Parcial 1. Clasificación, regresión y series temporales

Contenido

[1. Técnicas de clasificación 3](#_Toc86862207)

[1. Etapas del proceso de clasificación 3](#_Toc86862208)

[2. Criterios para evaluar un clasificador 4](#_Toc86862209)

[3. Algunos ejemplos de clasificadores sencillos 5](#_Toc86862210)

[4. Técnicas de validación de clasificadores 5](#_Toc86862211)

[Hold-out 6](#_Toc86862212)

[Validación cruzada 6](#_Toc86862213)

[Leaving-one-out 6](#_Toc86862214)

[Bootstrap (o 0.632 bootstrap) 6](#_Toc86862215)

[2. Clasificación con árboles y reglas 6](#_Toc86862216)

[1. Árboles de decisión 6](#_Toc86862217)

[1.1. Construcción de los árboles de decisión 7](#_Toc86862218)

[1.2. Construcción de árbol de decisión 8](#_Toc86862219)

[1.3. Criterios de selección de variables 9](#_Toc86862220)

[1.4. Particionamiento del espacio con árbol de decisión. 9](#_Toc86862221)

[1.5. Ventajas e inconvenientes 10](#_Toc86862222)

[1.6. Algunos algoritmos basados en árboles de decisión 10](#_Toc86862223)

[2. Clasificadores basados en reglas 11](#_Toc86862224)

[Generación de reglas mediante árboles de decisión 12](#_Toc86862225)

[Generación de reglas mediante cobertura 12](#_Toc86862226)

[3. Clasificación con otras técnicas 13](#_Toc86862227)

[3.1. Clasificadores basados en métodos bayesianos 13](#_Toc86862228)

[Clasificador Naive Bayes 13](#_Toc86862229)

[3.2. Clasificadores basados en instancias 13](#_Toc86862230)

[Algoritmo KNN 14](#_Toc86862231)

[3.3. Clasificadores basados en redes neuronales 14](#_Toc86862232)

[3.4. Clasificadores basados en máquinas de soporte vectorial 15](#_Toc86862233)

[4. Multiclasificadores 16](#_Toc86862234)

[4.1. Combinación de clasificadores 16](#_Toc86862235)

[4.2. Bagging 16](#_Toc86862236)

[4.3. Boosting 16](#_Toc86862237)

[4.4. Descomposición de problemas de multiclase. 16](#_Toc86862238)

[5. Integración, limpieza y transformación 17](#_Toc86862239)

[5.1. Integración de datos 17](#_Toc86862240)

[5.2. Limpieza de datos 17](#_Toc86862241)

[5.3. Transformación de los datos 17](#_Toc86862242)

[Min-Max 17](#_Toc86862243)

[Zero-mean (Z-score) 17](#_Toc86862244)

[Por escala decimal 17](#_Toc86862245)

[6. Datos imperfectos 18](#_Toc86862246)

[6.1. Valores perdidos 18](#_Toc86862247)

[6.2. Tipos de ruido 18](#_Toc86862248)

[Ensemble Filter (EF) 18](#_Toc86862249)

[Cross-Validated Committees Filter (CVCF) 18](#_Toc86862250)

[Iterative Pertitioning Filter (IPF) 18](#_Toc86862251)

[6.3. Outliers 19](#_Toc86862252)

[7. Reducción de datos 19](#_Toc86862253)

[7.1. Selección de características 19](#_Toc86862254)

# Técnicas de clasificación

Los problemas de clasificación es predecir una clase determinada (categoría) para un objeto. Se construye un modelo (entrenamiento) que permita clasificar nuevos casos. El aprendizaje es **supervisado** porque se conoce la verdadera categoría (clase) de cada uno de los ejemplos de entrenamiento. El **modelo** (clasificador) puede ser un conjunto de reglas, árbol, etc.

*El problema de la clasificación está* relacionado con la separabilidad de las clases

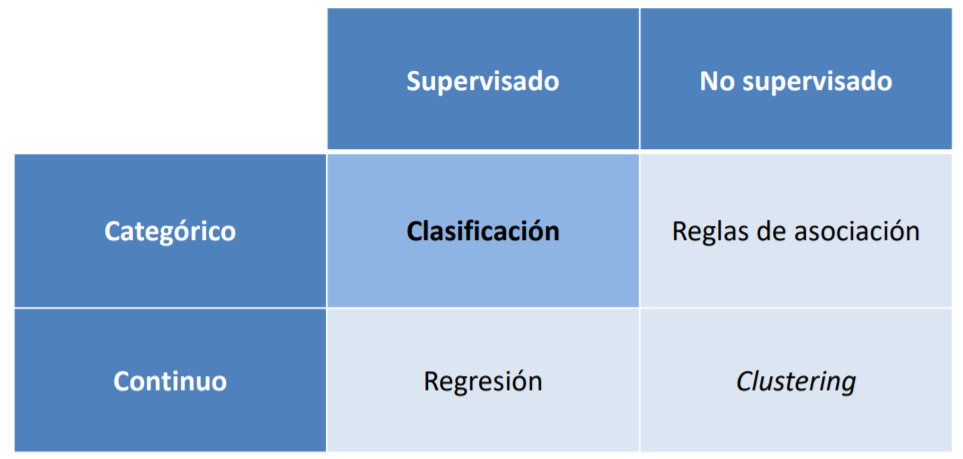
Existen dos tipos de variables:

