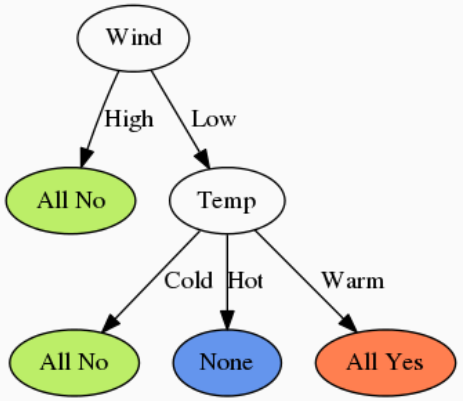


ÁRBOLES DE CLASIFICACIÓN

C4.5, C5.0 Y RANDOM FOREST

Luis María Costero Valero
Jesús Javier Doménech Arellano

Enero 2016

Índice

ID3

TDIDT[1]

TDIDT

```
tdidt( C:conjunto_datos, l:atributos_candidatos)
  cp := clase que aparece más veces en C

  si todas las instancias en C son de clase cj
  entonces
    return new Hoja(cj);

  si l es vacía entonces
    return new Hoja(cp);

  a := selecciona_atributo(C,l);
  n := new NodoInterno(a);
  para cada valor aj del atributo a
    Cj := particion(C,a,aj);
    si Cj = ∅ entonces n' := new Hoja(cp);
    si no n' := tdidt(Cj,l\{a});
    n.añadeHijo(n',aj);
  return n;
```

[1] Diapositiva sacada del temario de SGDI

Ganancia de información

- Lo más interesante es realizar la selección del atributo teniendo en cuenta la calidad del particionado que genera: nos interesa que los subconjuntos resultantes tengan la mínima variedad de clases posibles.
- Para ello utilizaremos la medida de **ganancia de información** (o reducción de la **entropía**), lo que da lugar al algoritmo **ID3**.

[2] Diapositiva sacada del temario de SGDI

C4.5

C4.5

El algoritmo ID3 es mejorado por el C4.5. Esta mejora, aparte de la optimización de partes de código, incluye[3]:

- Permite atributos continuos.
- Permite dar un peso diferente a cada atributo.
- Permite a una instancia no tener definido un valor en sus atributos.
- Mejora la selección del atributo clasificador.
- Realiza una poda del árbol después de la creación.

[3] Artículo con las bases del algoritmo [Quinlan, 1986]

Algoritmo C4.5[4]

Split y Funciones Test: Para divir en subconjuntos las instancias test se divide el dominio donde haya mayor ganancia de información. Esta división se traduce en funciones test de tipo $x > 40$ ó $x < 4$, y para atributos discretos $x = soleado$.

[4]Obtenido del libro [**?**] y su review [Salzberg, 1994]

Algoritmo C4.5[4]

Split y Funciones Test: Para divir en subconjuntos las instancias test se divide el dominio donde haya mayor ganancia de información. Esta división se traduce en funciones test de tipo $x > 40$ ó $x < 4$, y para atributos discretos $x = soleado$.

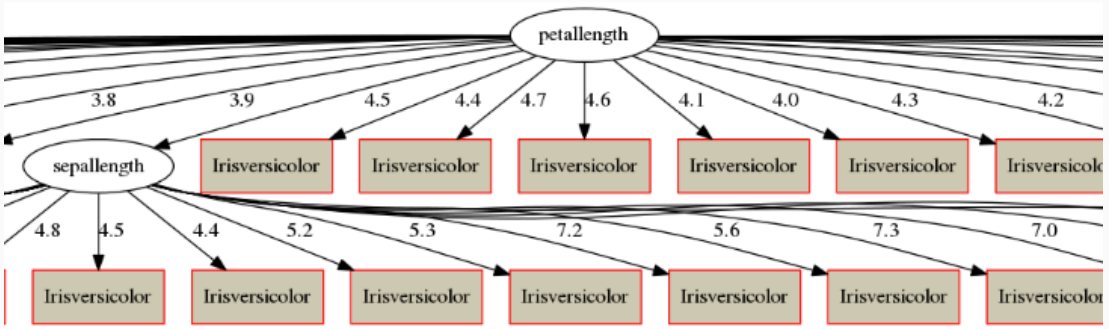
Selección del atributo: La selección del atributo por el que dividir consiste en escoger el atributo con mayor ganancia de información normalizado y ponderado.

