Clasificación con reglas

Dra. Amparo López Gaona

Fac. Ciencias, UNAM
Mayo 2018

Tareas de MD

Tareas básicas de minería de datos:

Tareas de MD

Tareas básicas de minería de datos:

- Aprendizaje no-supervisado:

Tareas de MD

Tareas básicas de minería de datos:

- Aprendizaje no-supervisado: No hay etiqueta en los datos y se trata de encontrar patrones comunes, agrupando registros similares.
  - Ejemplo: carritos de compra

Tareas de MD

Tareas básicas de minería de datos:

- Aprendizaje no-supervisado: No hay etiqueta en los datos y se trata de encontrar patrones comunes, agrupando registros similares.
  - Ejemplo: carritos de compra
- Aprendizaje supervisado: Aprende de un conjunto etiquetado de registros.
  - Objetivo:

Tareas de MD

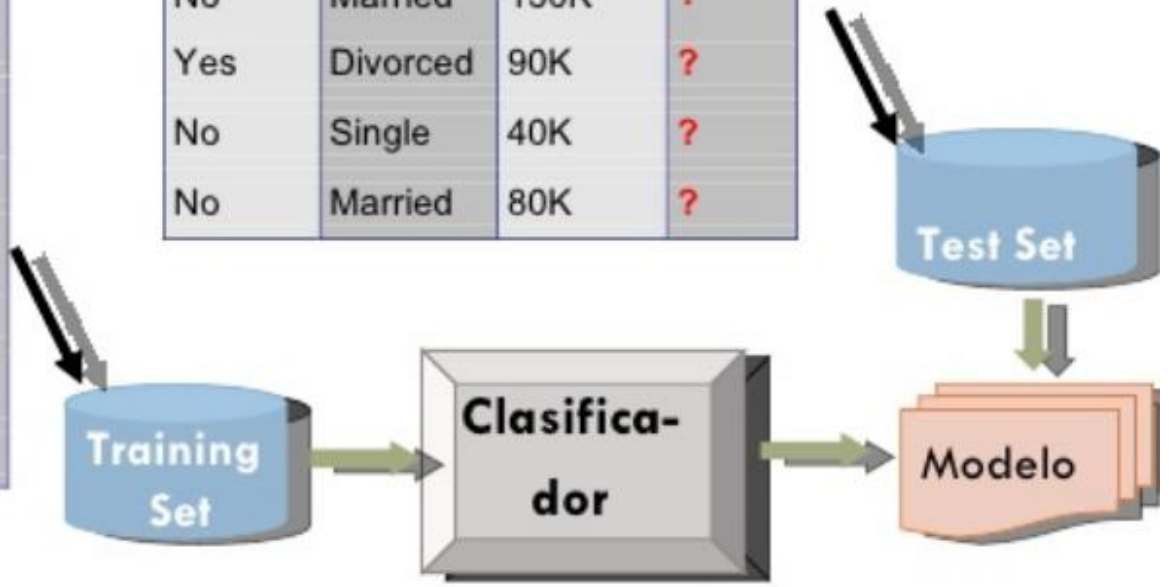
Tareas básicas de minería de datos:

- Aprendizaje no-supervisado: No hay etiqueta en los datos y se trata de encontrar patrones comunes, agrupando registros similares.
  - Ejemplo: carritos de compra
- Aprendizaje supervisado: Aprende de un conjunto etiquetado de registros.
  - Objetivo: Crear un modelo para predecir la clase (clasificar) de nuevos registros.
  - Dada una base de datos $D = \{t_1, t_2, ...t_n\}$ y un conjunto de clases $C = \{C_1, C_2, ...C_m\}$ el problema de clasificación consiste en encontrar una función $f : D \rightarrow C$, para asignar (predecir) una clase C_j, a cada t_i; con la mayor precisión posible.
  - La base de datos se divide en un conjunto de entrenamiento, con el cual se construye el modelo o función; y un conjunto de prueba utilizado para determinar la precisión del modelo.

... Clasificación

Tid	Refund	Marital Status	Taxable Income	Cheat
	categoria	categoria	Continuo	clase
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?



Evaluación del modelo

- Una **matriz de confusión** contiene información acerca de las predicciones hechas por un modelo de clasificación, comparando las realizadas en los registros de aprendizaje contra la clase a la que realmente pertenecen.
- Ejemplo:

		Predicción	
		Mal Pagador	Buen Pagador
Valor Real	Mal Pagador	800	200
	Buen Pagador	500	1500

Evaluación del modelo

- Una **matriz de confusión** contiene información acerca de las predicciones hechas por un modelo de clasificación, comparando las realizadas en los registros de aprendizaje contra la clase a la que realmente pertenecen.
- Ejemplo:

		Predicción	
		Mal Pagador	Buen Pagador
Valor Real	Mal Pagador	800	200
	Buen Pagador	500	1500

- De "Mal pagador" se predijo correctamente el 80 %.
- De "Buen pagador" se predijo correctamente el 75 %.

Evaluación del modelo

- Una **matriz de confusión** contiene información acerca de las predicciones hechas por un modelo de clasificación, comparando las realizadas en los registros de aprendizaje contra la clase a la que realmente pertenecen.
- Ejemplo:

		Predicción	
		Mal Pagador	Buen Pagador
Valor Real	Mal Pagador	800	200
	Buen Pagador	500	1500

- De "Mal pagador" se predijo correctamente el 80 %.
- De "Buen pagador" se predijo correctamente el 75 %.
- Se acertó en 2300 de 3000 predicciones para una efectividad de 76.6 %

... Evaluación del modelo

		Predicción	
		Negativo	Positivo
Valor Real	Negativo	a	b
	Positivo	c	d

- La **precisión** de un modelo de predicción es la proporción de predicciones acertadas con respecto a la cantidad total de predicciones.

$$Precision = \frac{a+d}{a+b+c+d}$$

- El **error** = 1 - Precisión.

$$Error = \frac{b+c}{a+b+c+d}$$

... Evaluación del modelo

- La precisión puede ser engañosa.

		Predicción	
		Fraude	No Fraude
Valor Real	Fraude	0	8
	No Fraude	3	989

- La precisión es del

... Evaluación del modelo

- La precisión puede ser engañosa.

		Predicción	
		Fraude	No Fraude
Valor Real	Fraude	0	8
	No Fraude	3	989

- La precisión es del 98.9 %,

... Evaluación del modelo

- La precisión puede ser engañosa.

		Predicción	
		Fraude	No Fraude
Valor Real	Fraude	0	8
	No Fraude	3	989

- La precisión es del 98.9 %, sin embargo, el modelo no detectó ningún fraude.

... Evaluación del modelo

		Predicción	
		Negativo	Positivo
Valor Real	Negativo	a	b
	Positivo	c	d

- La precisión positiva (PP) es la proporción de casos positivos acertados y se calcula como $PP = d/(c + d)$.
- La precisión negativa (PN) es la proporción de casos negativos acertados y se calcula como $PN = a/(a + b)$.
- Falsos positivos es la proporción de casos negativos clasificados como positivos sin serlo: $FP = b/(a + b)$.
- Falsos negativos es la proporción de casos positivos clasificados como negativos sin serlo: $FN = c/(c + d)$.

... Evaluación del modelo

- La precisión puede ser engañosa.

		Predicción	
		Fraude	No Fraude
Valor Real	Fraude	0	8
	No Fraude	3	989

- La precisión es del 98.9
- PP $= 99.6\,\%$ y PN

... Evaluación del modelo

- La precisión puede ser engañosa.

		Predicción	
		Fraude	No Fraude
Valor Real	Fraude	0	8
	No Fraude	3	989

- La precisión es del 98.9
- PP $= 99.6\,\%$ y PN $= 0\,\%$.

Modelos generados

Modelos generados

- Árbol de decisión.
- Red neuronal.

Modelos generados

- Árbol de decisión.
- Red neuronal.
- Reglas.
- Ninguno.

Modelo de clasificación basada en reglas

- Representan el conocimiento como un conjunto de reglas de la forma `if` *condición* `then` *y*, o bien, (*condicion*) $\rightarrow$ *y*
  - El antecedente (condición) está formado por la combinación de valores de algunos/ todos atributos independientes.
  - En el consecuente (conclusión) se especifica la clase a la que pertenece la tupla a clasificar, en caso de que el consecuente sea verdadero.
- Ejemplos:
  - `IF` edad = joven `AND` estudia= si `THEN` compra = si
  - (edad = joven) $\wedge$ (estudia= si) $\rightarrow$ compra = si
  - (ingreso $<$ 50K) $\wedge$ (reembolso = si) $\rightarrow$ evasor = no

... Modelo de clasificación basada en reglas

- El resultado de estos modelos de clasificación es más directo y fácil de entender que los árboles de decisión.
- Trabajan de manera iterativa identificando una regla que cubre un subconjunto de tuplas del conjunto de entrenamiento y separando estas tuplas del resto. Hasta que no haya más tuplas.