* **Numérica** (existe un orden). A su vez, existen dos tipos de numéricas:

Continua: temperatura, energía, peso, etc

Discreta: edad, nivel de educación, número de hijos, etc

* **Categórica o nominal** (no hay orden): pueden ser booleanas u otras

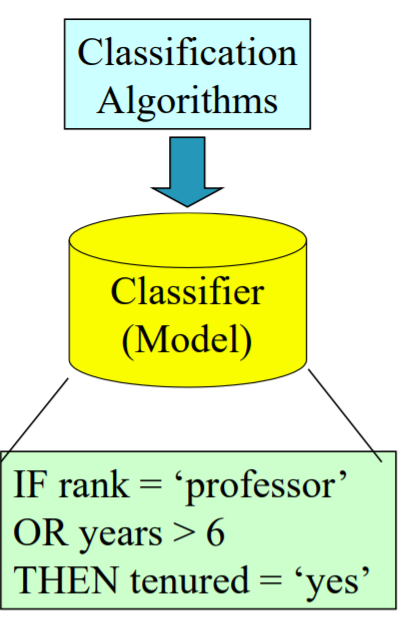


## Etapas del proceso de clasificación

Inicialmente se parte el dataset en dos partes: conjunto de entrenamiento y conjunto de prueba.

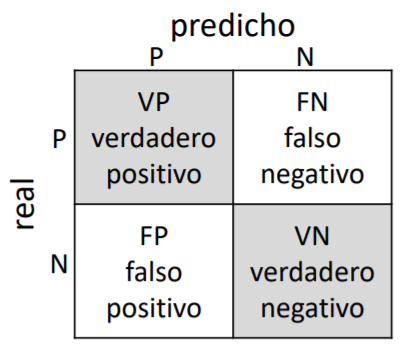
1. Construcción del modelo: **Fase de aprendizaje**

Se utiliza un conjunto de ejemplos para la construcción del modelo (*conjunto de entrenamiento).* Nos devuelve algo así como un conjunto de reglas de clasificación/árboles de decisión/fórmula,etc. Un “*programa”.*



1. **Validación:** Se prueba le modelo con un conjunto de datos distintos a los de entrenamiento (*conjunto de prueba)*. Para cada ejemplo de test se compara la clase determinada por el modelo con la clase real.

## Criterios para evaluar un clasificador

Vamos a ver lo que predice el modelo y lo que realmente le corresponde.

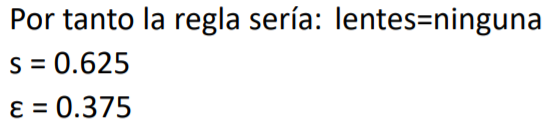
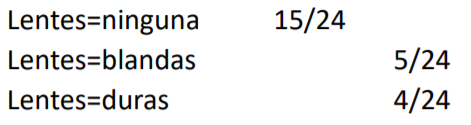
* **Precisión o exactitud:** el porcentaje de acierto y el de error.
* **Velocidad:** tiempo para construir el modelo y tiempo para usarlo.
* **Robustez:** capacidad de tratar con valores desconocidos.
* **Escalabilidad:** Aumento de la velocidad con el tamaño de la BD.
* **Interoperabilidad:** comprensibilidad del modelo.
* **Complejidad del modelo:** Tamaño del árbol, número de reglas, etc.

## Algunos ejemplos de clasificadores sencillos

**Clasificador ZeroR**

Es el más sencillo. Se usa como base para realizar comparaciones. Cualquier algoritmo con menor rendimiento, estaría mal hecho).

La regla sería que las instancias se clasifican como pertenecientes a la mayoritaria. Es decir: la que mayor datos tenga, es la acierto y lo demás el error.



**Clasificador OneR o 1R**

Se utiliza una única variable. Se genera reglas del tipo: si variable = X entonces Clase = Y

Se crea un clasificador formado con reglas de única condición. Es decir: solo se pregunta por un atributo.

Los valores perdidos se utilizan como un valor especial de las variables.

## Técnicas de validación de clasificadores

Utilizar como la medida de bondad la tasa de acierto no es lo mejor, ya que el porcentaje será muy optimista ya que el modelo estará sobreajustado a los datos durante el aprendizaje. Por ello, existe distintas técnicas:

### Hold-out

Se parte el dataset en dos: CE (entrenamiento) y CT(test). El de entrenamiento suele ser más grande. Esta validación se utiliza para BBDD grandes.

### Validación cruzada

Se trata en partir el dataset en K folds y aprender K clasificadores con CE distintos. Al final se sacan medias.

Si hay 100 datos, se hacen 5 carpetas. Se hacen 5 entrenamientos cogiendo 4 carpetas y dejando una para el test. Finalmente tendremos 5 resultados de los que sacaremos la media.

Se usa en BBDD de tamaño moderado.