Para ello, se tiene en cuenta la proporción de instancias que queda en cada rama para el atributo candidato y el peso de importancia dado a dicho atributo. Siendo D el conjunto de instancias, la fórmula para el atributo i queda:

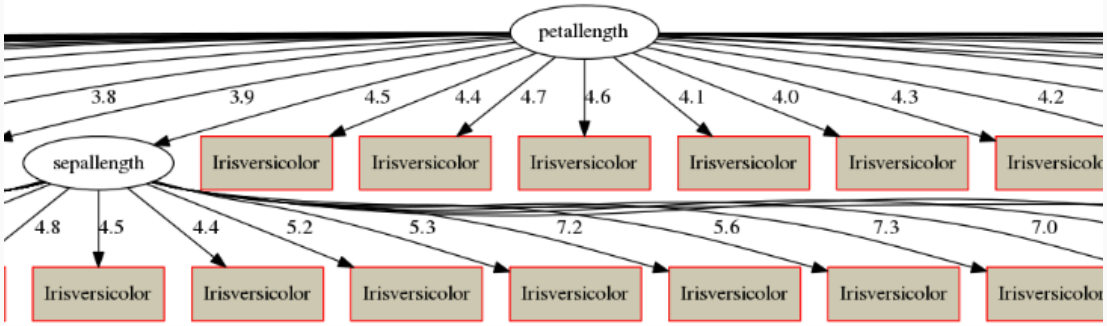
$$Ganancia_i = Peso_i * \sum_j \frac{|D_j|}{|D|} * Info_j$$

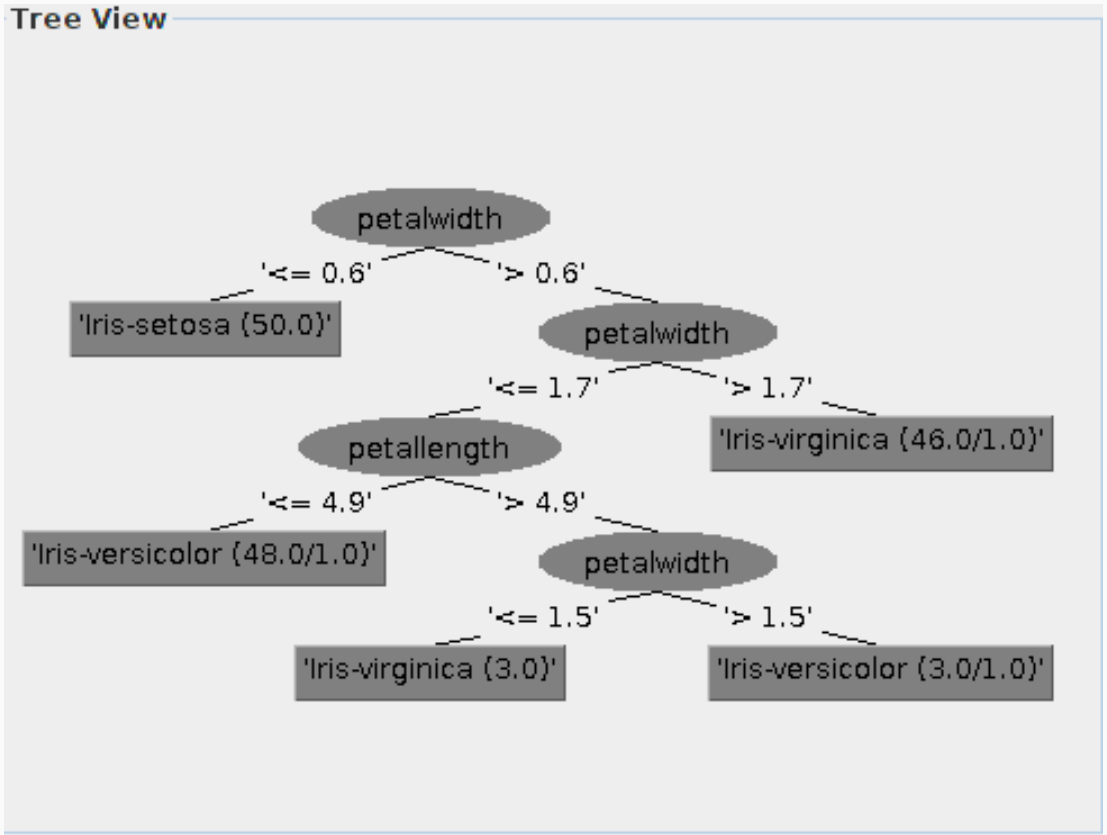
[4]Obtenido del libro [**?**] y su review [Salzberg, 1994]

