Árboles de Clasificación y Regresión

1EST17 - Aprendizaje Estadístico I

Mg. Enver Gerald Tarazona Vargas

enver.tarazona@pucp.edu.pe

Maestría en Estadística

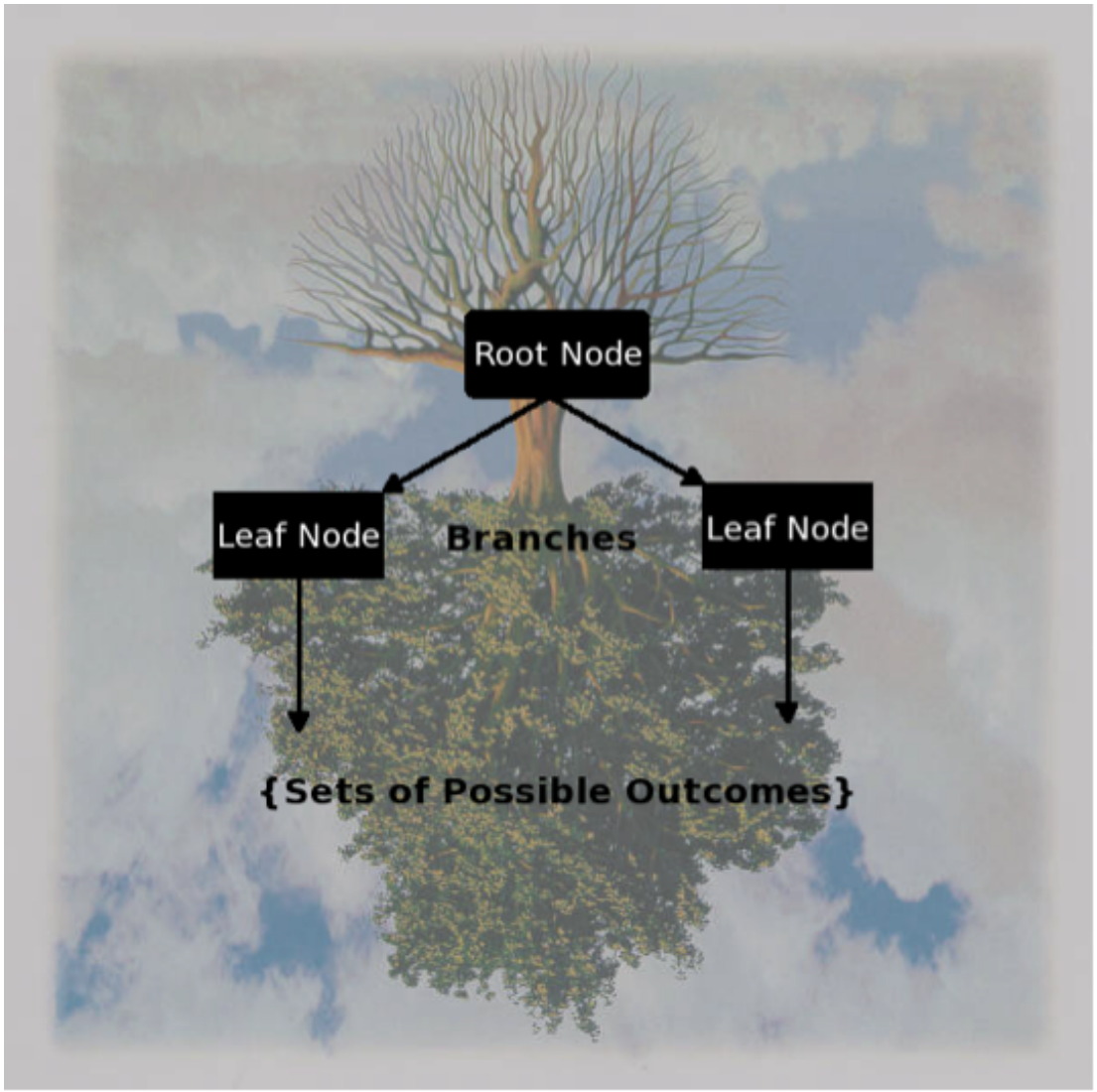
Escuela de Posgrado

Resumen I

Representación

Representación

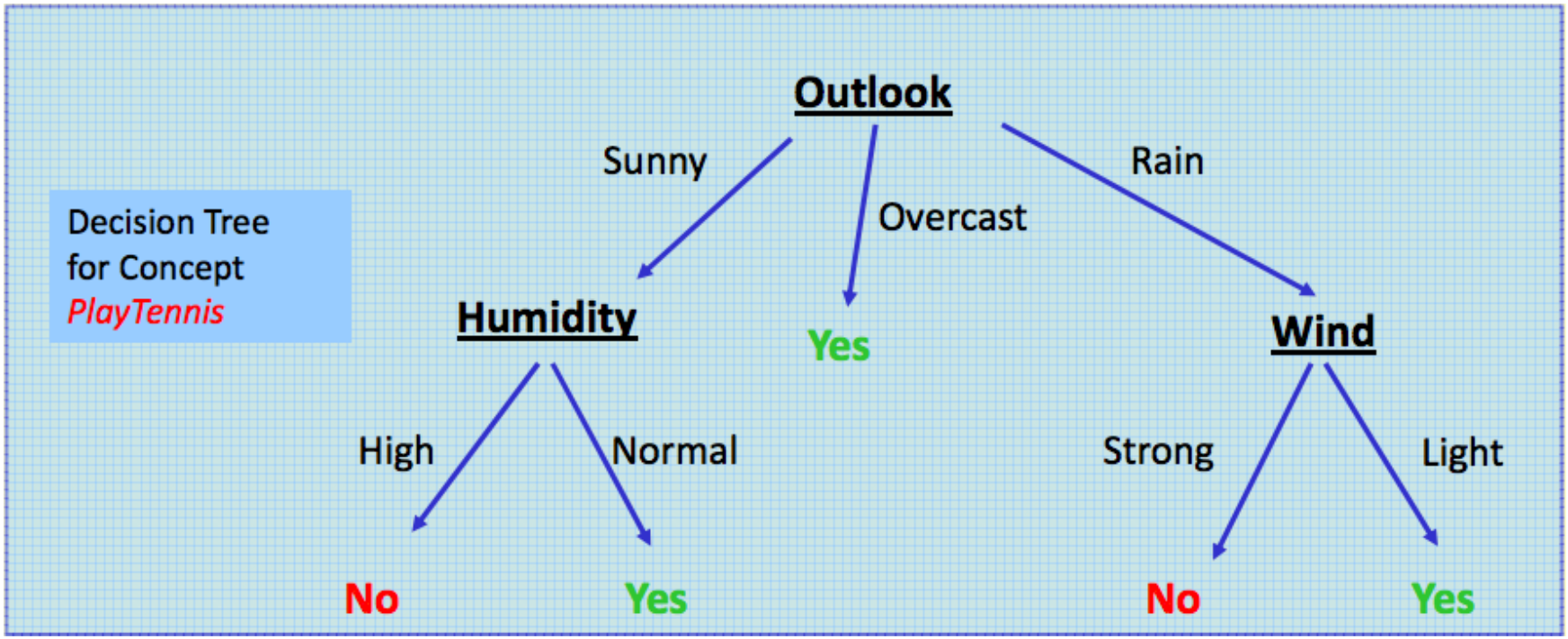
Representación



Partes de un Árbol de Decisión I

- **Nodo Interno (Root Node)**: denota una prueba sobre un atributo.
- **Rama (Branch)**: corresponde a un valor de atributo y representa el resultado de una prueba
- **Nodo Terminal (Leaf Node)**: representa una etiqueta de clase o de distribución de clase