### Leaving-one-out

Es un caso especial de validación cruzada pero K es igual al número de registros. Se utiliza en BBDD muy pequeñas ya que implica un costo tremendo. K = número de registros.

### Bootstrap (o 0.632 bootstrap)

Si hay 100 elementos, se cogerán 100 elementos de manera aleatoria, aunque sea repetido. Lo bueno es que entra el 100% de los datos. La probabilidad de que un ejemplo no sea elegido es del 0.368%.

El CE tendrá un 63,2% y el CT el resto de la BD.

# Clasificación con árboles y reglas

## Árboles de decisión

Es un clasificador que en función de un conjunto de atributos permite determinar a qué clase pertenece el objeto.

Cada hoja es una clase. Cada nodo es un nodo de decisión que especifica una prueba simple. Los descendientes de cada nodo son los posibles resultados.

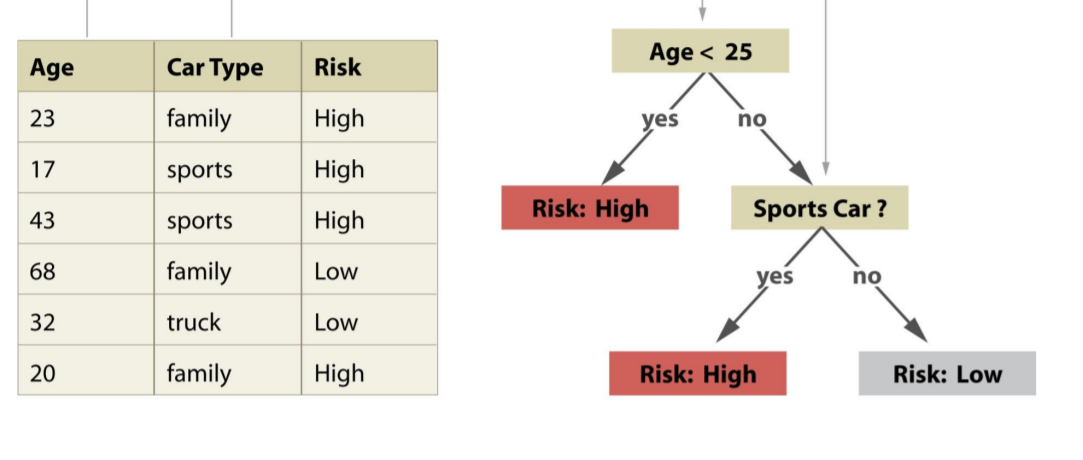
Entonces, cada nodo es un atributo a comprobar (se pregunta por el/los valores de uno de los atributos).

Un árbol de decisión puede transformarse en reglas pero un SBR no puede transformarse a un árbol.

Los atributos deben ser **categóricos. Si son continuos, se discretizan**

Los parámetros a cambiar son:

* Nivel de confianza para la poda del árbol generado (confidence level): influye en el tamaño y la capacidad de predicción del árbol.
* Tamaño del árbol: especificar el mínimo número de instancias por nodo.



Otra opción es generar un conjunto de reglas asociadas al árbol y utilizar un sistema basado en reglas.

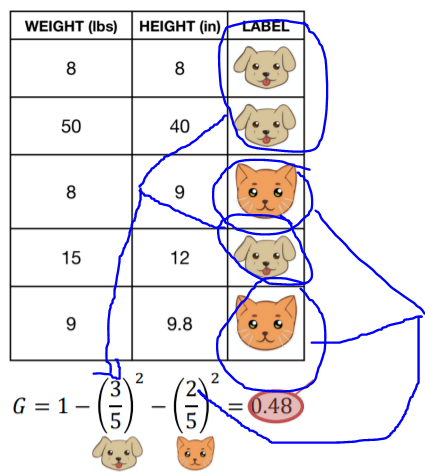
### Construcción de los árboles de decisión

Para construir árboles de decisión, nos interesa crear conjuntos con la menor diversidad posible. Esto es para que los grupos se encuentren bien definidos.

Tenemos una medida para medir la desigualdad, que es “*Gini”,* que se encuentra entre 0 y 1.

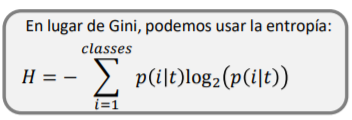
0 corresponde con la perfecta igualdad y 1 con la desigualdad. Con lo cual, siempre intentaremos acercanos a 0. Gini nos informa de la capacidad predictiva de una variable.

En la ecuación, “i” representa el número de clases que han salido de un tipo y “t” significa la sumatoria de todos.



Usaremos también la fórmula para determinar la ganancia de información (la capacidad predictiva de una variable).

En lugar de usar Gini, podemos usar la entropía, que nos marca el nivel de dispersión:



Quiere decir que el nivel máximo es el 0.5 (hay las mismas probabilidades de uno que de otro). Cuanto más distante esté del 0,5, mejor.