Ejemplo – ID3 vs C4.5 (atributos continuos)



EJEMPLO – ID3 VS C4.5 (ATRIBUTOS CONTINUOS)





C4.8 (J48)[5]: Mejoras a C4.5

Problemas:

"The accuracy of T2's trees rivalled or surpassed C4.5's on 8 of the 15 datasets, including all but one of the datasets having only continuous attributes."[Auer et al., 1995]

"C4.5's performance was significantly improved on two data sets ... using the entropy discretization method and did no significantly degrade on any datasets [...] We conjeture that the C4.5 induction algorithm is not taking full advantage of possible local discretization."[Dougherty et al., 1995]

C4.8 (J48)[5]: Mejoras a C4.5

Problemas:

"The accuracy of T2's trees rivalled or surpassed C4.5's on 8 of the 15 datasets, including all but one of the datasets having only continuous attributes."[Auer et al., 1995]

"C4.5's performance was significantly improved on two data sets ... using the entropy discretization method and did no significantly degrade on any datasets [...] We conjeture that the C4.5 induction algorithm is not taking full advantage of possible local discretization."[Dougherty et al., 1995]

Solución [Quinlan, 1996]:

- Aumentar el coste de usar atributos continuos disminuyendo el peso de importancia del atributo.
- Simplificar la división de un atributo continuo, en lugar de maximizar la ganacia, basta con superar un límite.
- Se añade el generador de reglas.

C5.0

C5.0

C5.0 tiene licencia comercial y licencia GPL para un solo proceso.

Mejoras [6]:

- Aumenta el rendimiento del algoritmo.
- Reduce el consumo de memoria local y total.
- Devuelve árboles de clasificación reducidos.
- Poda atributos irrelevantes para la clasificación.
- Hace la ponderación de atributos más precisa.
- Agrupa valores de atributos discretos en una misma rama.
- Genera reglas con más acierto que las de C4.8.
- Facilita la aplicación de técnicas como bagging y boosting, mejorando sus resultados.[7]

[6] Basado en [Pandya and Pandya, 2015] y la página oficial del autor [Quinlan,]
[7] [Freund and Mason, 1999]

C5.0: Example in R[8]

```
crx<-read.table(file="./crx.data",header=FALSE,sep=",")
head( crx, 6 )
```

C5.0: Example in R[8]

```
crx<-read.table(file="./crx.data",header=FALSE,sep=",")
head( crx, 6 )
crx <- crx[ sample( nrow( crx ) ), ]
X <- crx[,1:15]
y <- crx[,16]
```

C5.0: Example in R[8]

```
crx<-read.table(file="./crx.data",header=FALSE,sep=",")
head( crx, 6 )
crx <- crx[ sample( nrow( crx ) ), ]
X <- crx[,1:15]
y <- crx[,16]
trainX <- X[1:600,]
trainy <- y[1:600]
testX <- X[601:690,]
testy <- y[601:690]
```

C5.0: Example in R[8]

```
crx<-read.table(file="./crx.data",header=FALSE,sep=",")
head( crx, 6 )
crx <- crx[ sample( nrow( crx ) ), ]
X <- crx[,1:15]
y <- crx[,16]
trainX <- X[1:600,]
trainy <- y[1:600]
testX <- X[601:690,]
testy <- y[601:690]
library(C50)
model <- C50::C5.0( trainX, trainy )
summary( model )
```

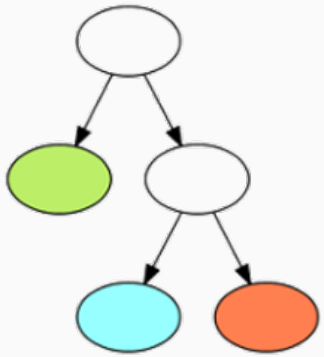
Random Forest

Introducción – I

Hasta ahora:

Un conjunto de entrenamiento $\rightarrow$ Un árbol de clasificación

Sexo	Altura	Peso	Exp.	Act.
H	1,55	45	A+	Fútbol
H	1,67	58	C	Fútbol
M	1,45	45	B	Fútbol
M	1,58	50	A+	Pádel
H	1,20	40	B	Fútbol
M	1,80	60	A	Pádel
...	...	...	...	...

