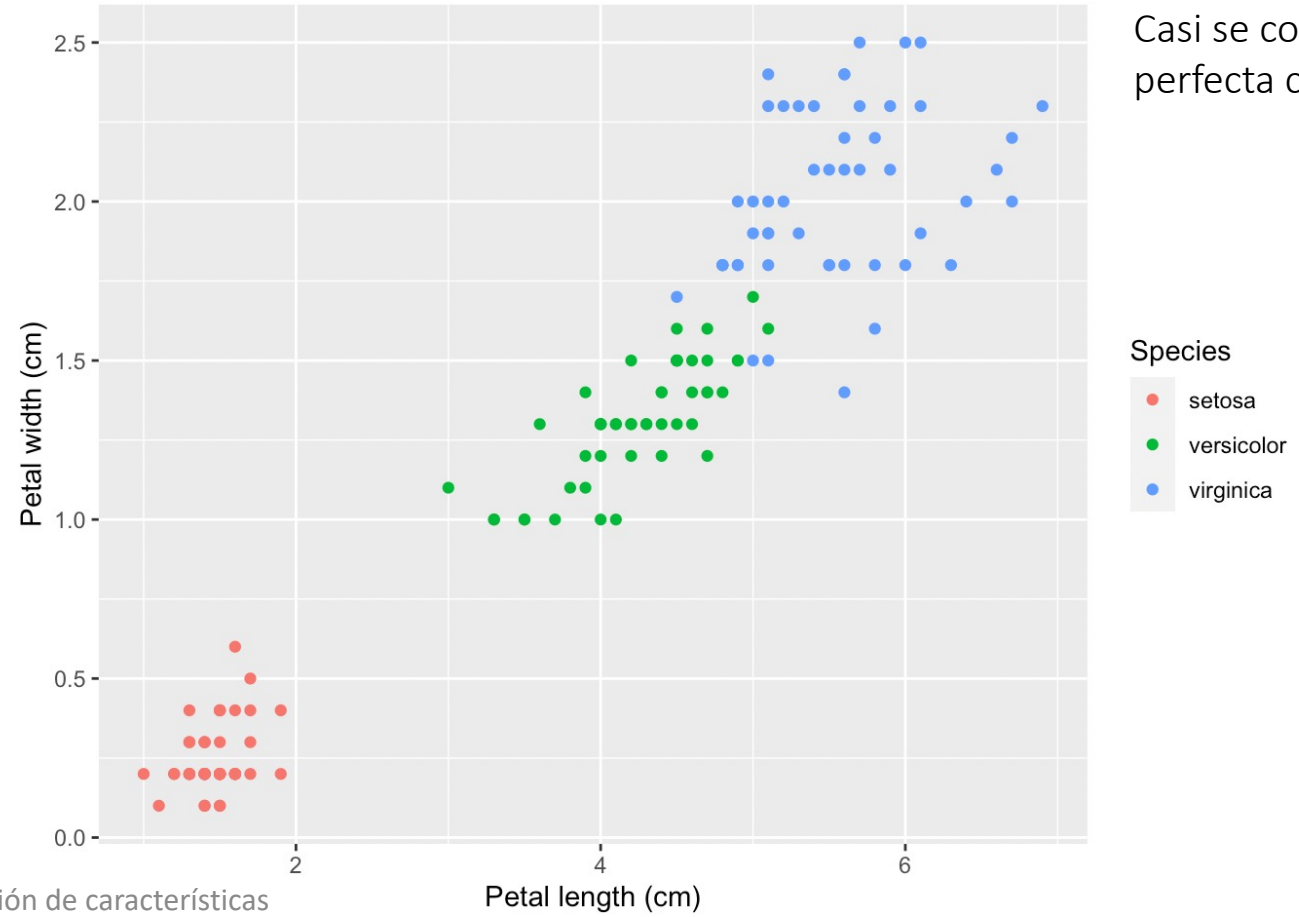
Igual que antes, se podría particionar el espacio de datos, igual que antes, usando ancho y largo del sépalo, y clasificar en base a la clase mayoritaria:



Nuevamente, se puede observar que, aunque se ha mejorado algo, la clasificación seguiría siendo poco exitosa en muchas rejillas.

Clasificación con dos variables

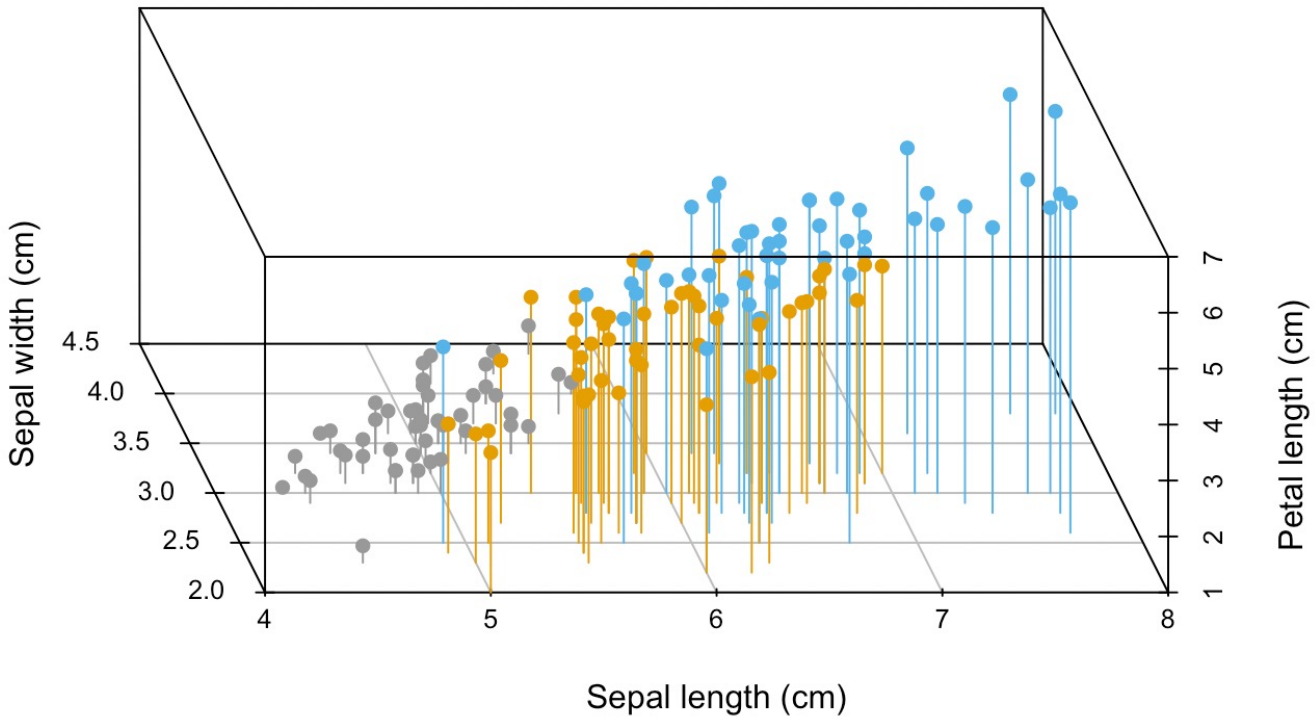
El orden de elección de las variables es también muy importante. A continuación se puede ver cómo quedarían los objetos si las variables elegidas son la longitud y el ancho del pétalo en lugar del sépalo:



Casi se consegue una clasificación perfecta con estas dos variables.

Clasificación con tres variables

Si se añade una tercera variable, la longitud del pétalo, a la longitud y ancho del sépalo:



Se puede observar que la clasificación ha mejorado. Cada vez que aumentamos el número de variables, la representación se hace más dispersa y es más fácil distinguir las clases.

¿A mayor número de variables, mejores resultados?

La maldición de la dimensionalidad

- En iris hay 150 registros. Supongamos una partición 100/50 para entrenamiento/prueba.

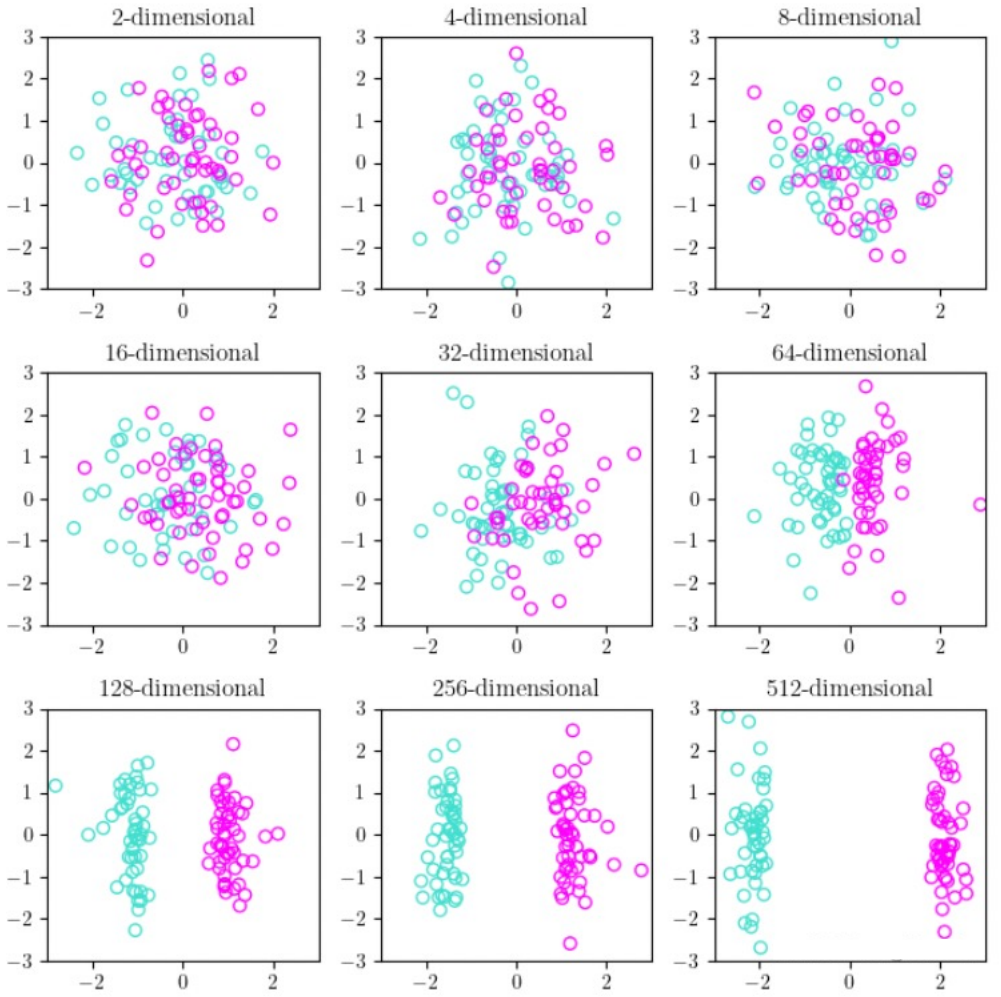
- Si particionamos el espacio en 4 partes, el número promedio de ejemplos por sub-espacio de la partición sería:
  - 1 variable: 25 ejemplos (100/4)
  - 2 variables: 6.25 ejemplos ($100/4^2$)
  - 3 variables: 1.56 ejemplos ($100/4^3$)
  - 4 variables: 0.39 ejemplos ($100/4^4$)
  - ….
  - 10 variables: 0.000095 ejemplos ($100/4^{10}$)

¿Cómo de cercano es el vecino más cercano?

La maldición de la dimensionalidad

- A medida que añadimos más características:
  - Los datos disponibles en nuestro espacio de características se vuelven exponencialmente más escasos.
  - Facilita la separación de los datos.