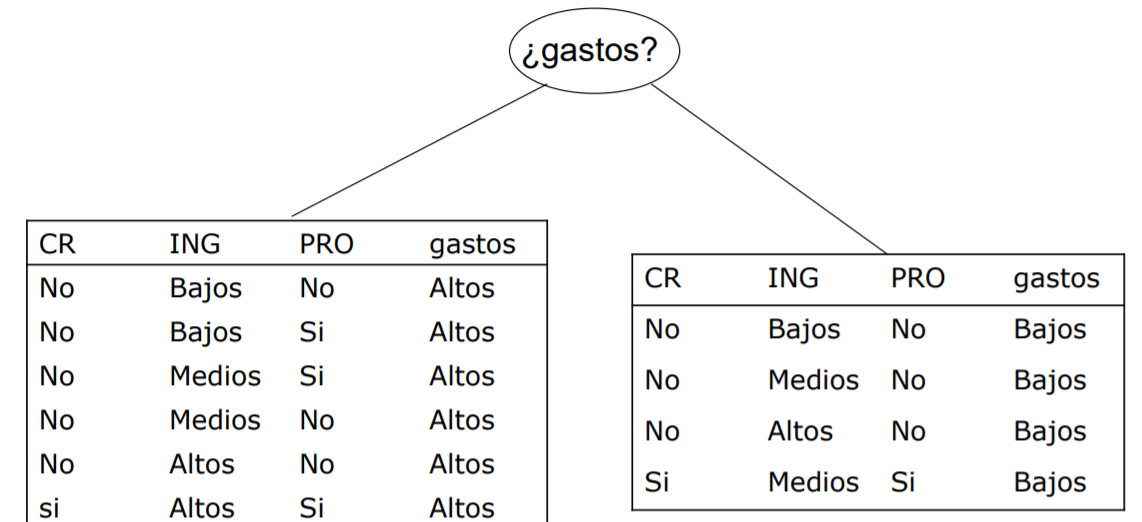
### Construcción de árbol de decisión

#### Fase 1

Cada atributo será un nodo. Por lo tanto, pueden crearse numerosos árboles de decisión. Se intenta podar el árbol para eliminar ruido. Tenemos que elegir el árbol que con menos nodos funcione mejor.

Inicialmente, todos los ejemplos del dataset de entrenamiento se encuentran en el nodo raíz. A continuación, los ejemplos se dividen a lo largo del árbol.

Los atributos de test se seleccionan en base a medida heurística o estadística, como ganancia de información. Esto quiere decir que elegiremos el nodo que más información nos dé (cuanto menor sea Gini, más adecuado será ese atributo para el nodo raíz).



En este caso, el atributo “*gastos”* es el atributo que menor índice de Gini tiene, ya que separa de la mejor manera posible las clases.

#### Fase 2

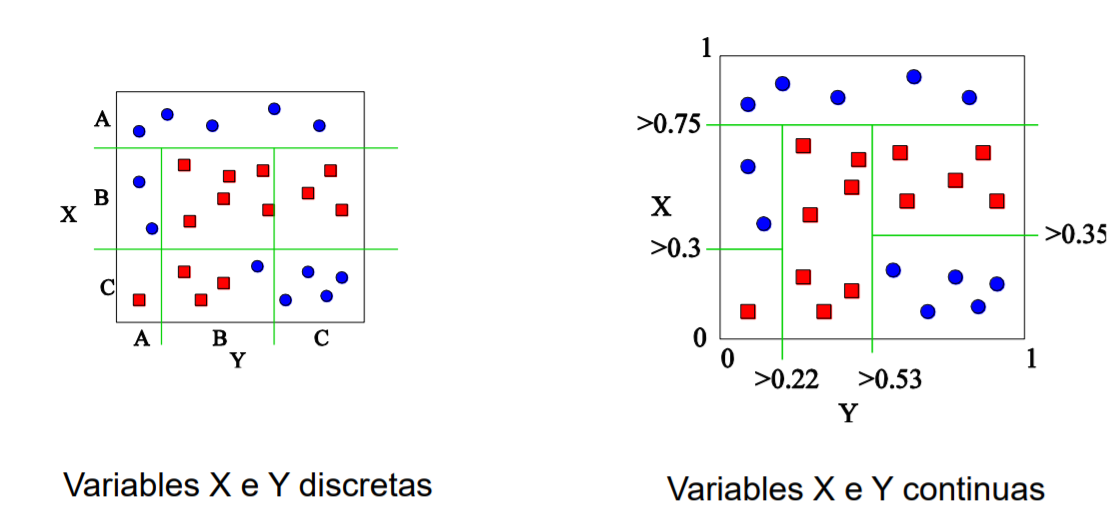
Se prueba el árbol de decisión para clasificar un elemento desconocido.

Sin embargo, conforme se va construyendo el árbol tenemos que seleccionar cual es el mejor atributo para colocar como siguiente nodo. Para ello podemos seguir un criterio para seleccionar estas variables.

### Criterios de selección de variables

Como hemos indicado antes, buscaremos el nodo (o atributo) con mayor ganancia de información. La **entropía** mide la aleatoriedad, sorpresa o incertidumbre al predecir una clase.

### Particionamiento del espacio con árbol de decisión.



Podemos observar cómo el árbol de decisión delimita muy bien el espacio.