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

- R1: (Da a luz = no) $\wedge$ (vuela = si) $\rightarrow$ Ave
- R2: (Da a luz = no) $\wedge$ (acuático = si) $\rightarrow$ Pez
- R3: (Da a luz = si) $\wedge$ (tipo de sangre = caliente) $\rightarrow$ mamífero
- R5: (Acuático = ocasionalmente) $\rightarrow$ Anfibio

Aplicación de un clasificador basado en reglas

- Se dice que una regla r **cubre** una instancia x, si los atributos de la instancia satisfacen la condición de la regla.

- Con las reglas:
  - R1: (Da a luz = no) $\wedge$ (vuela = si) $\rightarrow$ Ave
  - R2: (Da a luz = no) $\wedge$ (acuático = si) $\rightarrow$ Pez
  - R3: (Da a luz = si) $\wedge$ (tipo de sangre = caliente) $\rightarrow$ mamífero
  - R4: (Da a luz = no) $\wedge$ ((vuela = no) $\rightarrow$ Reptil
  - R5: (Acuático = ocasionalmente) $\rightarrow$ Anfibio
- Clasificar las siguientes tuplas:

nombre	tipodeSangre	dar a luz	volar	acuático	clase
halcón	caliente	no	si	no	?
oso	caliente	si	no	no	?

Aplicación de un clasificador basado en reglas

- Se dice que una regla r **cubre** una instancia x, si los atributos de la instancia satisfacen la condición de la regla.

- Con las reglas:
  - R1: (Da a luz = no) $\wedge$ (vuela = si) $\rightarrow$ Ave
  - R2: (Da a luz = no) $\wedge$ (acuático = si) $\rightarrow$ Pez
  - R3: (Da a luz = si) $\wedge$ (tipo de sangre = caliente) $\rightarrow$ mamífero
  - R4: (Da a luz = no) $\wedge$ ((vuela = no) $\rightarrow$ Reptil
  - R5: (Acuático = ocasionalmente) $\rightarrow$ Anfibio
- Clasificar las siguientes tuplas:

nombre	tipodeSangre	dar a luz	volar	acuático	clase
halcón	caliente	no	si	no	?
oso	caliente	si	no	no	?

- La regla R1 cubre que un halcón $\rightarrow$ ave
- La regla R3 cubre el oso pardo $\rightarrow$ mamífero

Evaluación de una regla

- Cobertura(R) = Porcentaje de registros que satisfacen el antecedente de una regla.
  Porcentaje de registros cubiertos por la regla.
- Precisión(R) = Fracción de registros que satisfacen tanto el antecedente como el consecuente de R.
  Porcentaje de los registros cubiertos, que fueron correctamente clasificados.
- Cálculo de cobertura y precisión para un dataset de entrenamiento D.
  - $n_{cubiertas}$ = cantidad de tuplas que cubiertas por R.
  - $n_{correctas}$ = cantidad de tuplas clasificadas correctamente por R.
  - D = dataset de entrenamiento.
  - $$\text{cobertura(R)} = \frac{n_{cubiertas}}{|D|}$$
  - $$\text{precisión(R)} = \frac{n_{correctas}}{n_{cubiertas}}$$

... Evaluación de una regla (Ejemplo)

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

`(edoCivil = soltero)` $\rightarrow$ `No`

tiene cobertura del

... Evaluación de una regla (Ejemplo)

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

`(edoCivil = soltero)` $\rightarrow$ `No`

tiene cobertura del 40 % y precisión del

... Evaluación de una regla (Ejemplo)

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

`(edoCivil = soltero)` $\rightarrow$ `No`

tiene cobertura del 40 % y precisión del 50 %.

... Evaluación de una regla (Ejemplo)

- Si una tupla x satisface una regla R_i, se dice que la **dispara/activa**.

... Evaluación de una regla (Ejemplo)

- Si una tupla x satisface una regla R_i, se dice que la **dispara/activa**.
- Ejemplo:
  - R1: (Da a luz = no) $\wedge$ (vuela = si) $\rightarrow$ Ave
  - R2: (Da a luz = no) $\wedge$ (acuático = si) $\rightarrow$ Pez
  - R3: (Da a luz = si) $\wedge$ (tipo de sangre = caliente) $\rightarrow$ Mamífero
  - R4: (Da a luz = no) $\wedge$ ((volar = no) $\rightarrow$ Reptil
  - R5: (Acuático = ocasionalmente) $\rightarrow$ Anfibio

nombre	tipodeSangre	dar a luz	volar	acuático	clase
lemur	caliente	sí	n	no	?
tortuga	fría	no	no	ocasionalmente	?
tiburón	fría	sí	no	sí	?

- Un lémur activa la regla R3, así que es clasificado como mamífero.
- Una tortuga activa

... Evaluación de una regla (Ejemplo)

- Si una tupla x satisface una regla R_i, se dice que la **dispara/activa**.
- Ejemplo:
  - R1: (Da a luz = no) $\wedge$ (vuela = si) $\rightarrow$ Ave
  - R2: (Da a luz = no) $\wedge$ (acuático = si) $\rightarrow$ Pez
  - R3: (Da a luz = si) $\wedge$ (tipo de sangre = caliente) $\rightarrow$ Mamífero
  - R4: (Da a luz = no) $\wedge$ ((volar = no) $\rightarrow$ Reptil
  - R5: (Acuático = ocasionalmente) $\rightarrow$ Anfibio

nombre	tipodeSangre	dar a luz	volar	acuático	clase
lemur	caliente	sí	n	no	?
tortuga	fría	no	no	ocasionalmente	?
tiburón	fría	sí	no	sí	?

- Un lémur activa la regla R3, así que es clasificado como mamífero.
- Una tortuga activa las reglas R4 y R5.
- Un tiburón

... Evaluación de una regla (Ejemplo)

- Si una tupla x satisface una regla R_i, se dice que la **dispara/activa**.
- Ejemplo:
  - R1: (Da a luz = no) $\wedge$ (vuela = si) $\rightarrow$ Ave
  - R2: (Da a luz = no) $\wedge$ (acuático = si) $\rightarrow$ Pez
  - R3: (Da a luz = si) $\wedge$ (tipo de sangre = caliente) $\rightarrow$ Mamífero
  - R4: (Da a luz = no) $\wedge$ ((volar = no) $\rightarrow$ Reptil
  - R5: (Acuático = ocasionalmente) $\rightarrow$ Anfibio

nombre	tipodeSangre	dar a luz	volar	acuático	clase
lemur	caliente	sí	n	no	?
tortuga	fría	no	no	ocasionalmente	?
tiburón	fría	sí	no	sí	?

- Un lémur activa la regla R3, así que es clasificado como mamífero.
- Una tortuga activa las reglas R4 y R5.
- Un tiburón no activa ninguna regla.

... Evaluación de una regla (Ejemplo)

- Si sólo se activa una regla, ésta devuelve la clase para la tupla X.
- Si se activa más de una regla, es necesario resolver los conflictos.

... Evaluación de una regla (Ejemplo)

- Si sólo se activa una regla, ésta devuelve la clase para la tupla X.
- Si se activa más de una regla, es necesario resolver los conflictos.
  - Orden del tamaño: asigna la mayor prioridad a las reglas activadas que tienen la mayor cantidad de atributos probados.
  - Orden basado en la regla (lista de decisión): las reglas se organizan en una lista de prioridad de acuerdo a algunas medidas de la calidad de los reglas dada por los expertos.
  - Orden basado en la clase: las reglas que pertenecen a la misma clase aparecen juntas y en orden decreciente de acuerdo a alguna medida de la calidad de las reglas.

... Evaluación de una regla (Ejemplo)

- Si sólo se activa una regla, ésta devuelve la clase para la tupla X.
- Si se activa más de una regla, es necesario resolver los conflictos.
  - Orden del tamaño: asigna la mayor prioridad a las reglas activadas que tienen la mayor cantidad de atributos probados.
  - Orden basado en la regla (lista de decisión): las reglas se organizan en una lista de prioridad de acuerdo a algunas medidas de la calidad de los reglas dada por los expertos.
  - Orden basado en la clase: las reglas que pertenecen a la misma clase aparecen juntas y en orden decreciente de acuerdo a alguna medida de la calidad de las reglas.
- Si ninguna regla se activa,

... Evaluación de una regla (Ejemplo)

- Si sólo se activa una regla, ésta devuelve la clase para la tupla X.
- Si se activa más de una regla, es necesario resolver los conflictos.
  - Orden del tamaño: asigna la mayor prioridad a las reglas activadas que tienen la mayor cantidad de atributos probados.
  - Orden basado en la regla (lista de decisión): las reglas se organizan en una lista de prioridad de acuerdo a algunas medidas de la calidad de los reglas dada por los expertos.
  - Orden basado en la clase: las reglas que pertenecen a la misma clase aparecen juntas y en orden decreciente de acuerdo a alguna medida de la calidad de las reglas.
- Si ninguna regla se activa, se tiene una regla por omisión basada en el conjunto de entrenamiento.