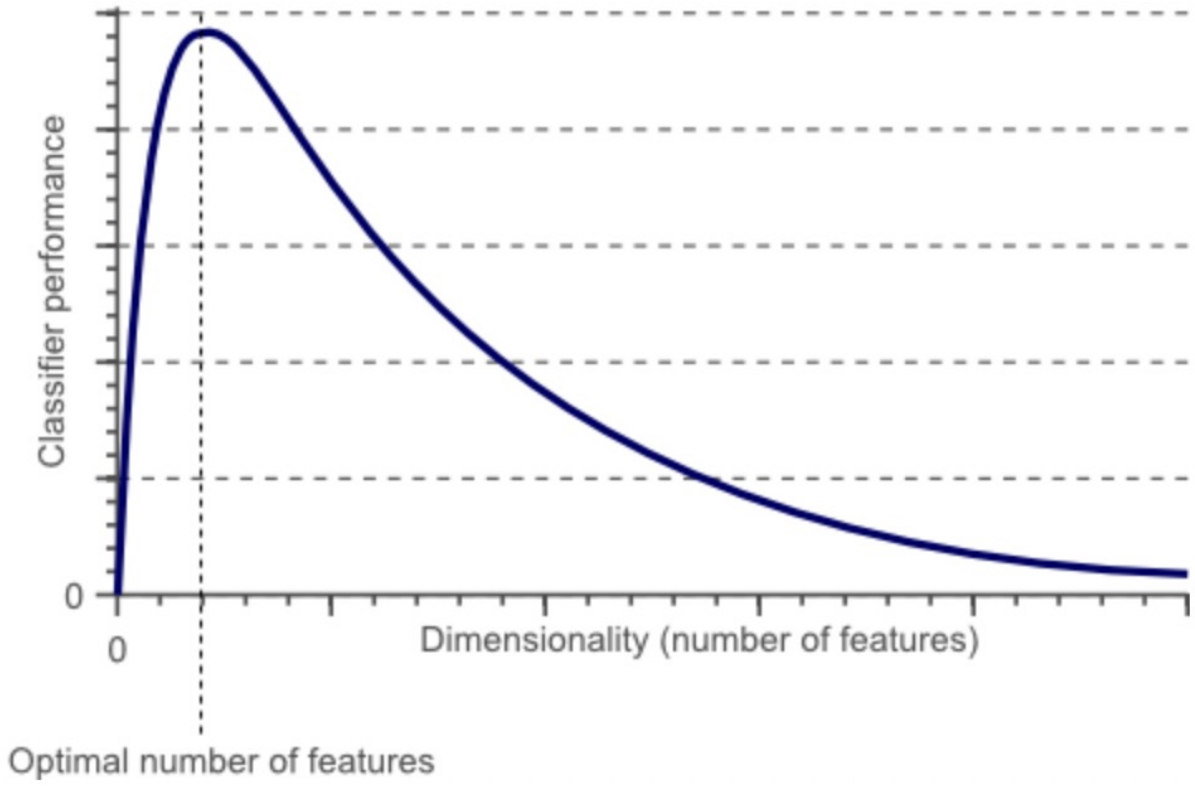
- Sin embargo, esto no se debe a ningún patrón en los datos, en realidad es sólo la naturaleza de los espacios de mayor dimensión.



La maldición de la dimensionalidad

- El uso de un gran número de variables nos lleva a caer en un problema de sobreajuste:

La maldición de la dimensionalidad
Para mantener una buena estimación del modelo, el número de instancias debería crecer exponencialmente con el número de variables.



La maldición de la dimensionalidad

- Este problema se acentúa cuando el **número de características es muy grande**, pero hay **pocas instancias disponibles**.

Data Set	#Instances	#Features	#Classes	Keywords
GLI	85	22283	2	Microarray, Bio
GLA-BRA	180	49151	4	Microarray, Bio
CLL-SUB	111	11340	3	Microarray, Bio
TOX	171	5748	4	Microarray, Bio
SMK-CAN	187	19993	2	Microarray, Bio

Fuente: http://featureselection.asu.edu/old/datasets.php

- No afecta igual a todos los algoritmos:
  - **Naive Bayes**: estima tablas $P(Xi|C)$. No necesita más datos al crecer la dimensionalidad. Su complejidad es $\mathcal{O}(kn)$, por lo que para GLI necesitaría $89.132\ (4{\times}22.283)$ valores de probabilidad.
  - **Averaged one-dependence estimators (AODE)**: estima tablas $P(Xi|Xj,C)$. Necesita más datos al crecer la dimensionalidad. Su complejidad es $\mathcal{O}(kn^2)$, por lo que para GLI necesitaría $1.986.128.356\ (4{\times}22.283^2)$ valores de probabilidad.

La maldición de la dimensionalidad

- En general, en problemas con alta dimensionalidad tendremos:
  - Mayor nivel de ruido y, por consiguiente, mayor error.
  - Escasez de ejemplos para lograr buenas estimaciones.
  - Incremento en tiempo de aprendizaje e inferencia.
  - Obtención de modelos más complejos, menos comprensibles y con mayor necesidad de memoria.
  - Muchas características suelen implicar un sobreajuste.