### Ventajas e inconvenientes

**Ventajas**

* Son fáciles de utilizar y eficientes
* Reglas fáciles de interpretar
* Escalan bien
* Tratan bien datos con ruido
* No se ven muy influenciados por outliers

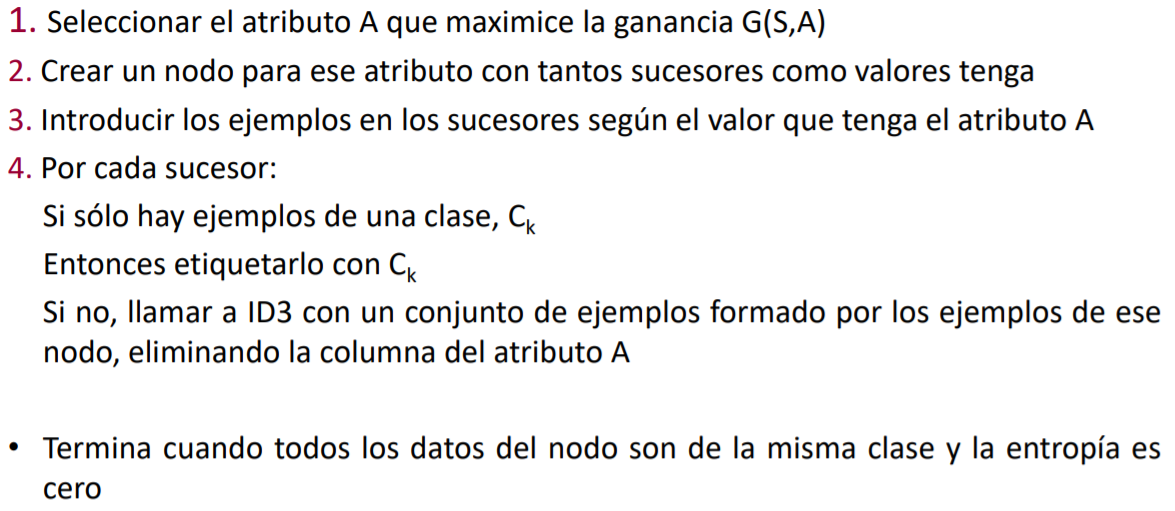
**Inconvenientes**

* No manejan de forma fácil los atributos continuos.
* Dificultad para tratar con valores perdidos.
* Problemas de sobreajuste. Solucionable con podas.
* No detectan correlaciones entre atributos.
* Son sensibles a desbalances.

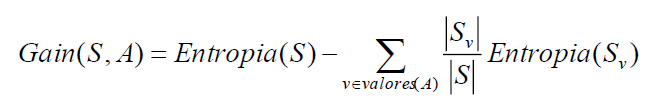
### Algunos algoritmos basados en árboles de decisión

#### Algoritmo ID3

Crea el árbol utilizando conceptos de teoría de información. Utiliza la **entropía: es la incertidumbre.** Siempre se buscará el atributo con máxima ganancia de información. Si lanzamos una moneda, la probabilidad de que salga una cara es de 0.5 -> máxima entropía. Siempre buscaremos la **menor entropía,** ya que queremos separar de la mejor manera posible los conjuntos. El esquema del algoritmo es el siguiente:



Para calcular la ganancia de información, se determina la diferencia entre la entropía del conjunto de datos de partida y la suma ponderada de las entropías una vez dividido el conjunto de ejemplos.



No puede determinar todos los árboles posibles y no puede proponer ejemplos que reduzcan el espacio de búsqueda. Por la ganancia de información, tiene a elegir atributos con muchos valores. Siempre encuentra un óptimo local pero puede no ser el global. Permite ruido en los datos de entrenamiento.

Se prefieren árboles cortos frente a largos.

#### Algoritmo C4.5

Mejora a ID3 en tratar los datos perdidos (se ignoran para construir y se predicen en base a los demás atributos para clasificar) y en tratar datos continuos (se dividen en rangos). Además, propone soluciones para el sobreaprendizaje con la pre-poda (cuándo dejar de sub-dividir el árbol) y post-poda (se construye y luego de poda).

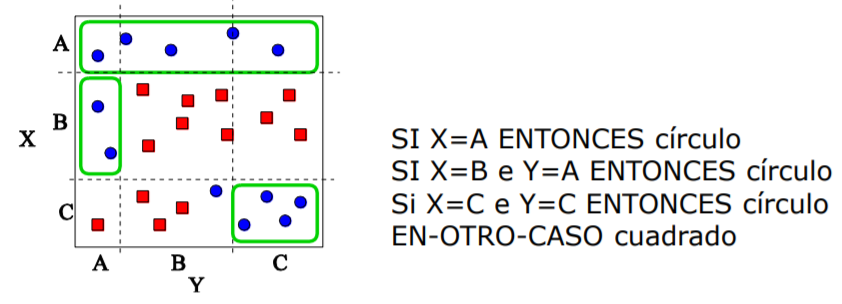
## Clasificadores basados en reglas

Estos van creando reglas, al igual que con los árboles a diferencia que no tiene un orden. Además, en un árbol se comprueban todas las clases. Sin embargo, cuando se genera una regla solo se examina una clase.