Métodos para generar reglas de clasificación

- Método directo:
  - Las reglas se extraen directamente de los datos de entrenamiento.
  - Ejemplos: OneR, RIPPER, CN2, AQ.
- Método indirecto:
  - Las reglas se obtienen de otros modelos de clasificación (árboles de decisión, redes neuronales, etc.)
  - Ejemplo: reglas C4.5

Método directo: Algoritmo OneR

Método directo: Algoritmo OneR

- Muchos modelos son tan complejos que no pueden entenderse.
- Objetivo: Buscar el mejor modelo sencillo que logre un balance entre la mejor precisión posible y un modelo suficientemente sencillo para entenderlo.
- Es un algoritmo de clasificación sencillo y preciso que genera una regla para cada predictor en los datos, luego selecciona la regla con el menor error total como su "one rule".
- Algoritmo.
  Para cada atributo A:
  - Para cada valor V de ese atributo, crear una regla como sigue:
    1. Contar la frecuencia de cada valor del objetivo (clase).
    2. Encontrar la clase más frecuente c.
    3. Hacer la regla `if A = V then C = c`
  - Calcular el error total de esta regla.

  Elegir el predictor con el menor error total.

Ejemplo: ¿jugar o no jugar?

Encontrar el mejor predictor con el menor error total usando OneR.

Pronóstico	Temperatura	Humedad	Viento	Jugar
Lluvioso	Caliente	Alta	Falso	No
Lluvioso	Caliente	Alta	Verdadero	No
Nublado	Caliente	Alta	Falso	Sí
Soleado	Templado	Alta	Falso	Sí
Soleado	Fresco	Normal	Falso	Sí
Soleado	Fresco	Normal	Verdadero	No
Nublado	Fresco	Normal	Verdadero	Sí
Lluvioso	Templado	Alta	Falso	No
Lluvioso	Fresco	Normal	Falso	Sí
Soleado	Templado	Normal	Falso	Sí
Lluvioso	Templado	Normal	Verdadero	Sí
Nublado	Templado	Alta	Verdadero	Sí
Nublado	Caliente	Normal	Falso	Sí
Soleado	Templado	Alta	Verdadero	No

... Ejemplo

Atributo	Reglas	Errores	Total de errores
Pronóstico	Soleado $->$ no	2/5	4/14
	Nublado $->$ sí	0/4	
	Lluvioso $->$ sí	2/5	
Temperatura	Caliente $->$ no	2/4	5/14
	Templado $->$ sí	2/6	
	Fresco $->$ sí	1/4	
Humedad	Alta $->$ no	3/7	4/14
	Normal $->$ sí	1/7	
Viento	Falso $->$ sí	2/8	5/14
	Verdadero $->$ no	3/6	

... Ejemplo

Atributo	Reglas	Errores	Total de errores
Pronóstico	Soleado $->$ no	2/5	4/14
	Nublado $->$ sí	0/4	
	Lluvioso $->$ sí	2/5	
Temperatura	Caliente $->$ no	2/4	5/14
	Templado $->$ sí	2/6	
	Fresco $->$ sí	1/4	
Humedad	Alta $->$ no	3/7	4/14
	Normal $->$ sí	1/7	
Viento	Falso $->$ sí	2/8	5/14
	Verdadero $->$ no	3/6	

- Por lo tanto el mejor predictor es

... Ejemplo

Atributo	Reglas	Errores	Total de errores
Pronóstico	Soleado $->$ no	2/5	4/14
	Nublado $->$ sí	0/4	
	Lluvioso $->$ sí	2/5	
Temperatura	Caliente $->$ no	2/4	5/14
	Templado $->$ sí	2/6	
	Fresco $->$ sí	1/4	
Humedad	Alta $->$ no	3/7	4/14
	Normal $->$ sí	1/7	
Viento	Falso $->$ sí	2/8	5/14
	Verdadero $->$ no	3/6	

- Por lo tanto el mejor predictor es **Pronóstico**.
- Las reglas son:
  - `Si pronóstico = Soleado entonces jugar = si`
  - `Si pronóstico = Nublado entonces jugar = si`
  - `Si pronóstico = Lluvioso entonces jugar = no`

... Evaluación del modelo

Matriz de confusión:

	Jugar	No jugar	
Jugar	7	2	Precisión positiva 0.78
No jugar	2	3	Precisión negativa 0.60

- Precisión del modelo = 0.71
- Error: 0.29

Trabajo con atributos numéricos

- Discretizar los atributos numéricos.
- Dividir cada rango de atributos en intervalos:
  - Ordenar las instancias de acuerdo a los valores de los atributos.
  - Colocar puntos de ruptura donde las clases cambian.
  - De esa forma se minimiza el error total.
- Ejemplo:

... Trabajo con atributos numéricos

Pronóstico	Temperatura	Humedad	Viento	Jugar
Lluvioso	29	85	Falso	No
Lluvioso	26	90	Verdadero	No
Nublado	28	86	Falso	Sí
Soleado	21	96	Falso	Sí
Soleado	20	80	Falso	Sí
Soleado	18	70	Verdadero	No
Nublado	17	65	Verdadero	Sí
Lluvioso	22	95	Falso	No
Lluvioso	20	70	Falso	Sí
Soleado	24	80	Falso	Sí
Lluvioso	24	70	Verdadero	Sí
Nublado	22	90	Verdadero	Sí
Nublado	27	75	Falso	Sí
Soleado	21	91	Verdadero	No

... Trabajo con atributos numéricos

- Temperatura:

17	18	20	20	21	21	22	22	24	24	26	27	28	29
sí	no	sí	sí	sí	no	no	sí	sí	sí	no	sí	si	no

- Discretización implica dividir esta secuencia colocando puntos de ruptura en donde la clase cambia.

17	18	20	20	21	21	22	22	24	24	26	27	28	29
sí	no	sí	sí	sí	no	no	sí	sí	sí	no	sí	si	no

... Trabajo con atributos numéricos (Sobreajuste)

- El problema de sobre-ajuste se produce cuando un atributo tiene varios posibles valores.
- Este procedimiento es muy sensible al ruido.
  - Una instancia con una etiqueta de clase incorrecta probablemente producirá un intervalo separado.
- Solución sencilla: forzar el número mínimo de instancias en la clase mayoritaria para cada intervalo.

... Trabajo con atributos numéricos

Particionamiento con mínimo conjunto de 3 temperaturas.

- Datos originales

17	18	20	20	21	21	22	22	24	24	26	27	28	29
sí	no	sí	sí	sí	no	no	sí	sí	sí	no	sí	si	no

... Trabajo con atributos numéricos

Particionamiento con mínimo conjunto de 3 temperaturas.

- Datos originales

17	18	20	20	21	21	22	22	24	24	26	27	28	29
sí	no	sí	sí	sí	no	no	sí	sí	sí	no	sí	si	no

- Primera partición:
  - `sí no sí sí | sí ...`
- Como lo que sigue es otro sí se incluye:

17	18	20	20	21	21	22	22	24	24	26	27	28	29
sí	no	sí	sí	sí	no	no	sí	sí	sí	no	sí	si	no

- La discretización final es:

17	18	20	20	21	21	22	22	24	24	26	27	28	29
sí	no	sí	sí	sí	no	no	sí	sí	sí	no	sí	si	no

- EL conjunto de reglas se:
  - temperatura: $\leq 77.5 \rightarrow$ `sí`
  - temperatura: $> 77.5 \rightarrow$ `no`

... Trabajo con atributos numéricos

Atributo	Reglas	Errores	Total de errores
Pronóstico	Soleado $->$ no	2/5	4/14
	Nublado $->$ sí	0/4	
	Lluvioso $->$ sí	2/5	
Temperatura	$\leq 77.5 ->$ sí	3/10	5/14
	$> 77.5 ->$ no	2/4	
Humedad	$\leq 82.5 ->$ sí	1/7	3/14
	$> 82.5 ->$ no	2/7	
Viento	Falso $->$ sí	2/8	5/14
	Verdadero $->$ no	3/6	