- Cada camino es una conjunción de valores de atributos

Ejemplo de Árbol de Decisión

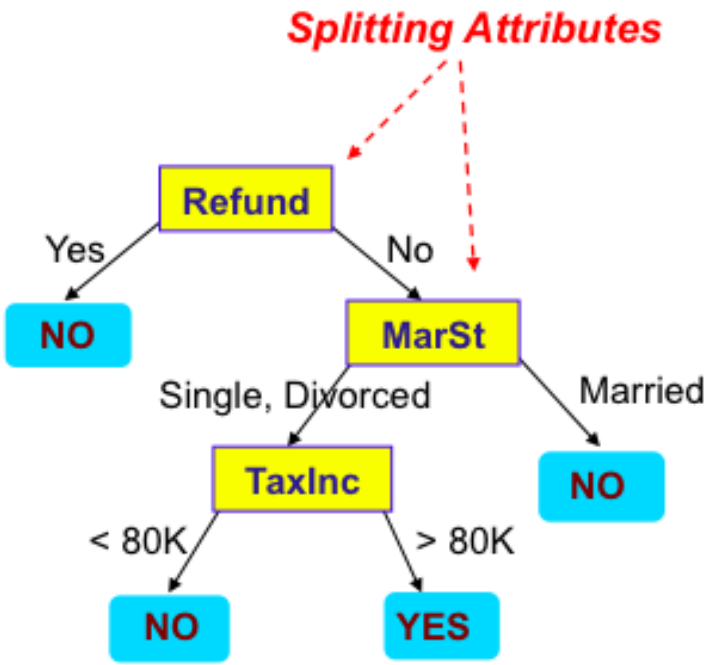


Ejemplo de Árbol de Decisión

Tid	Refund	Marital Status	Taxable Income	Cheat
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

categorical categorical continuous class

Training Data

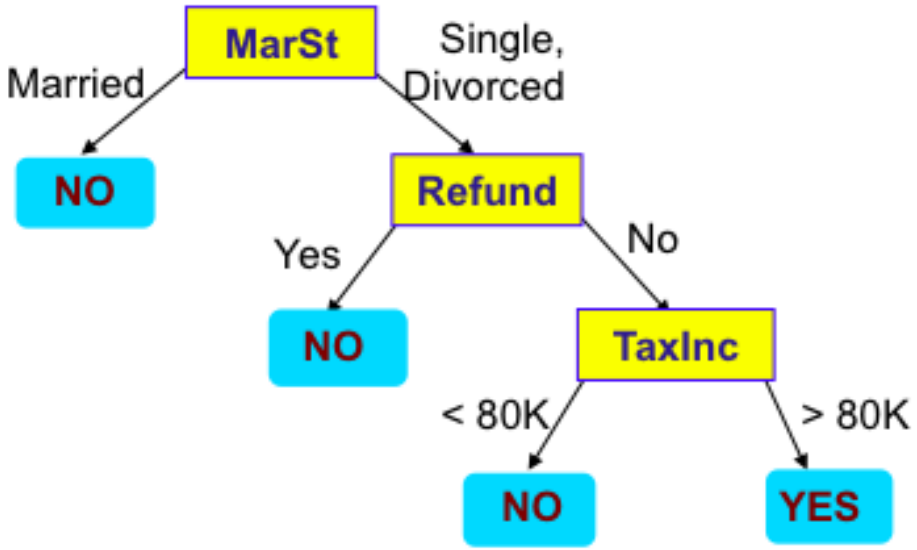


Model: Decision Tree

Otro ejemplo de Árbol de Decisión

Tid	Refund	Marital Status	Taxable Income	Cheat
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

categorical categorical continuous class



There could be more than one tree that fits the same data!