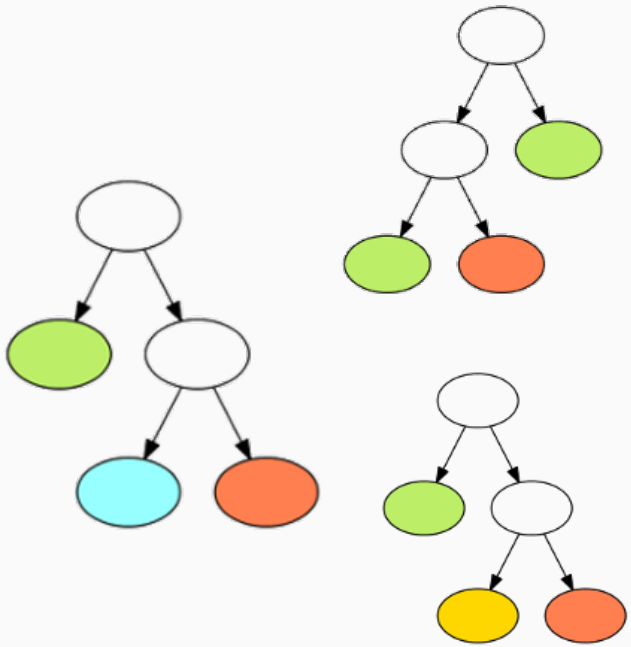


Introducción – I

Random forest:

Un conjunto de entrenamiento $\rightarrow$ Varios árboles de clasificación

Sexo	Altura	Peso	Exp.	Act.
H	1,55	45	A+	Fútbol
H	1,67	58	C	Fútbol
M	1,45	45	B	Fútbol
M	1,58	50	A+	Pádel
H	1,20	40	B	Fútbol
M	1,80	60	A	Pádel
...	...	...	...	...



Introducción – II

Random forest[9] **(No me gusta este título)**

- ¿Cómo se clasifica una instancia?
- ¿Cómo se generan varios árboles con el mismo conjunto de entrenamiento?
- ¿Cómo se genera un árbol en concreto?

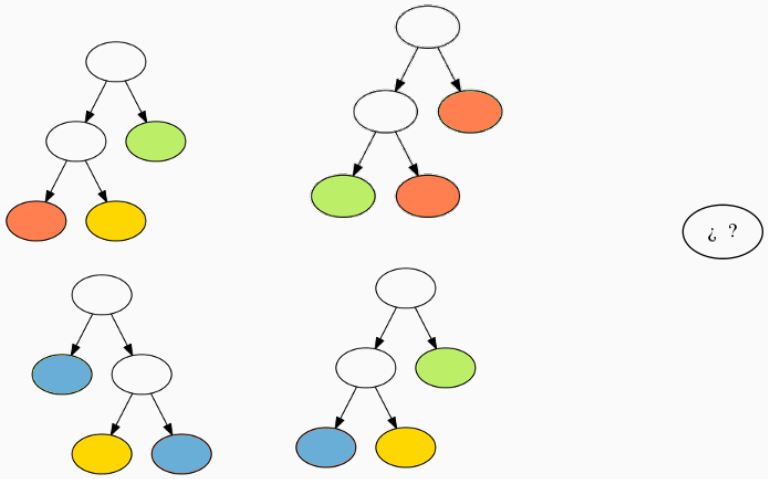
[9] Técnica propuesta por primera vez en [Breiman, 2001].

Clasificación de instancias

Al existir varios árboles, pueden existir varias clases posibles para una instancia.

La clase final será aquella que más veces aparece elegida (moda).

```
1  clasifica(x):
2    for_each a in arboles:
3        cjto += clasifica(x, a)
4
5    return moda(cjto)
```

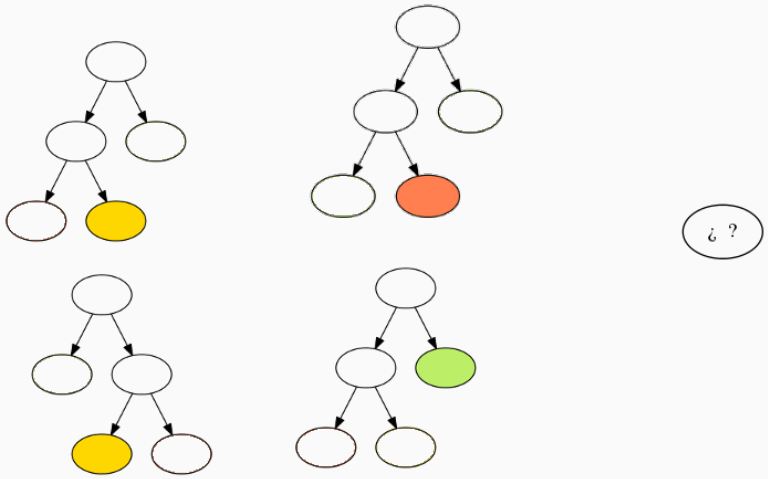


Clasificación de instancias

Al existir varios árboles, pueden existir varias clases posibles para una instancia.