... Trabajo con atributos numéricos

Atributo	Reglas	Errores	Total de errores
Pronóstico	Soleado $->$ no	2/5	4/14
	Nublado $->$ sí	0/4	
	Lluvioso $->$ sí	2/5	
Temperatura	$\leq 77.5 ->$ sí	3/10	5/14
	$> 77.5 ->$ no	2/4	
Humedad	$\leq 82.5 ->$ sí	1/7	3/14
	$> 82.5 ->$ no	2/7	
Viento	Falso $->$ sí	2/8	5/14
	Verdadero $->$ no	3/6	

Ejemplo con el dataset breastcancer

```
> install.packages("OneR")
> library(OneR)
> data(breastcancer)
> datos <- breastcancer
> str(datos)

data.frame': 699 obs. of  10 variables:
 $ Clump Thickness             : int  5 5 3 6 4 8 1 2 2 4 ...
 $ Uniformity of Cell Size     : int  1 4 1 8 1 10 1 1 1 2 ...
 $ Uniformity of Cell Shape    : int  1 4 1 8 1 10 1 2 1 1 ...
 $ Marginal Adhesion           : int  1 5 1 1 3 8 1 1 1 1 ...
 $ Single Epithelial Cell Size: int  2 7 2 3 2 7 2 2 2 2 ...
 $ Bare Nuclei                 : int  1 10 2 4 1 10 10 1 1 1 ...
 $ Bland Chromatin             : int  3 3 3 3 3 9 3 3 1 2 ...
 $ Normal Nucleoli             : int  1 2 1 7 1 7 1 1 1 1 ...
 $ Mitoses                     : int  1 1 1 1 1 1 1 1 5 1 ...
 $ Class                       : Factor w/ 2 levels "benign","malign
```

... Ejemplo oneR

```
> str(bin(datos))
'data.frame': 683 obs. of  10 variables:
 $ Clump Thickness            : Factor w/ 5 levels "(0.991,2.8]",..
 $ Uniformity of Cell Size    : Factor w/ 5 levels "(0.991,2.8]",..
 $ Uniformity of Cell Shape   : Factor w/ 5 levels "(0.991,2.8]",..
 $ Marginal Adhesion          : Factor w/ 5 levels "(0.991,2.8]",..
 $ Single Epithelial Cell Size: Factor w/ 5 levels "(0.991,2.8]",..
 $ Bare Nuclei                : Factor w/ 5 levels "(0.991,2.8]",..
 $ Bland Chromatin            : Factor w/ 5 levels "(0.991,2.8]",..
 $ Normal Nucleoli            : Factor w/ 5 levels "(0.991,2.8]",..
 $ Mitoses                    : Factor w/ 5 levels "(0.991,2.8]",..
 $ Class                      : Factor w/ 2 levels "benign","malign
 - attr(*, "na.action")=Class 'omit'  Named int [1:16] 24 41 140 14
   .. ..- attr(*, "names")= chr [1:16] "24" "41" "140" "146" ...
Warning message:
In bin(datos) : 16 instance(s) removed due to missing values
```

... Ejemplo oneR

```
> str(bin(datos, nbins = 3))
'data.frame': 683 obs. of  10 variables:
 $ Clump Thickness            : Factor w/ 3 levels "(0.991,4]","(4,
 $ Uniformity of Cell Size    : Factor w/ 3 levels "(0.991,4]","(4,
 $ Uniformity of Cell Shape   : Factor w/ 3 levels "(0.991,4]","(4,
 $ Marginal Adhesion          : Factor w/ 3 levels "(0.991,4]","(4,
 $ Single Epithelial Cell Size: Factor w/ 3 levels "(0.991,4]","(4,
 $ Bare Nuclei                : Factor w/ 3 levels "(0.991,4]","(4,
 $ Bland Chromatin            : Factor w/ 3 levels "(0.991,4]","(4,
 $ Normal Nucleoli            : Factor w/ 3 levels "(0.991,4]","(4,
 $ Mitoses                    : Factor w/ 3 levels "(0.991,4]","(4,
 $ Class                      : Factor w/ 2 levels "benign","malign
 - attr(*, "na.action")=Class 'omit'  Named int [1:16] 24 41 140 14
   .. ..- attr(*, "names")= chr [1:16] "24" "41" "140" "146" ...
Warning message:
In bin(datos, nbins = 3) : 16 instance(s) removed due to missing va
>
```

... Ejemplo oneR

```
> datos <-optbin(datos)

> str(datos)
'data.frame': 683 obs. of  10 variables:
 $ Clump Thickness            : Factor w/ 2 levels "(0.991,5.49]",..
 $ Uniformity of Cell Size    : Factor w/ 2 levels "(0.991,3.24]",..
 $ Uniformity of Cell Shape   : Factor w/ 2 levels "(0.991,3.51]",..
 $ Marginal Adhesion          : Factor w/ 2 levels "(0.991,3.13]",..
 $ Single Epithelial Cell Size: Factor w/ 2 levels "(0.991,3.44]",..
 $ Bare Nuclei                : Factor w/ 2 levels "(0.991,4.1]",...
 $ Bland Chromatin            : Factor w/ 2 levels "(0.991,3.87]",..
 $ Normal Nucleoli            : Factor w/ 2 levels "(0.991,3.2]",...
 $ Mitoses                    : Factor w/ 2 levels "(0.991,1.82]",..
 $ Class                      : Factor w/ 2 levels "benign","malign
 - attr(*, "na.action")=Class 'omit'  Named int [1:16] 24 41 140 14
```

... Ejemplo con oneR

```
> head(datos)
  Clump Thickness Uniformity of Cell Size Uniformity of Cell Shape
1    (0.991,5.49]              (0.991,3.24]                (0.991,3.51]
2    (0.991,5.49]                 (3.24,10]                   (3.51,10]
3    (0.991,5.49]              (0.991,3.24]                (0.991,3.51]
4       (5.49,10]                 (3.24,10]                   (3.51,10]
5    (0.991,5.49]              (0.991,3.24]                (0.991,3.51]
6       (5.49,10]                 (3.24,10]                   (3.51,10]
  Marginal Adhesion Single Epithelial Cell Size Bare Nuclei Bland Cl
1      (0.991,3.13]                (0.991,3.44] (0.991,4.1]    (0.99
2         (3.13,10]                   (3.44,10]    (4.1,10]    (0.99
3      (0.991,3.13]                (0.991,3.44] (0.991,4.1]    (0.99
4      (0.991,3.13]                (0.991,3.44] (0.991,4.1]    (0.99
5      (0.991,3.13]                (0.991,3.44] (0.991,4.1]    (0.99
6         (3.13,10]                   (3.44,10]    (4.1,10]       (3
  Normal Nucleoli      Mitoses     Class
1     (0.991,3.2] (0.991,1.82]    benign
2     (0.991,3.2] (0.991,1.82]    benign
```

... Ejemplo oneR

Creación del conjunto de entrenamiento (80 %) y el de prueba (20 %)

```
> set.seed(12)
> random <- sample(1:nrow(datos), 0.8 * nrow(datos))
> entrenamiento <- datos[random, ]

> prueba <- datos[-random, ]
```

... Ejemplo oneR

```
> modelo <-OneR(entrenamiento, verbose=TRUE)

    Attribute                    Accuracy
1 * Uniformity of Cell Size      92.31%
2   Uniformity of Cell Shape     91.76%
3   Bare Nuclei                  90.29%
4   Bland Chromatin              90.11%
5   Single Epithelial Cell Size  87%
6   Marginal Adhesion            86.45%
7   Normal Nucleoli              86.08%
8   Clump Thickness              84.8%
9   Mitoses                      77.11%
---
Chosen attribute due to accuracy
and ties method (if applicable): '*'
```

... Ejemplo oneR

```
> summary(modelo)

Call:
OneR.data.frame(x = entrenamiento, verbose = TRUE)

Rules:
If Uniformity of Cell Size = (0.991,3.24] then Class = benign
If Uniformity of Cell Size = (3.24,10]    then Class = malignant

Accuracy:
504 of 546 instances classified correctly (92.31%)

Contingency table:
           Uniformity of Cell Size
Class       (0.991,3.24] (3.24,10] Sum
  benign             * 338         9 347
  malignant             33     * 166 199
  Sum                  371       175 546
```

```
> prediccion <- predict(modelo, prueba)  #Haciendo predicciones con

> eval_model(prediccion, prueba)  #Evaluando las predicciones
Confusion matrix (absolute):
           Actual
Prediction  benign malignant Sum
  benign        95         4  99
  malignant      2        36  38
  Sum           97        40 137

Confusion matrix (relative):
           Actual
Prediction  benign malignant  Sum
  benign      0.69      0.03 0.72
  malignant   0.01      0.26 0.28
  Sum         0.71      0.29 1.00

Accuracy: 0.9562 (131/137)

Error rate: 0.0438 (6/137)
```

Ventajas y desventajas del OneR

Ventajas y desventajas del OneR

Ventajas:

- Genera un conjunto de reglas con un solo discriminante, fáciles de entender por las personas.
- EN general, el rendimiento es sorpresivamente bueno.
- Puede servir de punto de referencia para algoritmos más complejos.