Estos clasificadores pueden crear reglas con árboles de decisiones y generar reglas mediante cobertura

### Generación de reglas mediante árboles de decisión

Se pueden extraer las reglas de un árbol de decisión y usarlas como un SBR. En estos casos, no se realiza un particionado exhaustivo del espacio.



**Las reglas estarán ordenadas y siempre se intenta buscar las reglas que cubran el mayor número de ejemplos positivos** (o el menor número de ejemplos negativos).

### Generación de reglas mediante cobertura

Los algoritmos de cobertura **generan reglas para cubrir exactamente una clase**. A diferencia de los algoritmos de generación de árboles (seleccionan el mejor atributo), suelen elegir el mejor par (atributo, valor).



Se intenta reemplazar ¿?? Por el predicado que obtenga la mejor probabilidad de que la clase sea “alto”.

Dos ejemplos de algoritmos de cobertura son el 1R (genera una regla por cada atributo y selecciona la mejor de todas ellas que menos errores ocasione) y PRISM (que genera reglas para cada clase mirando en el conjunto de entrenamiento para que describan todos los ejemplos de dicha clase). Sin embargo, PRISM no garantiza obtener el conjunto óptimo de reglas. A no ser que haya inconsistencias, PRIMS **sobreajusta**. Con los atributos númericos, utilizan < y >. **Supone que no hay ruido.**

# Clasificación con otras técnicas

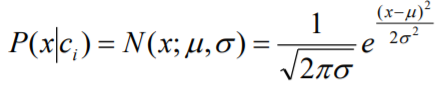
## Clasificadores basados en métodos bayesianos

Representan la incertidumbre, no una clase concreta. Representa la probabilidad de la clase. Entonces, para clasificar un nuevo ejemplo tenemos que identificar el valor de la clase más probable (si o no, por ejemplo) y devolverlo como clasificación.

### Clasificador Naive Bayes

Supone que todos los atributos son independientes, conociendo la variable clase. Aun siendo así, los resultados son competitivos. Según los tipos de variables:

* **Variables discretas:** se estima la probabilidad de que X se encuentre en la clase C como la frecuencia relativa de otros que teniendo el valor de X se encuentra en C. Se estima por máxima verisimilitud (EMV) que es la sumatoria de los datos entre los que son ciertos. Sin embargo, si no se encuentra ninguna vez sería 0. Por lo que se suaviza con LaPlace. Cuando hay muchos casos, tiene a la EMV. Si hay pocos, tiende a la uniforme.
* **Variables numéricas:** se estima mediante una función de densidad gaussiana. Es decir:para cada categoría de la variable se estima una distribución normal.



En este clasificador, intentaremos discretizar lo mejor posible para obtener un mejor rendimiento.

Ventajas

* Es fácil de implementar
* Obtiene buenos resultados

Desventajas

* Origina una falta de precisión al asumir que las variables son independientes entre sí.
* Aumenta coste computacional cuantos más atributos hay.

## Clasificadores basados en instancias

Se basan en el **aprendizaje por analogías.** En estos clasificadores, las BD son los propios modelos. Se trabaja cuando llega un nuevo caso a clasificar. Un ejemplo es el KNN

### Algoritmo KNN

Este algoritmo coge los K vecinos más cercanos, y el que más veces aparezca será el elegido.

Las variables numéricas se normalizan entre 0 y 1 (para evitar problemas en mucha variación entre dos distancias)

Hay que hacer una **selección previa de características** para evitar coger distancias entre **variables irrelevantes**.

En Knime está disponible en WEKA con el nombre IBk

IMPORTANTE NORMALIZAR.

**Ventajas**

* Trata bien los valores perdidos (se toma la máxima distancia, que es 1)
* Trata bien el ruido (cuando hay muchos K, no hay mucha variación)

**Desventajas**

* Ineficiente, ya que hay que cargar la DB en memoria
* Considera todos los atributos con la misma importancia. Hay veces que no la tienen.
* Si no se escogen previamente características, la distancia puede estar dominada por variables irrelevantes.

## Clasificadores basados en redes neuronales

**Ventajas**

* Tienen una gran tasa de acierto
* Son más robustos que los árboles por los pesos
* Son robustos frente a errores (ruido,etc)
* Gran capacidad de salida
* Son rápidos para evaluar nuevos casos
* Mejoran su rendimiento con el post-aprendizaje

**Desventajas**

* Necesitan mucho tiempo para entrenar, ya que gran parte es ensayo y error
* No hay interpretabilidad del modelo
* Los atributos de entrada deben ser numéricos
* Pueden tener problemas de sobreaprendizaje

**¿Cuándo utilizarlas?**

* La entrada tiene una dimensión muy alta
* Posibilidad de datos con ruido
* La salida es un valor discreto o real
* La forma de la función objetivo es desconocida
* La interpretabilidad de los datos no es importante

Sin embargo, para resolver un problema de clasificación implica unas ciertas cosas:

1. Determinar varios datos con conocimiento experto
2. Determinar pesos y funciones a utilizar

Entre otras

Entradas: por cada variable numérica o binaria/booleana se pone una neurona de entrada. Por cada variable nominal (co más de dos estados) se pone una neurona de entrada por cada estado posible.