Solución: reducción de la dimensionalidad

- Extracción de características
  - Transforma el espacio de representación, creando nuevas variables mediante combinación de las existentes:

$$\{X_1, \dots, X_n\} \rightarrow \{Z_1, \dots, Z_m\}, t.q. m < n;\ Z_j = f(subset(X_1, \dots, X_n))$$

  - El objetivo es obtener una representación de menor dimensionalidad que retenga la mayor parte de la información del conjunto inicial

- Selección de variables/características

Selección de características (feature selection)

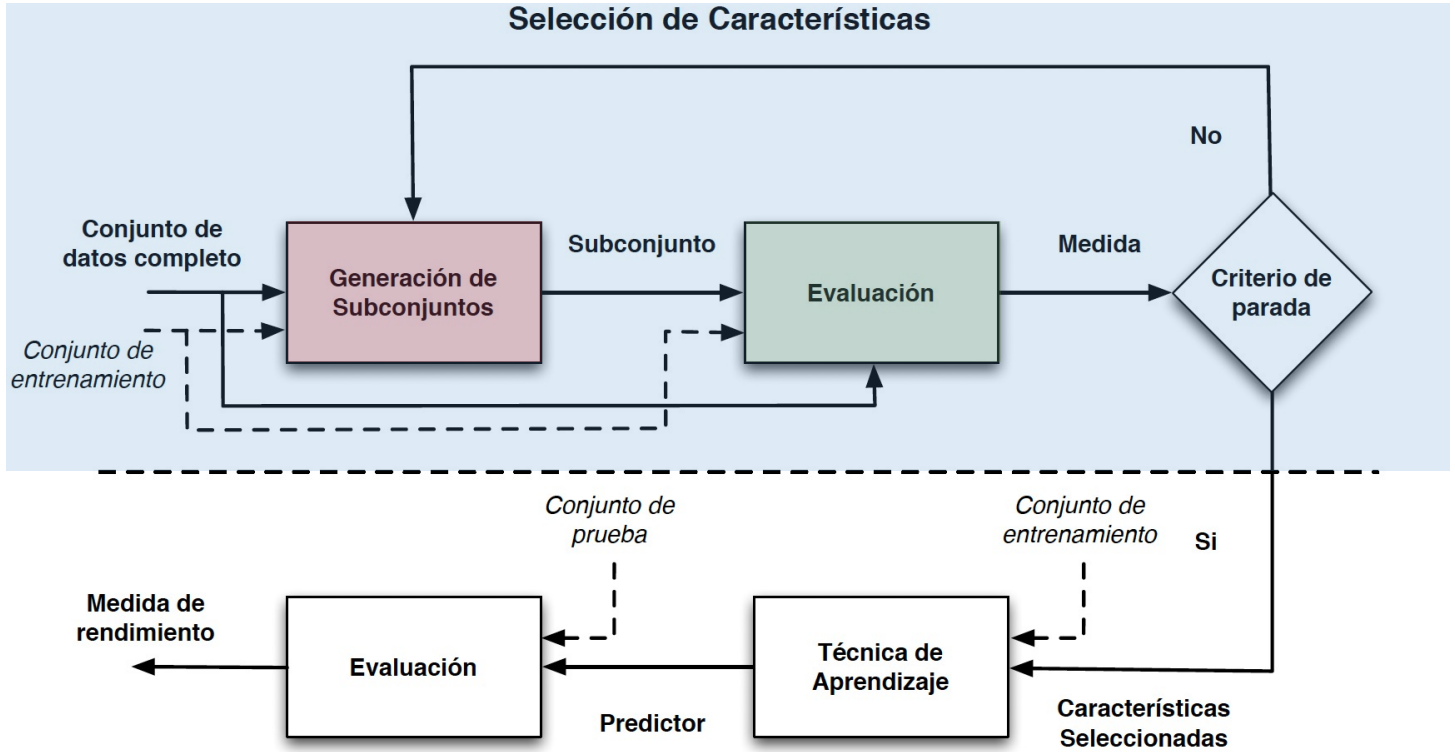
- Objetivo: Encontrar las k dimensiones, de las d dimensiones originales, que aportan la mayor cantidad de información, descartando el resto $(d - k)$.
  - Se trata de encontrar el subconjunto óptimo de variables de acuerdo con alguna determinada función objetivo.
  - Sólo se selecciona un subconjunto de las dimensiones originales sin transformar.

- ¿Por qué usar selección de características?
  - Disponer de más variables no siempre implica una mayor **capacidad de clasificación del modelo**.
  - Utilizar menos atributos puede **reducir el sobreajuste**.
  - Modelos más simples requieren menos instancias.
  - La **complejidad de los algoritmos** de aprendizaje suele ser función del número de atributos.
  - En muchos casos, la obtención de los valores para algunos atributos es costosa (sensores, pruebas complejas/dolorosas, etc).

Métodos de evaluación

Para definir una estrategia de selección de características es necesario especificar:

1. Una técnica de búsqueda que explore el espacio de posibles subconjuntos
2. Una función de evaluación que puntúe un subconjunto de acuerdo a su potencial para la tarea objetivo



¿Cómo evaluar el proceso de selección?

El proceso de selección de características puede verse como un proceso de optimización:

- Complejidad del proceso de selección (número de subconjuntos evaluados)
- Cardinalidad del subconjunto seleccionado.
- Generalidad del conjunto de características seleccionadas.
- Estabilidad del subconjunto de características seleccionado.
- Métricas del proceso de clasificación (acierto, AUC, etc.)

Dados n variables, hay 2^n subconjuntos posibles a evaluar:

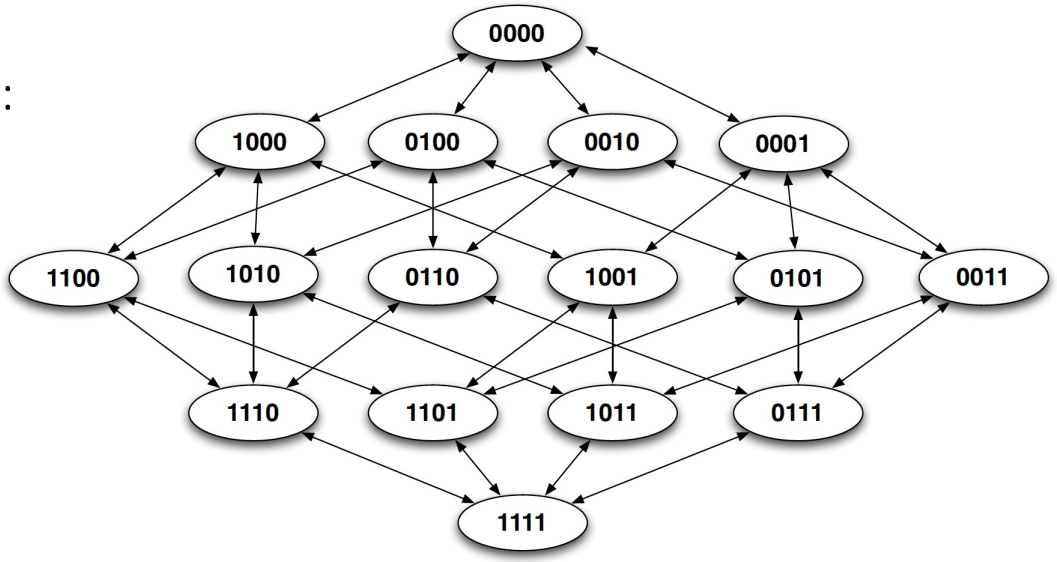
$n = 20$ entonces hay $2^{20} = 1.048.576$ subconjuntos