Desventajas:

Ventajas y desventajas del OneR

Ventajas:

- Genera un conjunto de reglas con un solo discriminante, fáciles de entender por las personas.
- EN general, el rendimiento es sorpresivamente bueno.
- Puede servir de punto de referencia para algoritmos más complejos.

Desventajas:

- Usa sólo un atributo.
- Probablemente es demasiado simplista.

RIPPER (Repeated Incremental Pruning to Produce Error Reduction)

- Precisión comparable ($\geq$) con la de los árboles de decisión.
- Usado para clasificar conjuntos con distribución no balanceada de las clases y con datos ruidosos.
- Es un algoritmo de cobertura secuencial:

RIPPER (Repeated Incremental Pruning to Produce Error Reduction)

- Precisión comparable ($\geq$) con la de los árboles de decisión.
- Usado para clasificar conjuntos con distribución no balanceada de las clases y con datos ruidosos.
- Es un algoritmo de cobertura secuencial:
  - Las reglas se aprenden una a la vez de manera secuencial.
  - Cada regla para una clase dada, idealmente, cubre la mayor parte de los registros de esa clase y "ninguno de otras clases".
  - Cada vez que una regla se aprende, los registros cubiertos por la regla se eliminan, y el proceso se repite con las instancias restantes.

RIPPER (Repeated Incremental Pruning to Produce Error Reduction)

- Precisión comparable ($\geq$) con la de los árboles de decisión.
- Usado para clasificar conjuntos con distribución no balanceada de las clases y con datos ruidosos.
- Es un algoritmo de cobertura secuencial:
  - Las reglas se aprenden una a la vez de manera secuencial.
  - Cada regla para una clase dada, idealmente, cubre la mayor parte de los registros de esa clase y "ninguno de otras clases".
  - Cada vez que una regla se aprende, los registros cubiertos por la regla se eliminan, y el proceso se repite con las instancias restantes.
- De manera simplista y general el algoritmo es un proceso de 3 pasos:
  - Crecimiento.
  - Poda.
  - Optimización.

... RIPPER

Crecimiento:

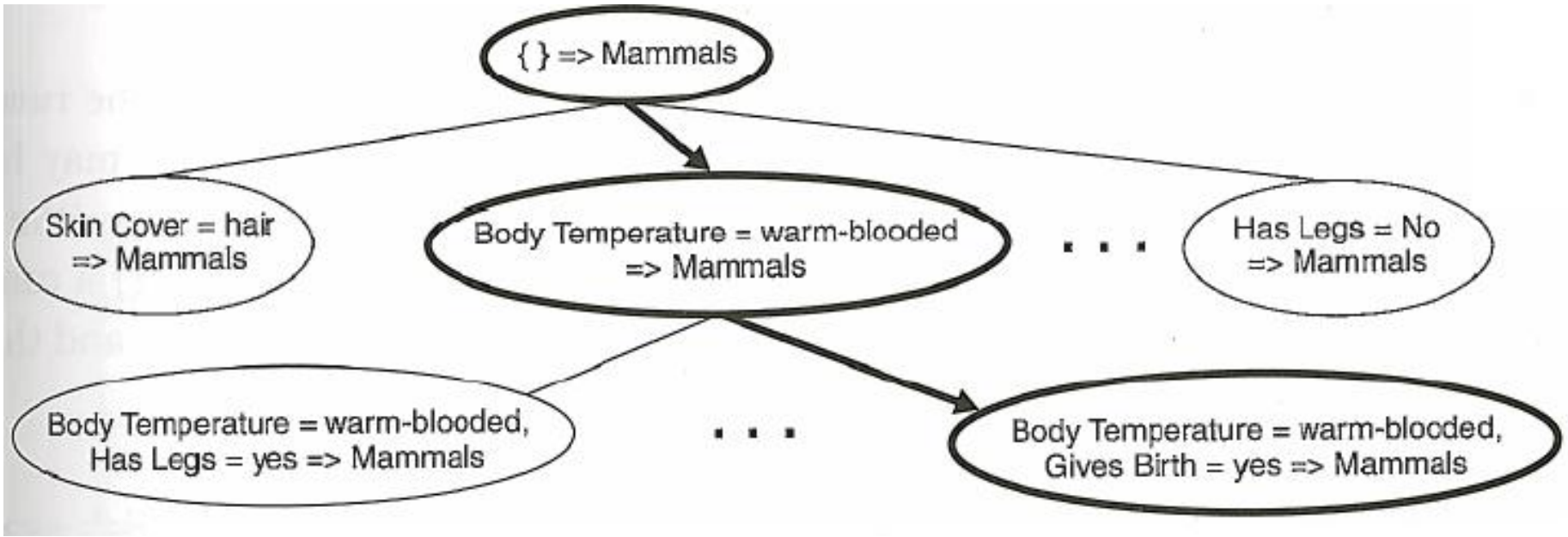
- Típicamente, las reglas crecen de lo lo general a lo particular.
- Se empieza con una regla vacía del lado izquiero y gradualmente se van agregando atributos.
- Al agregarlos se considera que se tiene una conjunción lógica como condición en el antecedente de la regla.

... RIPPER

Crecimiento:

- Típicamente, las reglas crecen de lo lo general a lo particular.
- Se empieza con una regla vacía del lado izquiero y gradualmente se van agregando atributos.
- Al agregarlos se considera que se tiene una conjunción lógica como condición en el antecedente de la regla.
- Ejemplo:
  - Empezar por lo general: $\{\} \rightarrow$ `mamífero`.
  - Agregar prueba para cada uno de los atributos restantes:
    - Cobertura de la piel = pelo $\rightarrow$ mamífero
    - Temperatura corporal = caliente $\rightarrow$ mamífero
    - ...
    - Tiene patas = sí $\rightarrow$ mamífero
- Cada vez que se agrega un atributo a la regla, se debe seleccionar el que mejore la calidad de la misma de acuerdo al conjunto de entrenamiento.
- Repetir el proceso, hasta que la regla resultante satisfaga un nivel de calidad aceptable.

... RIPPER


{ } => Mammals
Skin Cover = hair => Mammals
Body Temperature = warm-blooded => Mammals
Has Legs = No => Mammals
Body Temperature = warm-blooded, Has Legs = yes => Mammals
Body Temperature = warm-blooded, Gives Birth = yes => Mammals

... RIPPER

- Idealmente se espera que cuando se aprende una regla para una clase C, la regla cubra todas (o la mayoría) las tuplas de entrenamiento de la clase C y ninguna (o pocas) de las de otras clases.
- La condición de terminación puede ser: cuando no haya más tuplas de entrenamiento o la calidad de una regla esté por abajo del umbral especificado.
- El procedimiento `aprenderUnaRegla` encuentra la "mejor" regla para la clase actual, dado el conjunto de entrenamiento actual.

RIPPER vs árboles

- Ambos métodos podrían empezar a dividir el dataset usando el mismo atributo y probablemene dividan en el mismo lugar.
- Pero:
  - Los conjuntos de reglas probablemente, podrían ser más claros que el árbol de decisión si éste tiene subárboles replicados.
- También:
  - Si se tienen varias clases, el algoritmo de cobertura se concentra en una clase a la vez, en tanto el árbol de decisión toma en cuenta todas las clases.