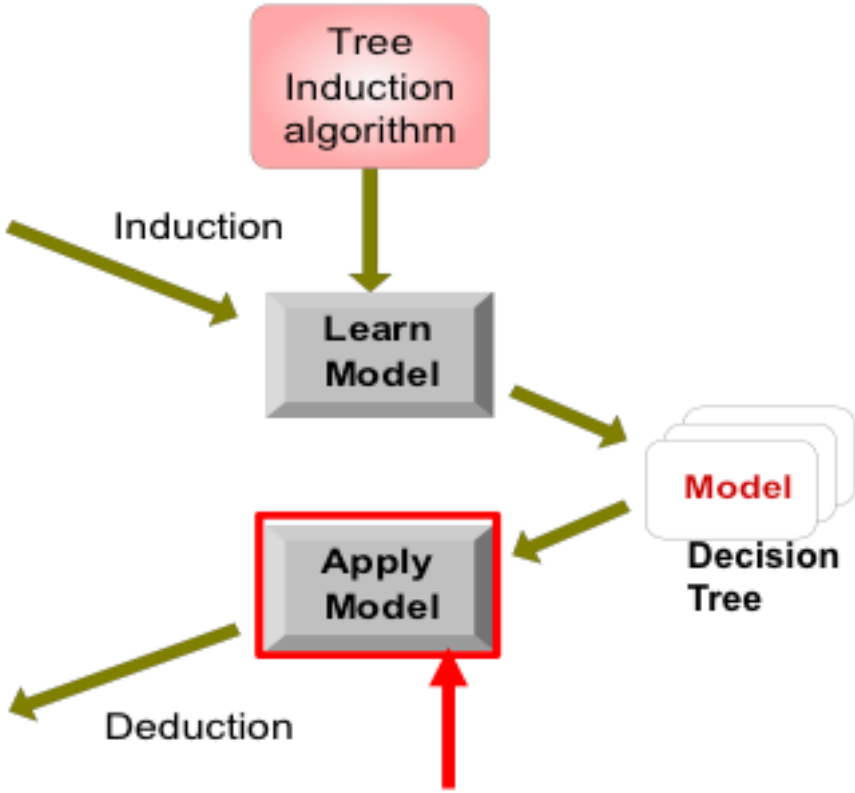
Construcción de un Árbol de Decisión

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

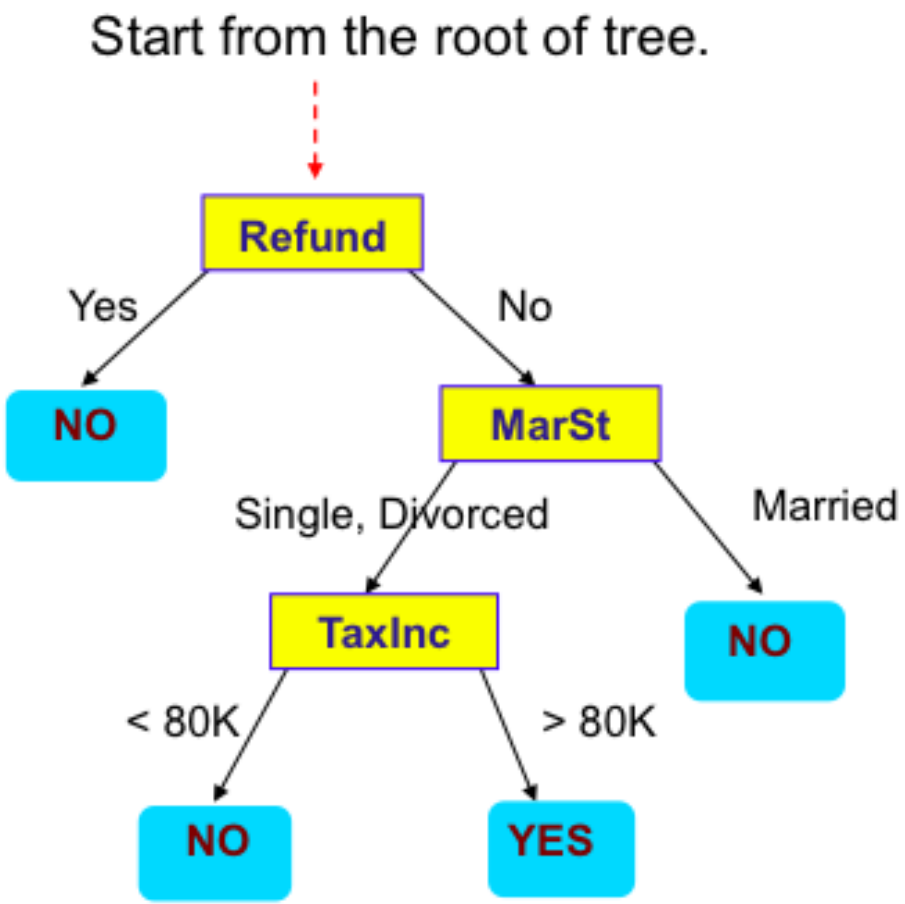
Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