$n = 100$ entonces hay $2^{100} = 1.125.899.906.842.624$ subconjuntos

$n = 100.000$ entonces hay $2^{100.000} = \ldots$

¿Búsqueda exhaustiva?

Espacio de estados para 4 variables



¿Cómo evaluar el proceso de selección?

Si limitamos a $m \leq n$ el tamaño de los subconjuntos, entonces hay $\binom{n}{m}$ subonjuntos.

- $n = 20, m = 10$ entonces hay $\binom{20}{10} = 184.756$ subconjuntos
- $n = 100, m = 10$ entonces hay $\binom{100}{10} = 1{,}73 \times 10^{13}$ subconjuntos
- $n = 100.000, m = 10$ entonces hay $\binom{100.000}{10} = 2{,}75 \times 10^{43}$

¿Búsqueda exhaustiva?

Sigue siendo un valor muy grande... Se necesita una estrategia de búsqueda

Estrategias de búsqueda

Se pueden utilizar diferente estrategias de búsqueda en el espacio de posibles subconjuntos:

- Exhaustiva. Se barre todo el espacio de posibles subconjuntos.
  - Garantiza solución óptima.
  - Se puede recorrer el espacio en profundidad o en anchura.
  - Computacionalmente intratable $\mathcal{O}(2^n)$ (solo es posible para pocas características).

- Aleatoria. Explora un pequeño número de subconjuntos.
  - No garantiza la solución óptima.
  - Mediante una pequeña transformación, se va modificando la configuración inicial para dirigir la búsqueda hasta la solución final.
  - Complejidad computacional $\mathcal{O}(kn)$ (no suelen funcionar).

- Heurística. Disponen de alguna información sobre qué subconjunto es el más prometedor.
  - No garantiza la solución óptima.
  - Normalmente encuentran una buena solución en un tiempo razonable.
  - Algoritmos de rankings $\mathcal{O}(n)$, algoritmos secuenciales $\mathcal{O}(n^2)$, algoritmos metaheurísticos.

Rankers

- Se evalúa cada característica por separado.
- Una vez evaluadas todas las variables, se crea un ranking $\boldsymbol{X_1}, \boldsymbol{X_2}, \cdots, \boldsymbol{X_n}$, donde X_{j-1} precede a X_j en el ranking si y solo si tiene mejor valor de evaluación.
- Se seleccionan las primeras $\boldsymbol{k}$ variables del ranking.
- Se suelen basar en test estadísticos:
  - Paramétricos: t-test, ANOVA, Test de Welch, ...
  - No paramétricos: Test de rangos de Wilcoxon, test de Kruskal-Wallis, ...

Ventajas:

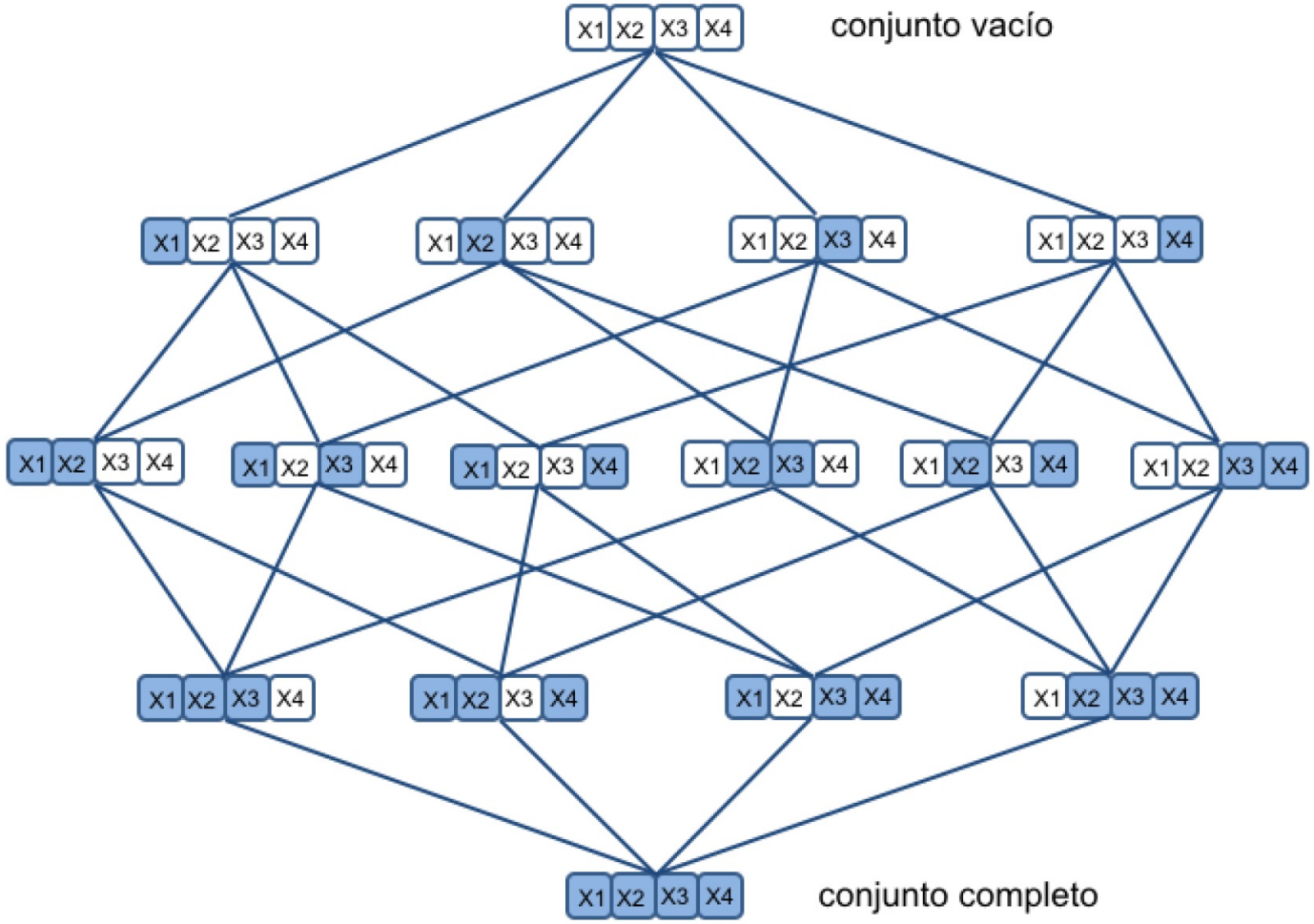
- Computacionalmente eficiente y escalable.