Salidas: si es predicción 🡪 una neurona de salida por cada valor de la clase.

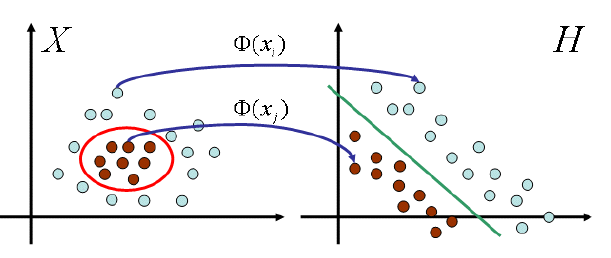
Capas ocultas: 0 🡪 no se pone capa oculta

Numeros enteros separados por comas: cada número representa la cantidad de neuronas de esa capa. P.e: 3,4,5 representa una red con 3 capas ocultas de 3,4 y 5 neuronas respectivamente

**Todas las variables numéricas se normalizan [-1,1].**

## Clasificadores basados en máquinas de soporte vectorial

Una **SVM (**support vector machine) es un modelo de aprendizaje que se fundamenta en la *Teoría de Aprendizaje Estadístico*. Se requiere que los datos sean linealmente separables.



Se pueden usar tanto para regresión como para clasificación.

Se trata de encontrar una línea de separación en la que se maximice el margen entre los puntos de 2 clases distintas.

Sin embargo, aunque haya ruido debemos buscar una buena generalización. Estos datos (los ruidosos) por lo tanto los clasificaremos “mal”. Para controlar esto, podemos usar el hiper-parámetro C

# Multiclasificadores

## Combinación de clasificadores

La idea es pasar la salida de un clasificador a otro clasificador. Se puede aplicar al mismo clasificador o a otro.

## Bagging

Funciona bien para aquellos algoritmos de aprendizaje “inestables”, que son aquellos en los que un pequeño cambio en el conjunto de entrenamiento puede provocar grandes cambios en la predicción (redes neuronales, árboles de decisión, etc). Se puede usar con valores continuos pero la salida se promedia. Se reduce el valor esperado del error. Se puede mejorar si se quita la opción de podado en los árboles, que los hace más inestables.

## Boosting

En lugar de hacer un muestreo aleatorio de los datos de entrenamiento, se ponderan las muestras para concentrar el aprendizaje en los ejemplos más difíciles. Los ejemplos más cercanos a la frontera son los más difíciles de clasificar y recibirán pesos más altos.

Los modelos se construyen influenciados por el anterior. La idea es prestar mayor atención a los ejemplos mal clasificados.

## Descomposición de problemas de multiclase.

Se utiizan estrategias de descomposición “*One-vs-One”* (OVO)

Tema 5. Preprocesamiento de datos

El preprocesado suele llevarse un 80% del trabajo total. El preprocesamiento de datos incluye la preparación de los datos, compuesta por:

* Integración
* Limpieza
* Normalización
* Transformación de los datos

Y las tareas de reducción de datos, compuesta por:

* Selección de características
* Selección de instancias
* Discretización

# Integración, limpieza y transformación

## Integración de datos

Se incorporan y se ajustan los datos. Se sacan de distintas fuentes y se unifican. Al integrar datos, debemos preguntarnos sobre la integración del esquema, la detección de datos duplicados e inconsistencias y la redundancia. Una forma de detectar redundancia es mediante análisis de correlaciones (mediar la fuerza con la que un atributo implica a otro).

## Limpieza de datos

El objetivo es eliminar/reparar datos que provoquen problemas.

## Transformación de los datos

Algunas operaciones típicas son:

* Agregación
* Generalización de los datos (sacar datos a partir de otros)
* Transformación de variables nominales a otras
* Normalización: pasar atributos a un rango mejor.

Ejemplos de técnicas de normalización puede ser:

### Min-Max

Realiza la transformación lineal de los datos originales.

### Zero-mean (Z-score)

Se normaliza en función de la media y la desviación estándar. Útil cuando se desconocen los límites o cuando los datos anómalos pueden dominar la normalización min-max.

### Por escala decimal

Normaliza moviendo el punto decimal de los valores. El nº de puntos movidos depende del valor absoluto máximo de A.

# Datos imperfectos

Vamos a tratar de solucionar los datos imperfectos.

## Valores perdidos

Se pueden utilizar múltiples opciones:

* Ignorar la tupla: se usa cuando la variable no tiene valor
* Rellenar manualmente
* Rellenar utilizando media/desviación del resto de tuplas
* Rellenar con el valor más probable: se utilizan técnicas de inferencia
* Rellenar mediante técnicas de aprendizaje automático: es la mejor.