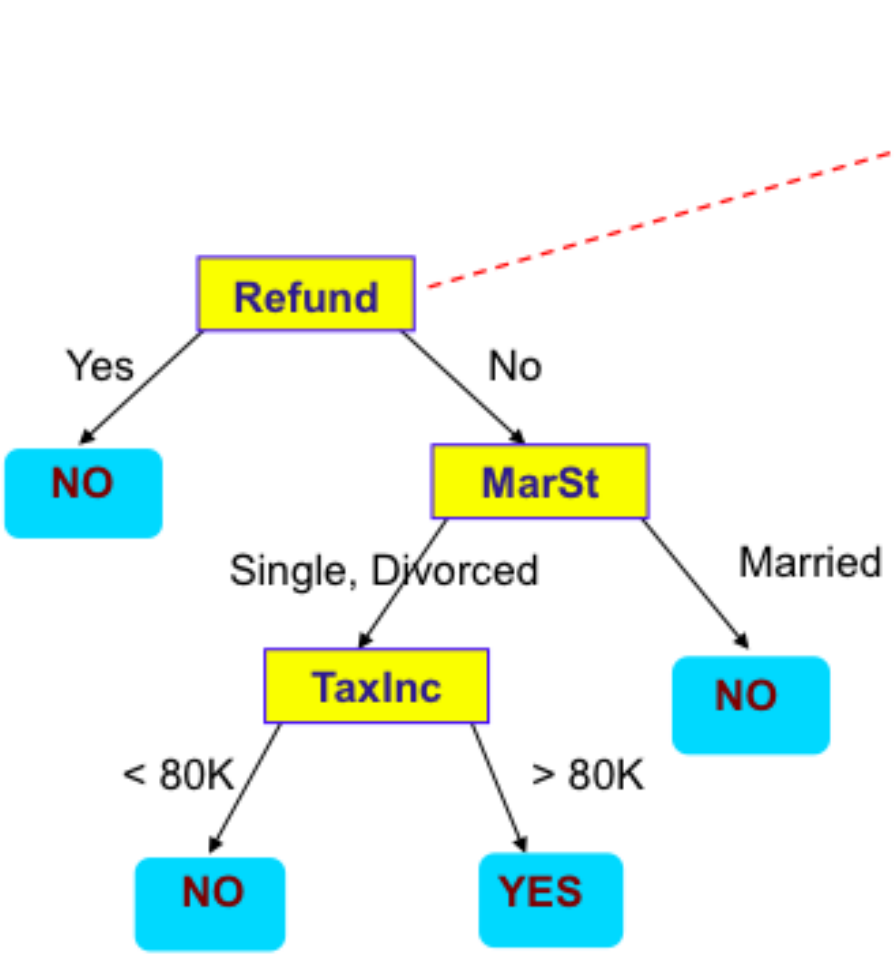
Aplicación del Modelo a Datos de Prueba



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

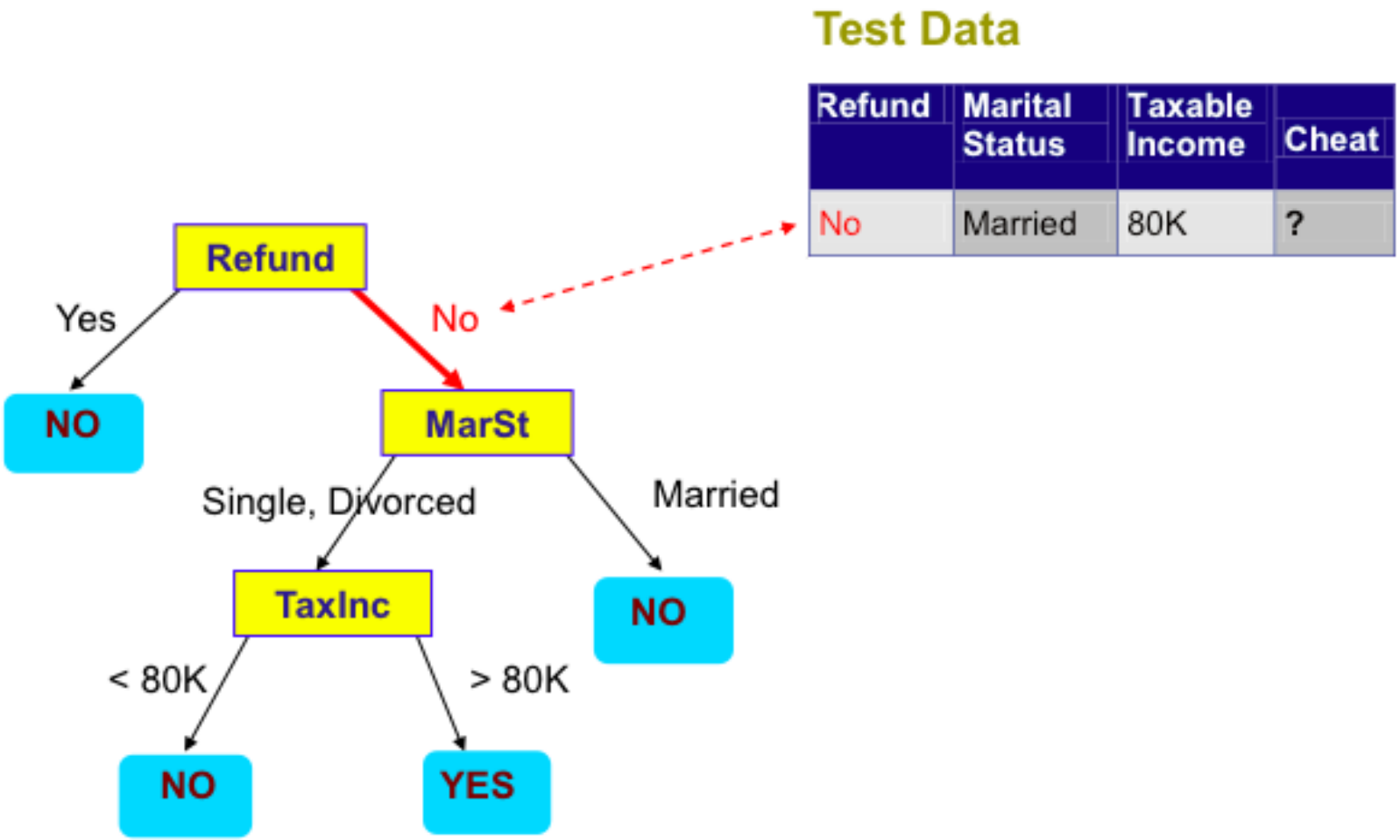
Aplicación del Modelo a Datos de Prueba



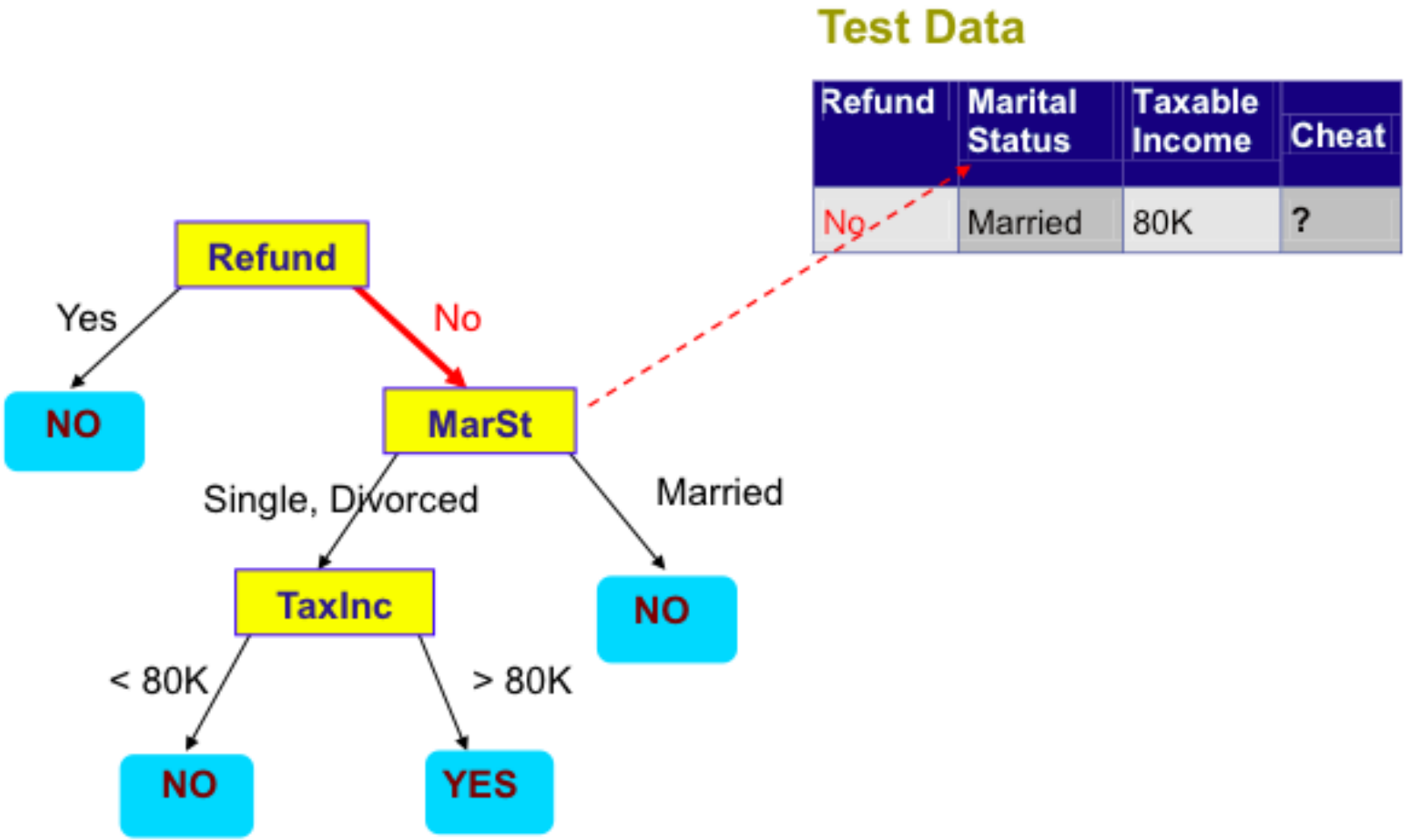
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

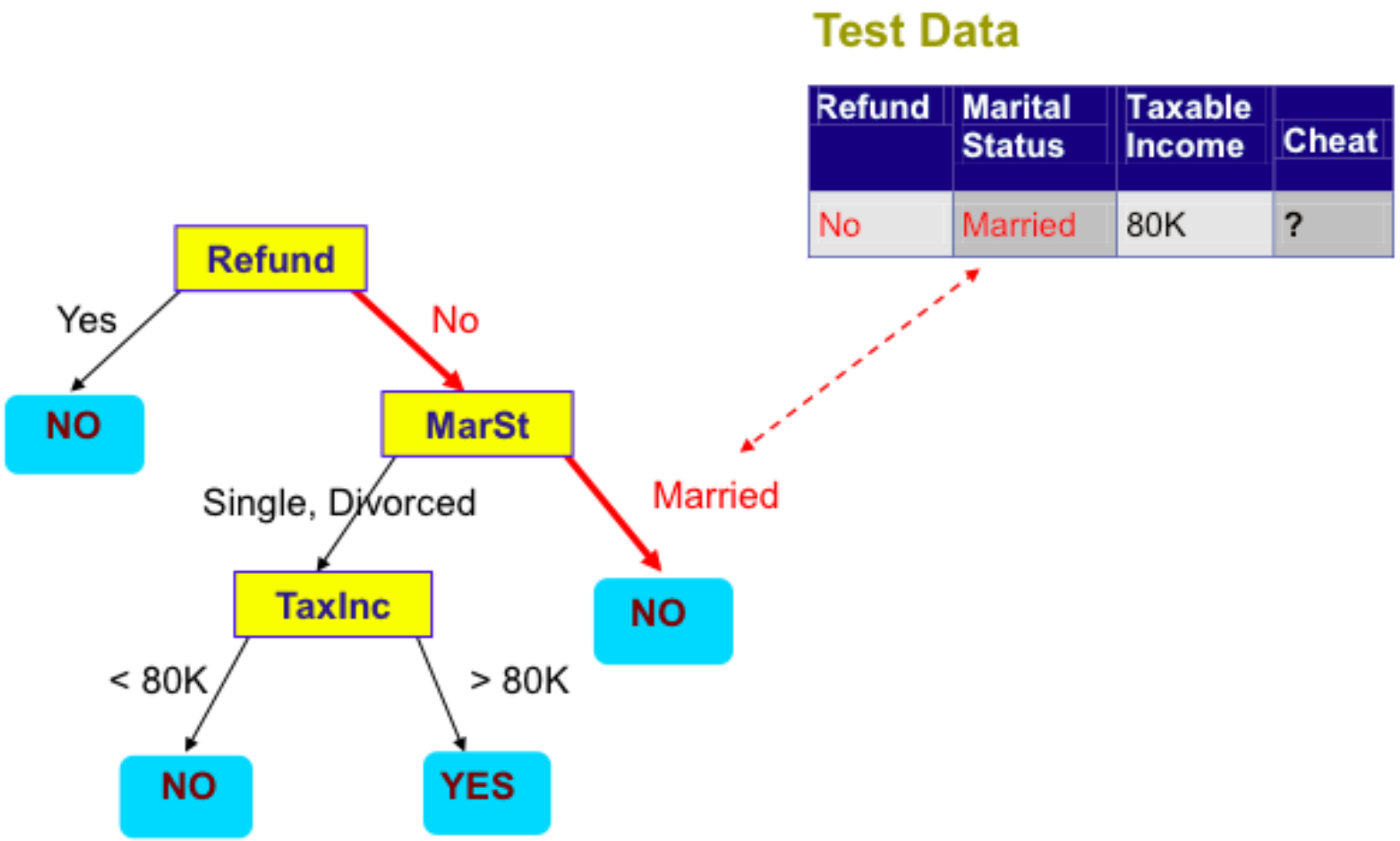
Aplicación del Modelo a Datos de Prueba



Aplicación del Modelo a Datos de Prueba



Aplicación del Modelo a Datos de Prueba



Características I

Los Árboles de Decisión son atractivos de usar en Minería de datos debido a que:

- Se usan mucho por su interpretabilidad
- Para evaluar que variables son importantes y como ellas interactúan una con otra.
- Los resultados se pueden expresar fácilmente mediante reglas.
- Son robustos a los outliers.
- Los algoritmos recursivos tienen una forma especial de tratar los valores faltantes: como un nivel aparte de la variable objetivo.
- Los valores faltantes pueden ser agrupados con otros valores y ser tratados en un nodo.
- Pueden realizarse imputaciones.