Desventajas:

- ¿Cómo se fija el valor de k?
- Pueden escogerse variables redundantes.
- Pueden descartarse variables, en principio irrelevantes, que en combinación con otras sean informativas.

Búsqueda secuencial

- Son métodos voraces que exploran el espacio de soluciones de forma local, añadiendo o eliminando un atributo en cada iteración.



Búsqueda secuencial hacia delante

- Comienza con $S = \emptyset$ y de manera secuencial añade al subconjunto actual S el atributo X_i que maximiza $f(S \cup \{X_i\})$.

- Funciona mejor cuando el subconjunto óptimo tiene pocas variables
- Evalúa conjuntos mucho más pequeños, por lo que es más rápido.
- Tiende a evaluar menos subconjuntos.

```
1:  procedimiento SELECCIÓNFORWARD(D datos)
2:      S ← ∅; X ← ATRIBUTOS(D)
3:      evalS ← EVALUA(S,D)
4:      hacer
5:          evalSX_i ← máx_{X_i∈X} EVALUA(S ∪ {X_i},D)
6:          X_i ← arg máx_{X_i∈X} EVALUA(S ∪ {X_i},D)
7:          si MEJOR(evalSX_i,evalS) entonces
8:              S ← S ∪ {X_i};  X ← X \ {X_i}
9:              evalS ← evalSX_i
10:         fin si
11:     mientras S cambia y X ≠ ∅
12:     devolver S
13: fin procedimiento
```

Búsqueda secuencial hacia atrás

- Comienza con el conjunto completo $\boldsymbol{S = \{X_1, \ldots, X_n\}}$ y de manera secuencial elimina al atributo $X_i \in S$ que maximiza $f(S \setminus \{X_i\})$.

- Funciona mejor cuando el subconjunto óptimo tiene muchas variables
- Evalúa conjuntos grandes, por lo que es más lento. No es útil con alta dimensionalidad.

```
1:  procedimiento SELECCIÓNBACKWARD(D datos)
2:      S ← X
3:      evalS ← EVALUA(S,D)
4:      hacer
5:          evalSX_i ← máx_{X_i∈S} EVALUA(S \ {X_i},D)
6:          X_i ← arg máx_{X_i∈S} EVALUA(S \ {X_i},D)
7:          si MEJOR-O-IGUAL(evalSX_i, evalS) entonces
8:              S ← S \ {X_i}
9:              evalS ← evalSX_i
10:         fin si
11:     mientras S cambia y S ≠ ∅
12:     devolver S
13: fin procedimiento
```

Métodos más avanzados de búsqueda

- Estrategias específicas que intentan determinar subconjuntos óptimos: CMIN (Fast FSS using Conditional Mutual Information); FCBF (Fast Correlation Based Filter); mRMR (minimum Redundancy Maximum Relevance); etc.
- Algoritmos deterministas más sofisticados, con vuelta atrás, etc.: Best first search, Beam search, Branch and Bound, etc.
- Algoritmos no deterministas. Algoritmos estocásticos, p.e. metaheurísticas, que evalúan un porcentaje pequeño del espacio de búsqueda, pero de forma guiada, aprendiendo del paisaje del espacio de búsqueda a partir de las soluciones evaluadas.

La selección de características es un problema NP-completo

Ninguno de los métodos heurísticos usados garantiza la obtención del mejor subconjunto posible