## Tipos de ruido

El ruido son datos que se cuelan donde no deben. Existen 3 técnicas para el filtrado de ruido en una clasificación:

### Ensemble Filter (EF)

Se utilizan varios clasificadores (C4.5,1-NN y LDA) para crear clasificadores en varios subconjuntos del conjunto de entrenamiento. Para cada algoritmo se utiliza una validación cruzada para etiquetar las validaciones como correcto o incorrecto. A continuación se utiliza un esquema de votación. Si un ejemplo está mal clasificado por todos, se elimina (votación por consenso) y se elimina si está mal clasificado por la mitad de los clasificadores (voto por mayoría).

### Cross-Validated Committees Filter (CVCF)

Es similar al anterior pero con dos diferencias:

* Solo se usa el C4.5 como algoritmo, ya que los árboles funcionan bien como filtros de ruido
* Cada clasificador construido se utiliza para etiquetar **todos los ejemplos de entrenamiento (no solo el conjunto de validación).**

### Iterative Pertitioning Filter (IPF)

Se eliminan los datos ruidosos aplicando CVCF hasta un criterio de parada. Se detiene si durante un número de iteraciones consecutivas, el número de ejemplos ruidosos es inferior a un porcentaje del tamaño del conjunto de datos de entrenamiento. No lo elimina por completo.

Sin embargo, no es lo mismo el ruido que un dato anómalo. Para eliminar el error, se utilizan técnicas de suavizado:

* Binning: se suavizan valores consultando sus vecinos.
* Regresión: los datos se suavizan ajustándolos a una función con técnicas de regresión.

## Outliers

Son datos que tienen unas mediciones “especiales”. Son correctos aunque sean anómalos estadísticamente hablando.

Para tratar los valores anómalos:

* Ignorar
* Filtrar
* Reemplazar el valor
* Discretizar: si transformamos un valor continuo en discreto (muy alto, alto, etc) siempre caerán en las esquinas)

# Reducción de datos

Para la reducción de datos, tenemos 3 vías:

## Selección de características

Se trata en encontrar un subconjunto de variables que optimice el clasificador. Se puede ver como un problema de búsqueda, pero cuantas más variables hay más difícil es. Es exponencial. Existe una función objetivo que evalua la bondad del subconjunto seleccionado. Se distinguen dos enfoques:

* Filtro: se evalúa los subconjuntos basándose en la información que contienen. Se utilizan como función objetivo separabilidad de clases, dependencias estadísticas,etc. Las medidas filtro son las siguientes:

Medidas de separabilidad: miden la separabilidad entre clases.

Correlaciones: Serán buenos subconjuntos los que estén muy correlacionados.

Medidas basadas en teoría de información

Medidas de consistencia: las 3 anteriores buscan encontrar las características que puedan predecir una clase frente al resto. Las de consistencia tratan de encontrar el mínimo número de características que puedan separar las clases de la misma forma que la original.

* Envolvente: aprender distintos algoritmos con los subconjuntos. El valor devuelto suele ser el porcentaje de acuerdo del clasificador construido. Son más eficaces.

**Ventajas**

Envolventes:

* Exactitud: son más exactos que los filtros
* Evitan el sobreajuste

Filtro:

* Rápidos
* Generalidad: datos pueden ser utilizados por cualquier clasificador

**Desventajas**

Envolventes:

* Costoso. Hay que aprender un modelo y validarlo cada vez.
* Pérdida de generalidad: la solución está sesgada hacia el clasificador utilizado

Filtros:

* Tendencia a incluir muchas variables. Se deberá seleccionar un umbral

Existen distintas clasificaciones:

Según la evaluación: filter y envolvente

Disponibilidad de clase: Supervisados y no supervisados

Según la búsqueda: completa, heurística y aleatoria

Según la salida del algoritmo: Ranking (devuelven lista de atributos ordenados según criterio) o subconjunto de atributos (devuelven subconjunto de atributos según criterio de evaluación)

Algunos algoritmos relevantes de selección de caracterisicas son:

* Exhaustivos: Buenos pero alto coste
* Heurísticos: malos pero tardan poco. Encuentran óptimos locales.
* Estocásticos: termino medio.

## Selección de instancias

Se pretende seleccionar los ejemplos más relevantes. Se trata de quedarnos con los puntos suficientes para que la “separación” sea la misma. Existen 3 medios de selección de instancias:

### Muestreo

Se cogen unos prototipos del conjunto de entrenamiento para entrenar al modelo. Dos algoritmos clásicos pueden ser el CNN (Inserta solo las mal clasificadas) y el ENN (se eliminan las instancias clasificadas incorrectamente usando KNN. Suaviza la frontera.).

Los algoritmos de selección de prototipos se enfrentan a dificultades como eficiencia, recursos, generalización y representación. Para solucionar este problema en las grandes BBDD usamos una estrategia de combinación de estratificación con los algoritmos de selección de instancias.

### Desbalanceamiento

Algunos datasets no están balanceados. Se obtienen resultados muy buenos al predecir pero no son útiles. Se pueden balancear eliminando o añadiendo datos. En weka utilizaremos en SMOTE.

# Discretización

Se busca transformar datos continuos/discretos en valores nominales no ordenados. Se trata de colocar todos los valores en distintos intervalos.

Algoritmos supervisados: basados en entropía, métodos Chi-square