¿Por qué usar Árboles de Decisión? I

Los Árboles de Decisión son atractivos de usar en Minería de datos debido a que:

- Tienen una representación intuitiva, fácil de ser asimilado por humanos.
- Pueden ser rápidamente construidos a diferencia de otros métodos.
- La precisión de los clasificadores basados en árboles es comparable o superior a otros modelos.

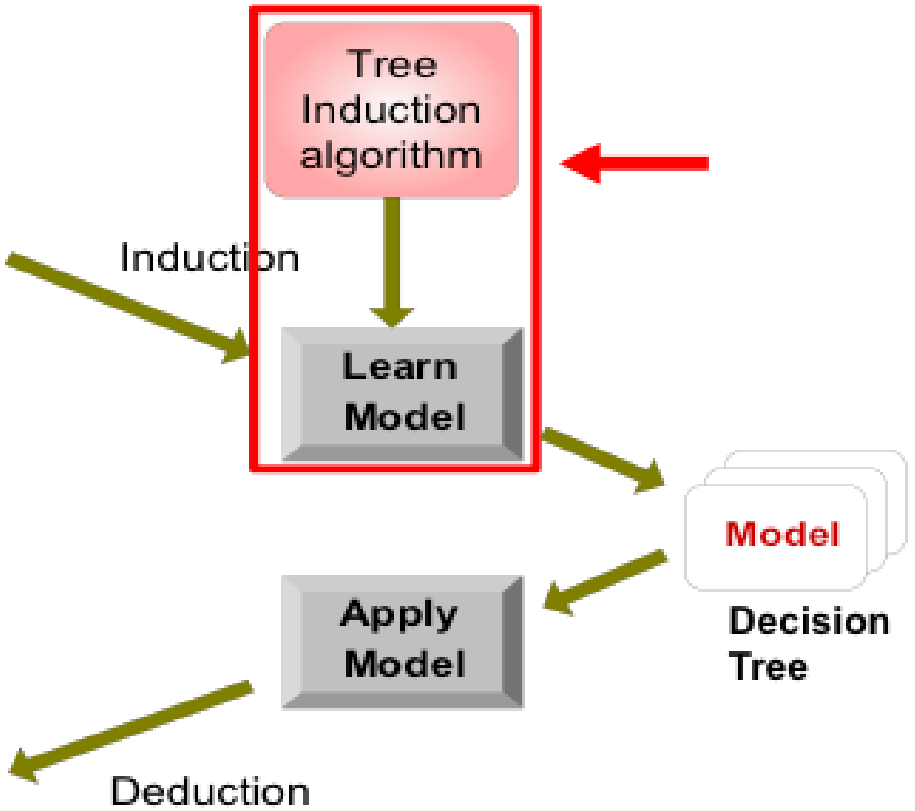
Construcción del Modelo

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Inducción en Árboles de Decisión I

La generación de un árbol de decisión consiste de dos fases

- Construcción del Árbol
  - Al inicio, todas las observaciones de entrenamiento están en la raiz.
  - Se realizan particiones recursivas basadas en los atributos seleccionados.
- Poda del Árbol
  - Identifican y remueven ramas que causen ruido o tengan outliers.

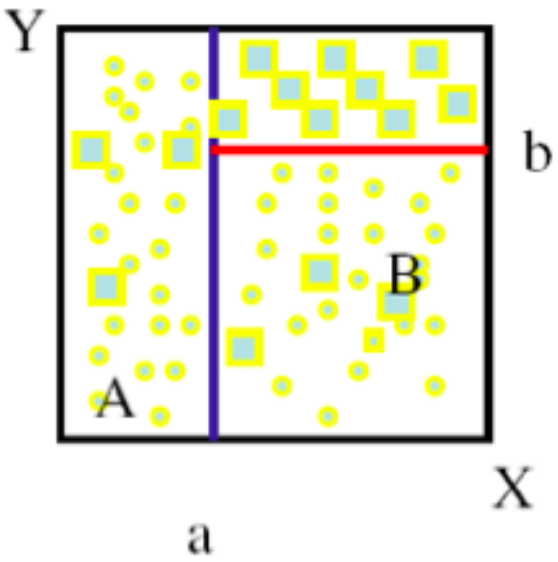
Construcción I

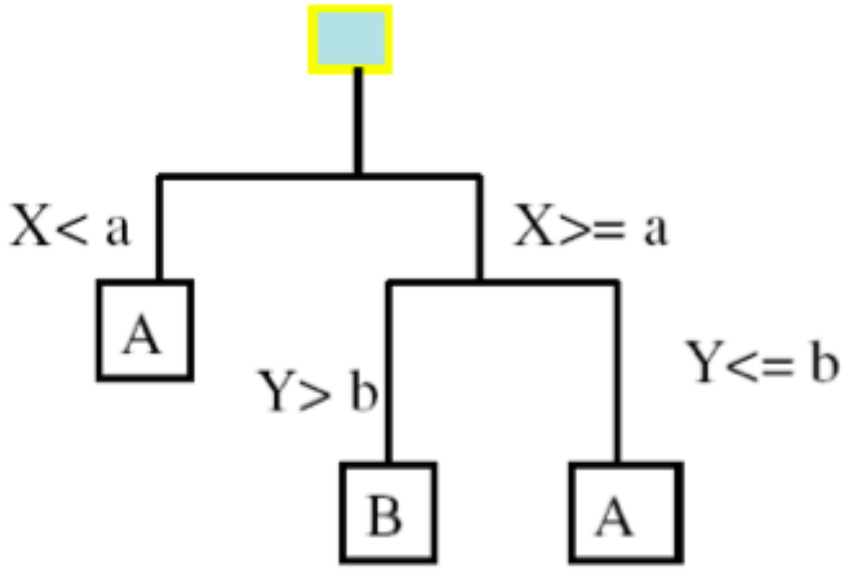
Un árbol de decisión particiona el espacio de variables predictoras en un conjunto de hiper-rectángulos y en cada uno de ellos ajusta un modelo sencillo, generalmente una constante. Es decir, $y = c$, donde y es la variable de respuesta. La construcción de un árbol de decisión se basa en cuatro elementos

- Un conjunto de preguntas binarias Q de la forma $\{x \in A\}$
- El método usado para particionar los nodos.
- La estrategia requerida para el crecimiento del árbol.
- La asignación de cada nodo terminal a una clase de la variable respuesta.