La clase final será aquella que más veces aparece elegida (moda).

```
1  clasifica(x):
2    for_each a in arboles:
3        cjto += clasifica(x, a)
4
5    return moda(cjto)
```

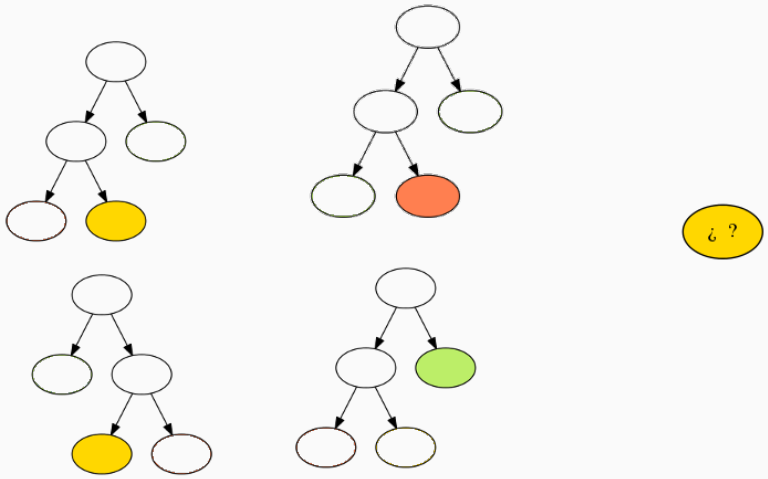


Clasificación de instancias

Al existir varios árboles, pueden existir varias clases posibles para una instancia.

La clase final será aquella que más veces aparece elegida (moda).

```
1  clasifica(x):
2    for_each a in arboles:
3        cjto += clasifica(x, a)
4
5    return moda(cjto)
```



Generación de árboles — Selección de instancias

Esto es un borrador rápido de lo que quiero poner aquí:

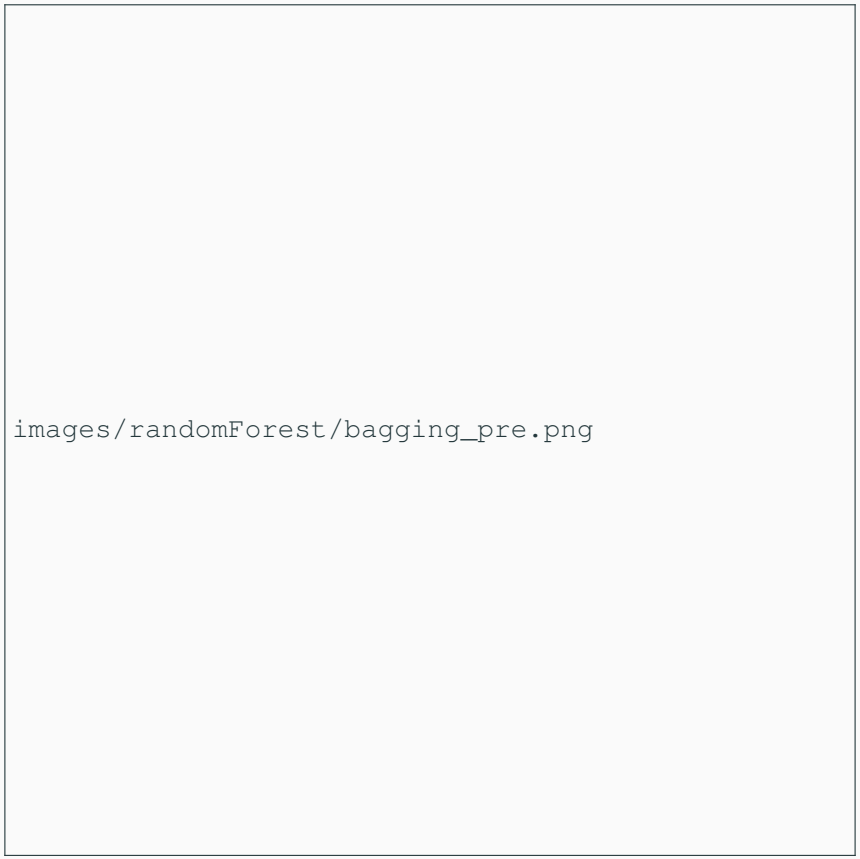
- Nombrar la técnica de bagging o bootstrap aggregating, y poner referencia.
- Mencionar los tamaños de cada cosa, y el número de árboles es ilimitado.
- Nombrar muestreo aleatorio con reemplazamiento.
- Explicar que significa con reemplazamiento.
- Poner el ejemplo de la tabla, y sacar flechas con los distintos árboles. (opcional, puede quedar chulo).