Ejemplo RIPPER

```
> install.packages("RWeka")
> library(RWeka)

Attaching package: 'RWeka'

The following object is masked from 'package:OneR':

    OneR

> otroModelo <- JRip(Class ~ ., data=entrenamiento)
```

... Ejemplo RIPPER

```
> otroModelo
JRIP rules:
===========

(Uniformity of Cell Size=(3.24,10]) => Class=malignant (175.0/9.0)
(Clump Thickness = (5.49,10]) and (Uniformity of Cell Shape =
                                (3.51,10]) => Class=malignant (9.0/0.0)
(Bare Nuclei = (4.1,10]) => Class=malignant (28.0/9.0)
 => Class=benign (334.0/5.0)

Number of Rules : 4
```

... Ejemplo RIPPER

```
> summary(otroModelo)

=== Summary ===

Correctly Classified Instances         523               95.7875 %
Incorrectly Classified Instances        23                4.2125 %
Kappa statistic                          0.9103
Mean absolute error                      0.0717
Root mean squared error                  0.1893
Relative absolute error                 15.469  %
Root relative squared error             39.3363 %
Total Number of Instances              546

=== Confusion Matrix ===

   a   b   <-- classified as
 329  18 |   a = benign
   5 194 |   b = malignant
```

... Ejemplo RIPPER

```
> otraPredic <-predict(otroModelo, prueba)
> summary(otraPredic)
   benign malignant
       95        42

> eval_model(otraPredic, prueba)
Confusion matrix (absolute):
           Actual
Prediction  benign malignant Sum
  benign        94         1  95
  malignant      3        39  42
  Sum           97        40 137

Confusion matrix (relative):
           Actual
Prediction  benign malignant  Sum
  benign      0.69      0.01 0.69
  malignant   0.02      0.28 0.31
```

... Ejemplo RIPPER

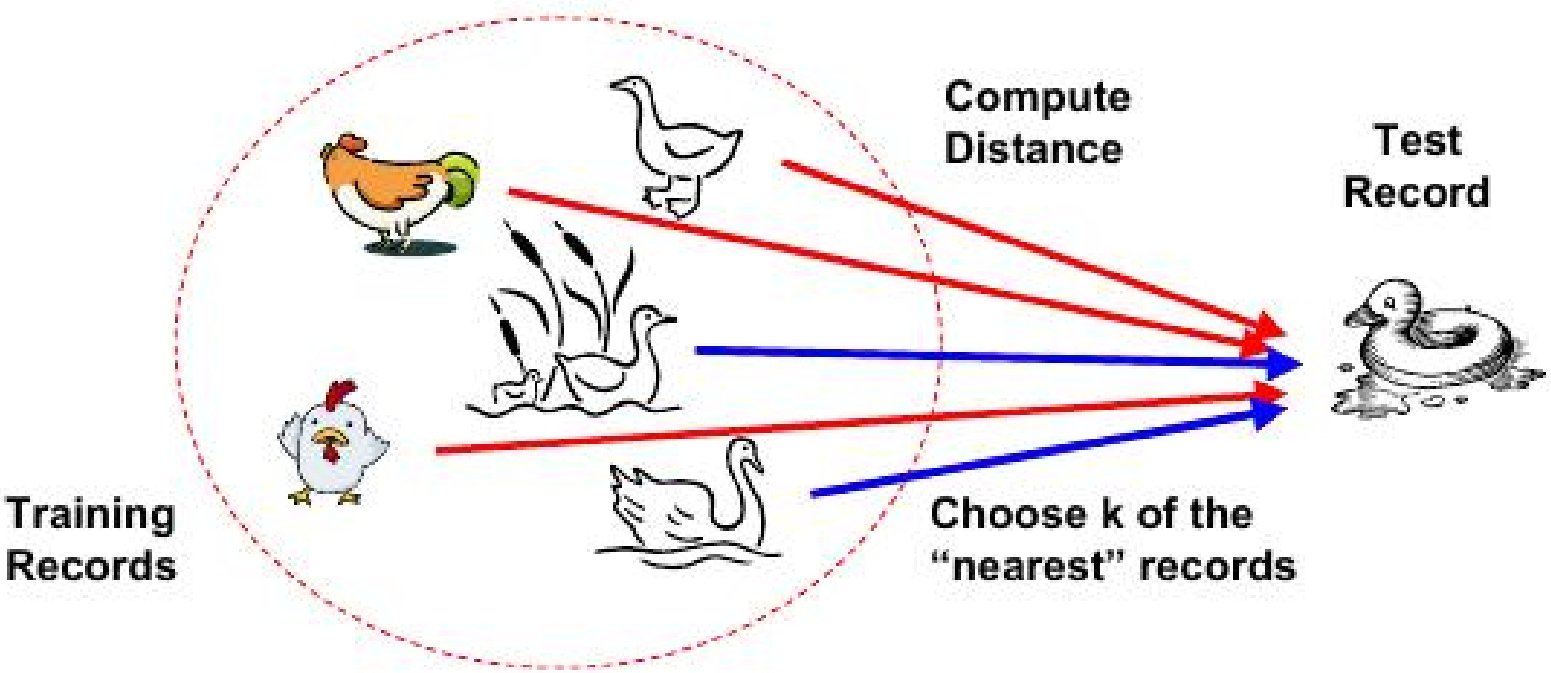
```
Accuracy:
0.9708 (133/137)

Error rate:
0.0292 (4/137)

Error rate reduction (vs. base rate):
0.9 (p-value = 1.252e-15)

>
```

Ventajas de clasificación basada en reglas

- Tienen tanta expresividad como los árboles de decisión.
- Fáciles de interpretar.
- Fáciles de generar.
- Pueden clasificar, rápidamente, nuevas instancias.
- El rendimiento es comparable con los árboles de decisión.

Clasificación KNN

- El algoritmo de k vecinos más cercanos, o Knn por sus siglas en Inglés (K-nearest neighbors) es un algoritmo de aprendizaje supervisado.
- La idea básica es muy sencilla:
  - Si algo camina como un pato, dice cuac, cuac, y parece pato entonces probablemente es un pato.



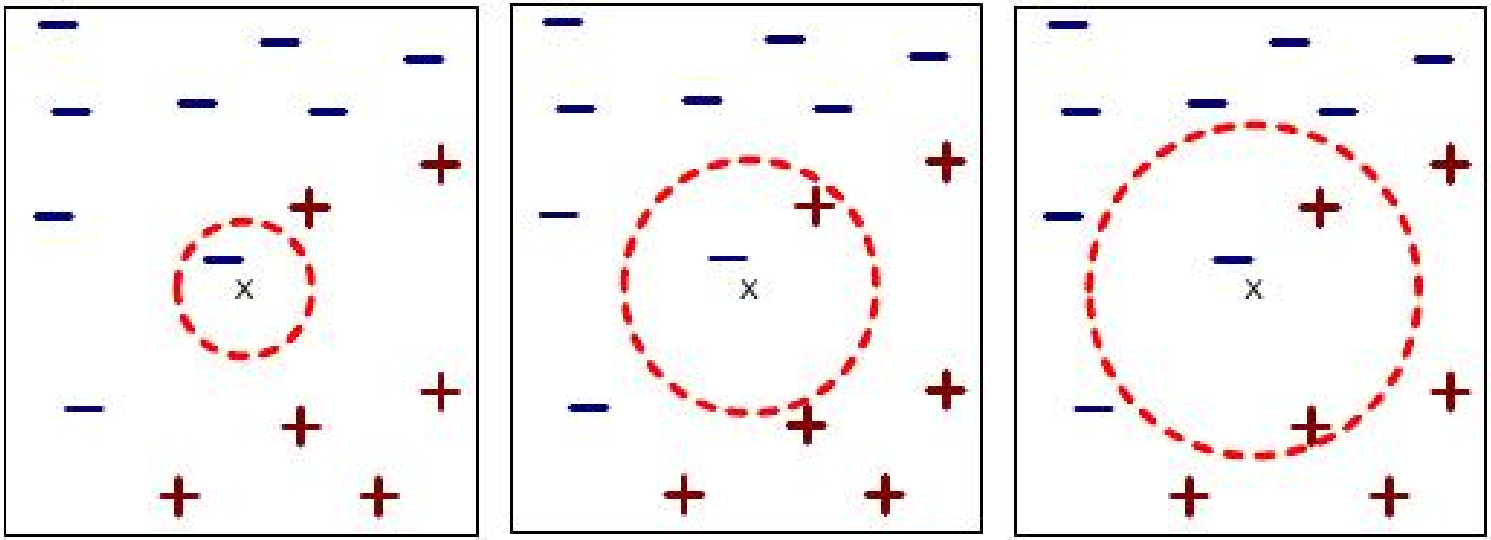
... Introducción KNN

- KNN es el proceso de clasificar datos no etiquetados asignándoles la clase de la etiqueta que es más parecida e ellos.
- Es una técnica de aprendizaje supervisado.
- Apropiada para clasificación cuando las relaciones entre los atributos independientes y el dependiente son numerosos, complicados o extremadamente difíciles de entender.
  - SI un concepto es difícil de definir pero se sabe cómo hacerlo
- Se requieren tres cosas:
  - El conjunto de registros etiquetados.
  - Una función de distancia entre registros.
  - El valor de **k**, la cantidad de vecinos más cercanos a recuperar.