Evaluación

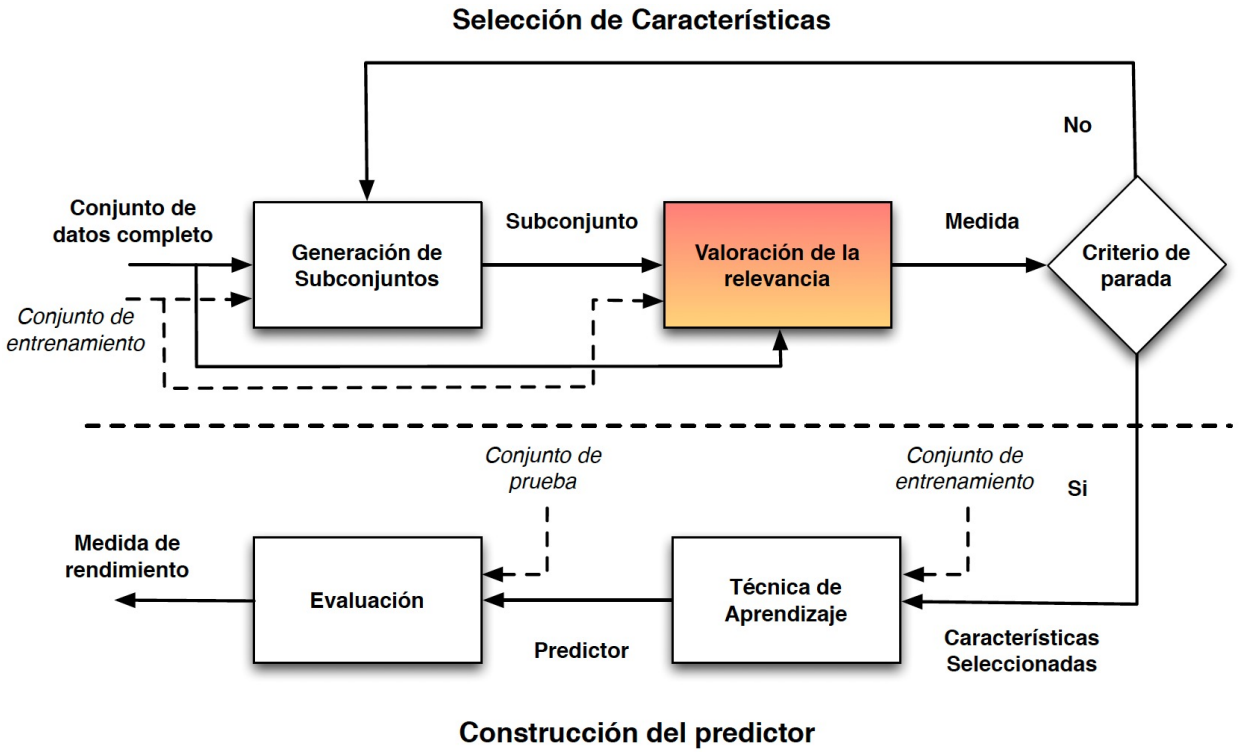
Existen 3 enfoques de evaluación:

- **Métodos basados en filtros** (Filters)
  - Cada conjunto de candidatos S es evaluado usando medidas (estadísticas, etc.) que se obtienen directamente de los datos.
- **Métodos basados en envoltura** (Wrappers)
  - Cada conjunto de candidatos S es evaluado por una medida de rendimiento (acierto, error, etc.), obtenida por un algoritmo de aprendizaje cuando solamente se usan los atributos en S.
- **Métodos empotrados**
  - El proceso de selección de características está integrado en el algoritmo de aprendizaje.

Métodos basados en filtros

El valor de la función de evaluación para un subconjunto de atributos S se obtiene a partir de medidas obtenidas únicamente a partir de los datos:

- Medidas de información
  - Mayoritariamente basadas en entropía de Shannon.
  - Ganancia de información, incertidumbre simétrica, información mutua, etc.
- Distancias o medidas de separación entre clases
  - Euclídea, Manhattan, coseno, etc.
- Medidas de dependencia entre variables
  - Coeficiente de correlación de Pearson, estadísticos, etc.



Medidas de información

- Estas medidas están basadas en la ganancia de información proporcionada por las distintas características.
  - Ganancia de información, incertidumbre simétrica, información mutua, ...
- La ganancia de información se puede interpretar como la reducción de la incertidumbre al clasificar, dado un conjunto de elementos de nidos a través de un conjunto de características.
- Supongamos:
  - Un conjunto de datos definido a través de n características $X = \{x_1, x_2, \dots, x_n\}$.
  - Cada elemento de X está etiquetado como perteneciente a una clase del conjunto $C = \{c_1, c_2, \dots, c_l\}$.
- La incertidumbre inicial teniendo en cuenta solo la información procedente de la distribución de las clases se puede medir mediante la entropía:

$$E(C) = -\sum_{i=1}^{l} P(c_i) log_2 P(c_i)$$

- Mediante la entropía condicional podemos medir la incertidumbre asociadas a las clases teniendo en cuenta la información proporcionada por el conjunto datos X:

$$E(C|X) = -\sum_{j=1}^{n} P(x_j) \left(\sum_{i=1}^{l} P(c_i|x_j) log_2 P(ci|x_j) \right)$$

Medidas de información

- La ganancia de información teniendo en cuenta la información proporcionada por X es:

$$IG(C|X) = E(C) - E(C|X)$$

- $IG(C|X)$ nos da una medida de la capacidad del conjunto de características X a la hora de predecir las clases.
- El conjunto de características X es totalmente irrelevante si la ganancia de información es igual a 0.

Problema:

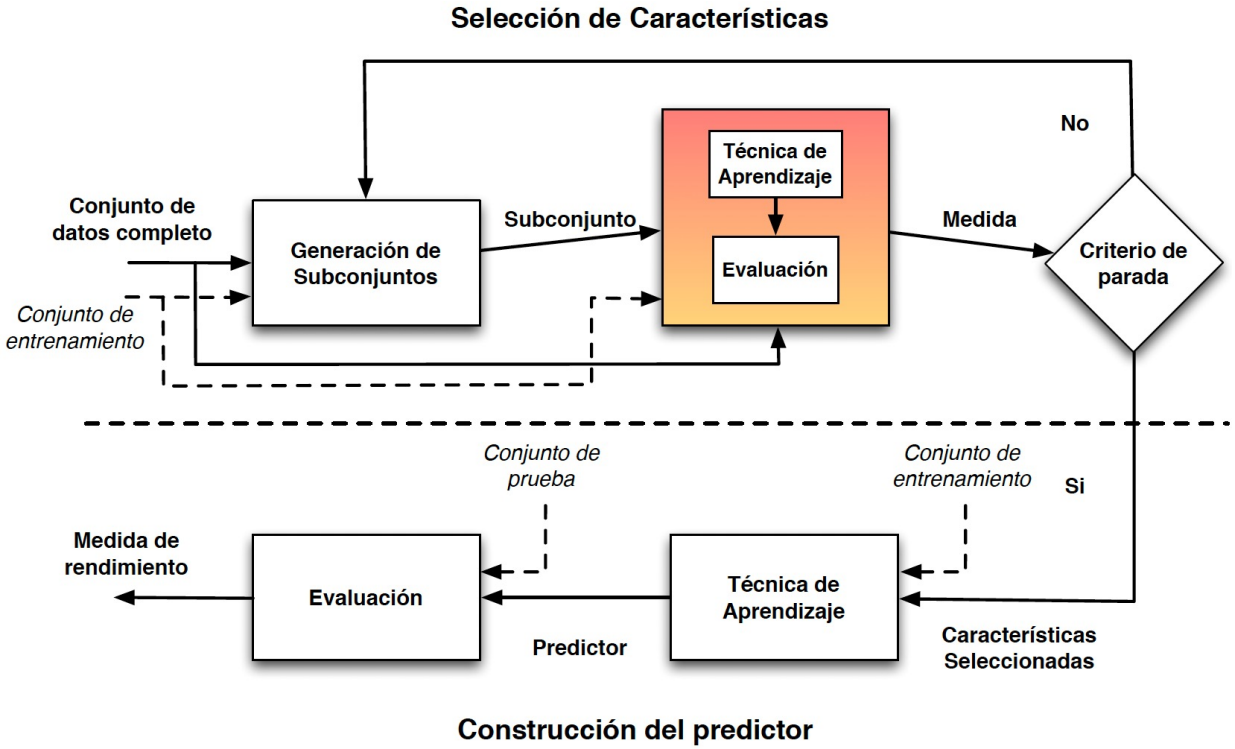
- Computacionalmente costoso.
- Es poco práctico en problemas de alta dimensionalidad y pocos datos, por la dificultad de estimar las probabilidades condicionales.

Solución:

- Aproximar la métrica.
- MIFS (Mutual Information Feature Subset), CFS (Correlation Feature Subset).

Métodos basados en envoltura (Wrappers)

- La evaluación de cada uno de los subconjuntos candidatos se realiza mediante la construcción de un clasificador (o regresor).
- Por lo tanto, como medida de evaluación utilizan la capacidad predictiva del clasificador.
- De esta forma, se selecciona el subconjunto de características que produce el mejor clasificador.
- Ventajas:
  - Producen mejores resultados al estar orientados al problema de clasificación.
- Desventajas:
  - Tienen un mayor riesgo de sobreajuste que los filtros.
  - Son más costosos computacionalmente.



Eliminación recursiva de características

- El mismo esquema se puede adaptar para una búsqueda hacia delante de subconjuntos.
- Se empieza por un conjunto de tamaño 1.
- Se van agregando las características más relevantes.
- La principal desventaja de este tipo de métodos, a parte de su coste computacional, es el sobreajuste.

Algoritmo Eliminación recursiva de características

1: Construir un modelo con todas las características
2: Evaluar el modelo
3: Calcular la relevancia de las características
4: Crear una lista con las características ordenadas de mayor a menor relevancia.
5: **para** $size = n$ to 1 **hacer**
6: Crear un conjunto S_{size} con las $size$ características más relevantes
7: Construir un modelo utilizando las características S_{size}
8: Evaluar el modelo
9: [Opcional] Recalcular la relevancia de las características
10: **fin para**
11: Crear una lista con todos los S_i y el resultado de la evaluación
12: Determinar el subconjunto óptimo S_{opt}

Eliminación recursiva de características

- Para evitar el sobreajuste, se incluye un bucle externo para llevar a cabo un remuestreo.

Wrappers más representativos:

- OBLIVION [Langley and Sage, 1994]: búsqueda voraz y árboles de decisión.
- RFE+SVM [Guyon et al., 2002]: eliminación recursiva de características y SVM.
- FFE + NNets [Goutte, 1997]: búsqueda hacia delante y redes neuronales.
- GA + C4.5 [Abbasimehr and Alizadeh, 2013]: búsqueda aleatoria y C4.5.

Algoritmo Eliminación recursiva de características con remuestreo

1: **para** *Cada iteración de remuestreo* **hacer**
2: Crear los conjunto de entrenamiento E y prueba T
3: Construir un modelo sobre E con todas las características
4: Evaluar el modelo en T
5: Calcular la relevancia de las características
6: Crear una lista con las características ordenadas de mayor a menor relevancia.
7: **para** $size = n$ to 1 **hacer**
8: Crear un conjunto S_{size} con las $size$ características más relevantes
9: Construir un modelo utilizando las características S_{size}
10: Evaluar el modelo
11: [Opcional] Recalcular la relevancia de las características
12: **fin para**
13: **fin para**
14: Crear una lista con todos los S_i y el resultado de la evaluación
15: Determinar el subconjunto óptimo S_{opt}
16: Crear un modelo con las variables S_{opt} y con el conjunto de entrenamiento original.

Conclusiones

- En la actualidad, es crucial la aplicación de técnicas de reducción de la dimensionalidad antes de abordar la creación de modelos predictivos.
- Estas técnicas se pueden agrupar en dos grandes grupos: Técnicas de extracción de características y selección de características.
- En este capítulo nos hemos centrado en las técnicas de selección de características, que se centran en encontrar un conjunto óptimo de características de acuerdo con algún criterio.
- Dependiendo del criterio elegido las técnicas puedes ser de tipo filtro o de tipo envoltura (wrapper):
  - Las técnicas de tipo filtro solo se apoyan en las propiedades intrínsecas de las características.
  - Las técnicas de tipo wrapper se basan en la construcción de clasificadores (o regresores) para evaluar la idoneidad de los conjuntos de características.
- Un aspecto importante en ambos tipos de métodos es la técnica de búsqueda utilizada para recorrer el espacio de búsqueda de posibles subconjuntos de características.
  - Búsquedas exhaustivas, heurísticas o aleatorias.