Las diferencias entre los algoritmos para construir árboles se hallan en la regla para particionar los nodos, la estrategia para podar los árboles, y el tratamiento de valores perdidos ("missing values")

Construcción del Modelo





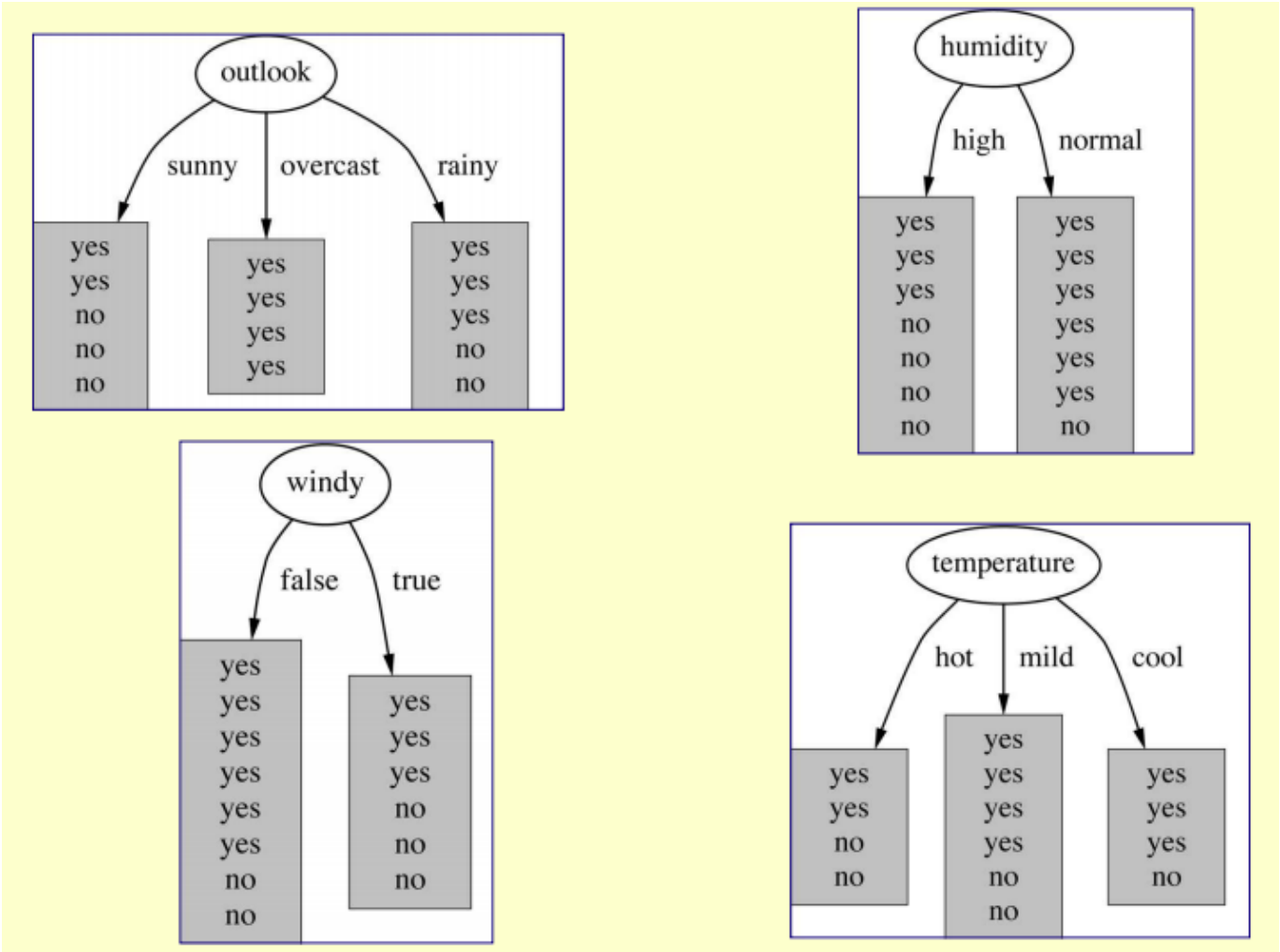
Múltiples Árboles de Decisión I

- Si los atributos son adecuados, es posible construir un árbol de decisión que clasifica correctamente a todas las observaciones de entrenamiento.
- Pueden existir muchos árboles de decisión correctos.
- Muchos algoritmos, elegir el árbol más simple (Navaja de Occam)
  - El principio establece que no se deben realizar más asunciones que el mínimo necesario.
  - El árbol más simple captura la máxima generalización y representa las relaciones más esenciales.

Ejemplo de Play Tennis

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

¿Cuál atributo seleccionar?



Inducción I

Estrategia:

- Dividir los registros basados en un atributo de prueba que optimice cierto criterio.

Discusión:

- Determinar como dividir los registros
  - ¿Cómo especificar la condición a evaluar?
  - ¿Cómo determinar la mejor partición?
- Determinar el criterio de parada.