... Introducción KNN

- KNN es el proceso de clasificar datos no etiquetados asignándoles la clase de la etiqueta que es más parecida e ellos.
- Es una técnica de aprendizaje supervisado.
- Apropiada para clasificación cuando las relaciones entre los atributos independientes y el dependiente son numerosos, complicados o extremadamente difíciles de entender.
  - SI un concepto es difícil de definir pero se sabe cómo hacerlo
- Se requieren tres cosas:
  - El conjunto de registros etiquetados.
  - Una función de distancia entre registros.
  - El valor de **k**, la cantidad de vecinos más cercanos a recuperar.
- Para clasificar un registro desconocido:
  - Calcula su distancia a los registros de entrenamiento.
  - Identifica los k más cercanos.
  - Determina la etiqueta del nuevo registro de acuerdo a las existentes tomando, por ejemplo, la clase de la mayoría de los vecinos cercanos.

... Introducción



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

- Los k vecinos más cercanos de un registro X son los registros (puntos) que tienen menor distancia a X.

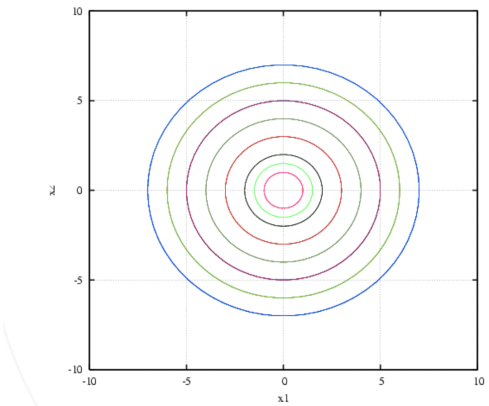
Detalles

- Calcular su distancia a los registros de entrenamiento.

Detalles

- Calcular su distancia a los registros de entrenamiento.
  - Distancia Euclidiana:

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

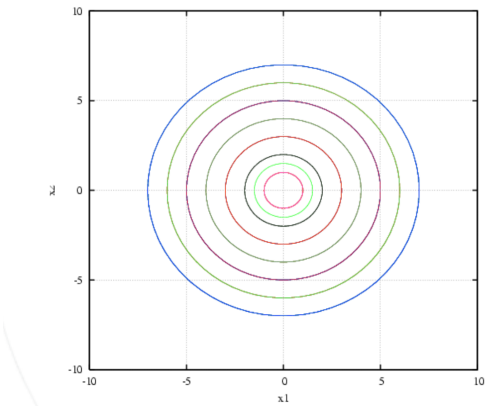


- Determinar la etiqueta del nuevo registro de acuerdo a las existentes.

Detalles

- Calcular su distancia a los registros de entrenamiento.
  - Distancia Euclidiana:

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$



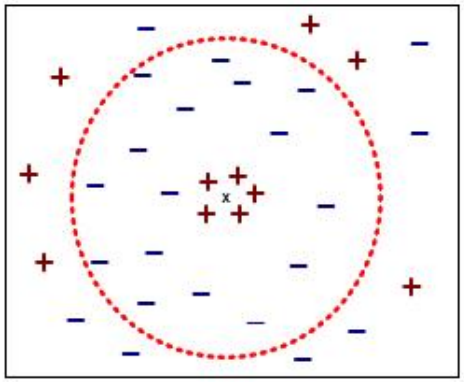
- Determinar la etiqueta del nuevo registro de acuerdo a las existentes.
  - Toma la etiqueta más frecuente, entre los k vecinos cercanos.

... Detalles

- Determinar el valor de k:

... Detalles

- Determinar el valor de k:
  - Si k es demasiado pequeña, el algoritmo es sensible al ruido.
  - Si k es demasiado grande, la vecindad puede incluir puntos de otras clases.



- Es difícil determinar el óptimo pero es práctica común elegir $k = \sqrt{cantidad\ de\ registros\ de\ entrenamiento}$.
- Dependerá de la complejidad de concepto a aprender y de la cantidad de registros de entrenamiento.

Elección de la k (Ejemplo)

Ingrediente	Dulzura	Efervescencia	Tipo
Monster Energy	8	8	Energizante
Boing	9	1	Bebida saludable
Coca cola	4	8	Bebida dulce
Vodka	2	1	Bebida fuerte

Elección de la k (Ejemplo)

Ingrediente	Dulzura	Efervescencia	Tipo
Monster Energy	8	8	Energizante
Boing	9	1	Bebida saludable
Coca cola	4	8	Bebida dulce
Vodka	2	1	Bebida fuerte

- Se desea saber la categoría a la que pertenece un jugo Jumex(8,2)

Elección de la k (Ejemplo)

Ingrediente	Dulzura	Efervescencia	Tipo
Monster Energy	8	8	Energizante
Boing	9	1	Bebida saludable
Coca cola	4	8	Bebida dulce
Vodka	2	1	Bebida fuerte

- Se desea saber la categoría a la que pertenece un jugo Jumex(8,2)

Ingrediente	Dulzura	Efervescencia	Tipo	Distancia
Monster Energy	8	8	Energizante	6.8
Boing	9	1	Bebida saludable	1.41
Coca cola	4	8	Bebida dulce	7.21
Vodka	2	1	Bebida fuerte	6.8

- Si k=1,

Elección de la k (Ejemplo)

Ingrediente	Dulzura	Efervescencia	Tipo
Monster Energy	8	8	Energizante
Boing	9	1	Bebida saludable
Coca cola	4	8	Bebida dulce
Vodka	2	1	Bebida fuerte

- Se desea saber la categoría a la que pertenece un jugo Jumex(8,2)

Ingrediente	Dulzura	Efervescencia	Tipo	Distancia
Monster Energy	8	8	Energizante	6.8
Boing	9	1	Bebida saludable	1.41
Coca cola	4	8	Bebida dulce	7.21
Vodka	2	1	Bebida fuerte	6.8

- Si k=1, el algoritmo considera Boing como el más cercano al Jumex y por tanto es bebida saludable.

Elección de la k (Ejemplo)

Ingrediente	Dulzura	Efervescencia	Tipo
Monster Energy	8	8	Energizante
Boing	9	1	Bebida saludable
Coca cola	4	8	Bebida dulce
Vodka	2	1	Bebida fuerte

- Se desea saber la categoría a la que pertenece un jugo Jumex(8,2)

Ingrediente	Dulzura	Efervescencia	Tipo	Distancia
Monster Energy	8	8	Energizante	6.8
Boing	9	1	Bebida saludable	1.41
Coca cola	4	8	Bebida dulce	7.21
Vodka	2	1	Bebida fuerte	6.8

- Si k=1, el algoritmo considera Boing como el más cercano al Jumex y por tanto es bebida saludable.
- Si k=3,

Elección de la k (Ejemplo)

Ingrediente	Dulzura	Efervescencia	Tipo
Monster Energy	8	8	Energizante
Boing	9	1	Bebida saludable
Coca cola	4	8	Bebida dulce
Vodka	2	1	Bebida fuerte

- Se desea saber la categoría a la que pertenece un jugo Jumex(8,2)

Ingrediente	Dulzura	Efervescencia	Tipo	Distancia
Monster Energy	8	8	Energizante	6.8
Boing	9	1	Bebida saludable	1.41
Coca cola	4	8	Bebida dulce	7.21
Vodka	2	1	Bebida fuerte	6.8

- Si k=1, el algoritmo considera Boing como el más cercano al Jumex y por tanto es bebida saludable.
- Si k=3, se consideran 3 vecinos cercanos y por votación puede ganar Bebida fuerte.

Consideraciones

- Los atributos deben tener valores en una escala similar, para prevenir que las medidas de distancia sean dominadas por alguno de ellos.
- Ejemplo:
  - La altura de una persona puede variar de 1.5m a 1.8m
  - El peso de una persona puede variar de 48 a 100 Kg.
  - El ingreso de una persona puede variar de $10K a $1M

Consideraciones

- Los atributos deben tener valores en una escala similar, para prevenir que las medidas de distancia sean dominadas por alguno de ellos.
- Ejemplo:
  - La altura de una persona puede variar de 1.5m a 1.8m
  - El peso de una persona puede variar de 48 a 100 Kg.
  - El ingreso de una persona puede variar de $10K a $1M
- Solución:

Consideraciones

- Los atributos deben tener valores en una escala similar, para prevenir que las medidas de distancia sean dominadas por alguno de ellos.
- Ejemplo:
  - La altura de una persona puede variar de 1.5m a 1.8m
  - El peso de una persona puede variar de 48 a 100 Kg.
  - El ingreso de una persona puede variar de \$10K a \$1M
- Solución: Normalizar los datos a que estén entre 0 y 1.

$$x_{nueva} = \frac{x - min(x)}{max(x) - min(x)}$$

Características

- Es altamente imparcial y no requiere suposiciones previas de los datos.
- Es sencillo y efectivo.
- Es un clasificador de aprendizaje perezoso.
  - No construye modelos explícitamente como los arboles de decisión y los sistemas basados en reglas.
- La clasificación de registros desconocidos es relativamente cara, pues requiere grandes cantidades de memoria.
- Los datos nominales y los datos perdidos requieren procesamiento adicional.