Generación de árboles — Selección de instancias

Problema: A partir de un mismo conjunto de entrenamiento se desean obtener distintos árboles de clasificación.

Solución: Técnica conocida como **Bagging** o *Bootstrap aggregating*:

- Dado un conjunto de entrenamiento con N instancias, obtener B conjuntos de entrenamiento de tamaño n' ($n' < N$).
- Utilizar *muestreo aleatorio con reemplazamiento*.



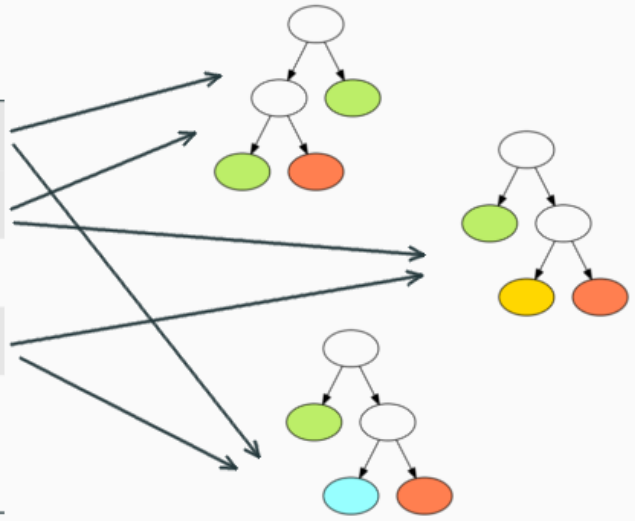
Generación de árboles — Selección de instancias

Problema: A partir de un mismo conjunto de entrenamiento se desean obtener distintos árboles de clasificación.

Solución: Técnica conocida como **Bagging** o *Bootstrap aggregating*:

- Dado un conjunto de entrenamiento con N instancias, obtener B conjuntos de entrenamiento de tamaño n' ($n' < N$).
- Utilizar *muestreo aleatorio con reemplazamiento*.

Sexo	Altura	Peso	Exp.	Act.
H	1,55	45	A+	Fútbol
H	1,67	58	C	Fútbol
M	1,45	45	B	Fútbol
M	1,58	50	A+	Pádel
H	1,20	40	B	Fútbol
...	...	...	...	...



Generación de árboles — Selección de atributos

Bibliografía

Bibliografía I


Auer, P., Holte, R. C., and Maass, W. (1995).
Theory and applications of agnostic pac-learning with small decision trees.
In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 21–29.


Breiman, L. (2001).
Random forests.
Machine Learning, 45(1):5–32.


Dougherty, J., Kohavi, R., Sahami, M., et al. (1995).
Supervised and unsupervised discretization of continuous features.
In *Machine learning: proceedings of the twelfth international conference*, volume 12, pages 194–202.

Bibliografía II


Freund, Y. and Mason, L. (1999).
The alternating decision tree learning algorithm.
In *icml*, volume 99, pages 124–133.


Pandya, R. and Pandya, J. (2015).
C5. 0 algorithm to improved decision tree with feature selection and reduced error pruning.
International Journal of Computer Applications, 117(16).


Quinlan, J. R.
Company jr quilan.
[Web; accedido el 28-12-2015].


Quinlan, J. R. (1986).
Induction of decision trees.
Machine learning, 1(1):81–106.

Bibliografía III


Quinlan, J. R. (1996).
Improved use of continuous attributes in c4.5.
Journal of artificial intelligence research, pages 77–90.


Salzberg, S. L. (1994).
C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993.
Machine Learning, 16(3):235–240.

¿ ?