Ejemplo con iris

```
> install.packages('class')
> library(class)

> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa

> table(iris[5])

    setosa versicolor  virginica
    50         50         50
```

Conjunto entrenamiento/prueba

```
> set.seed(1234)   #Para tomar siempre los mismos datos
>ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33))

> iris.training <- iris[ind==1, 1:4] #Muestra con 2/3 del conjunto
                                     # que tienen 1)
> head(iris.training)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
6          5.4         3.9          1.7         0.4
7          4.6         3.4          1.4         0.3

> iris.test <- iris[ind==2, 1:4]     #Muestra con 1/3 del conjunto
```

... Conjunto entrenamiento/prueba

```
> head(iris.test)

   Sepal.Length Sepal.Width Petal.Length Petal.Width
5           5.0         3.6          1.4         0.2
11          5.4         3.7          1.5         0.2
14          4.3         3.0          1.1         0.1
16          5.7         4.4          1.5         0.4
26          5.0         3.0          1.6         0.2
28          5.2         3.5          1.5         0.2

> iris.trainLabels <- iris[ind==1, 5]
> iris.testLabels <- iris[ind==2, 5]
```

Creación del modelo

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa

# "Crea el modelo"
iris_pred <- knn(train=iris.training, test = iris.test,
                 cl=iris.trainLabels, k= 3)
> iris_pred
 [1] setosa     setosa     setosa     setosa     setosa     setosa
 [7] setosa     setosa     setosa     setosa     setosa     setosa
[13] versicolor versicolor versicolor versicolor versicolor versicol
[19] versicolor versicolor versicolor versicolor versicolor versicol
[25] virginica  virginica  virginica  virginica  versicolor virgini
```

Validación del modelo

```
> irisTestLabels <-data.frame(iris.testLabels)
> merge <- data.frame(iris_pred, iris.testLabels)
> names(merge) <- c("Predicción", "Conocido")
>
> merge   #Aquí vemos como clasificó
          Predicción        Conocido
1             setosa          setosa
2             setosa          setosa
3             setosa          setosa
4             setosa          setosa
5             setosa          setosa
6             setosa          setosa
7             setosa          setosa
8             setosa          setosa
9             setosa          setosa
10            setosa          setosa
11            setosa          setosa
...
```

... Validación del modelo

```
# La matriz de confusión
> sum(iris_pred == iris.testLabels)
[1] 39
>
> table(iris_pred, iris.testLabels)
            iris.testLabels
iris_pred    setosa versicolor virginica
  setosa         12          0         0
  versicolor      0         12         1
  virginica       0          0        15
```

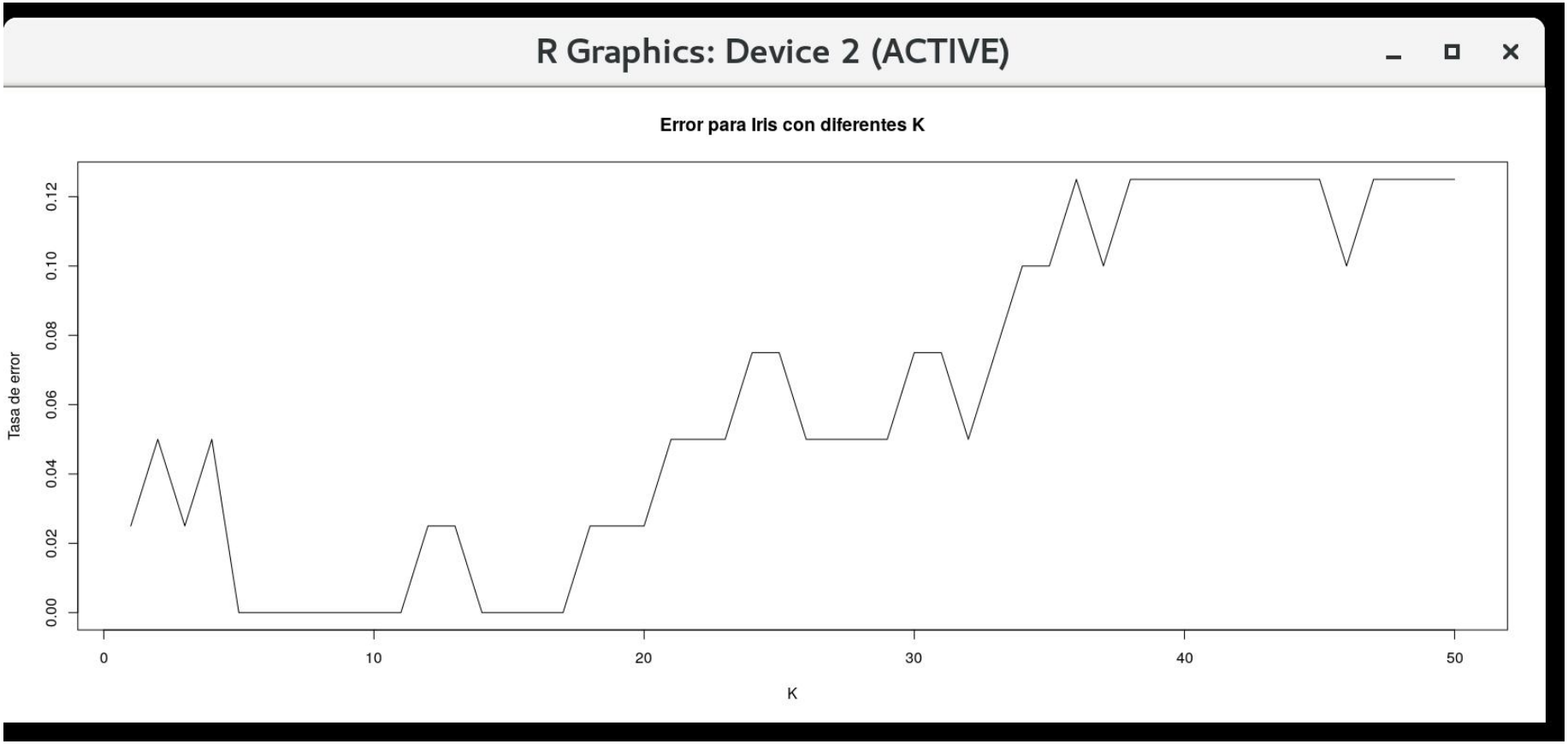
... Validación del modelo

```
> library(gmodels)
> CrossTable(x=iris.testLabels, y = iris_pred)

   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  40
```

	iris_pred			
iris.testLabels	setosa	versicolor	virginica	Row Total
setosa	12	0	0	12
	1.000	0.000	0.000	0.300
	1.000	0.000	0.000	
	0.300	0.000	0.000	
versicolor	0	12	0	12
	0.000	1.000	0.000	0.300
	0.000	0.923	0.000	
	0.000	0.300	0.000	
virginica	0	1	15	16
	0.000	0.062	0.938	0.400
	0.000	0.077	1.000	
	0.000	0.025	0.375	
Column Total	12	13	15	40
	0.300	0.325	0.375	

Elección de la k

```
> iris.acc <-numeric()  #Variable numerica

#Aplica el knn con k = i, desde 1 hasta 50
> for(i in 1:50){
   predict<-knn(train=iris.training, test = iris.test,
                cl=iris.trainLabels, k= i)
   iris.acc<-c(iris.acc, mean(predict==iris.testLabels))
}

#Grafica
 plot(1-iris.acc,type="l",ylab="Tasa de error",
    xlab="K",main="Error para Iris con diferentes k")
```


R Graphics: Device 2 (ACTIVE)
Error para Iris con diferentes K
Tasa de error
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
10
20
30
40
50
K

... Elección de la k

```
#Ahora se probará con 100 muestras y el tamaño de k entre 1 y 20
 trial.sum<-numeric(20)
 trial.n<-numeric(20)
 set.seed(6033850)
 for(i in 1:100){
 ir.sample<-sample(1:nrow(iris),size=nrow(iris)*.7)
 ir.train<-iris[ir.sample,]
 ir.test<-iris[-ir.sample,]
 test.size<-nrow(ir.test)
 for(j in 1:20){
 predict<-knn(ir.train[,-5],ir.test[,-5], ir.train$Species,k=j)
 trial.sum[j]<-trial.sum[j]+sum(predict==ir.test$Species)
 trial.n[j]<-trial.n[j]+test.size
 }
}
```

```
plot(1-trial.sum / trial.n,type="l",ylab="Error",
   xlab="K",main="Error para Iris con k variable y 100 muestras")
```