Especificación de la condición a evaluar I

Depende del tipo de atributo

- Nominal
- Ordinal
- Continuo

Depende del número de divisiones:

- Binaria
- Múltiple

Selección del Atributo I

Muchas variantes:

- Machine Learning: ID2 (Iterative Dichotomizer), C4.5 (Quinla 86, 93)
- Estadística: CART (Classification and Regression Trees) (Breiman et al 84)
- Reconocimiento de patrones: CHAID (Chi-squared Automated Interaction Detection) (Magidson 94)

Principal diferencia: Criterio de división

- ¿Qué atributo probar en cada nodo del árbol? El atributo que es más útil para la clasificación

Medidas de Impureza

- **Coeficiente de Gini** Para el nodo t y con J clases

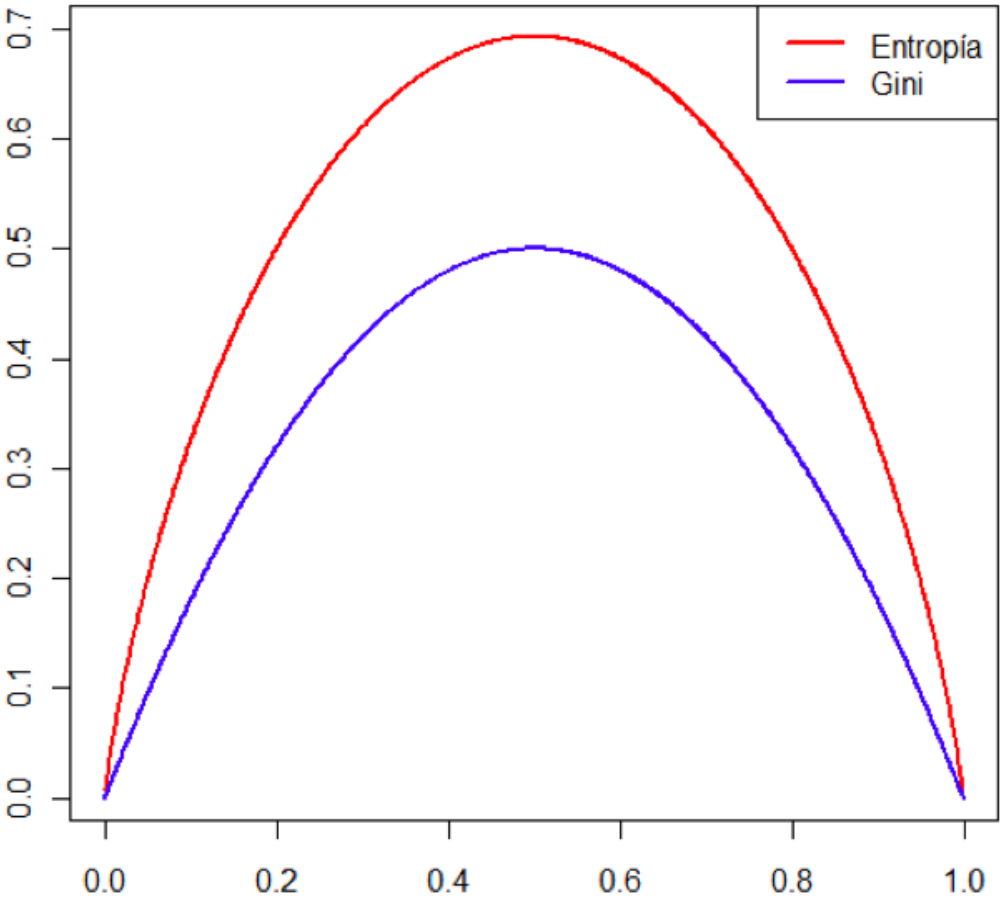
$$i_G(t) = \sum_{j=1}^{J} p(j \mid t)\,[1 - p(j \mid t)]$$

$$= 1 - \sum_{j=1}^{J} p(j \mid t)^2$$

- **Entropía** Para el nodo t y con J clases

$$i_E(t) = -\sum_{j=1}^{J} p(j \mid t) \log [p(j \mid t)]$$

Medidas de Impureza



Impureza del Árbol

$$I(T) = \sum_{t \in T} i(t)p(t)$$

donde T es el conjunto de nodos terminales del árbol y $p(t)$ es la probabilidad que un caso esté en el nodo t..

Ejemplo: Cálculo de Impureza

- Sin hacer ninguna partición tenemos que 24 alumnos aprobaron (clase P) y 8 alumnos desaprobaron (clase F)

P	24
F	8

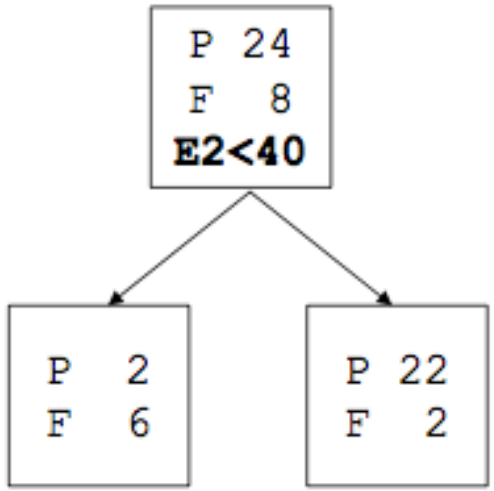
$$i_G(t) = 1 - \left(\frac{24}{32}\right)^2 - \left(\frac{8}{32}\right)^2 = 0.375$$

$$i_E(t) = \frac{24}{32}\log\left(\frac{24}{32}\right) + \frac{8}{32}\log\left(\frac{8}{32}\right) = 0.5623$$

Ejemplo: Cálculo de Impureza

- **Partición 1:** E2<40

$$i_G(1) = 0.3750 \quad i_G(2) = 0.1528$$
$$i_E(1) = 0.5623 \quad i_E(2) = 0.2868$$
$$p(1) = 0.25 \quad p(2) = 0.75$$



Nodo Terminal 1 Nodo Terminal 2

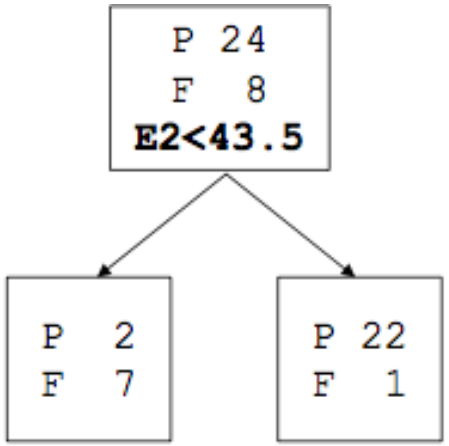
- **Impureza del árbol**

$$I_G(T) = 0.2083$$
$$I_E(T) = 0.3557$$

Ejemplo: Cálculo de Impureza

- **Partición 2:** `E2<43.5`

$$i_G(1) = 0.3457 \quad i_G(2) = 0.0832$$
$$i_E(1) = 0.5297 \quad i_E(2) = 0.1788$$
$$p(1) = 0.28125 \quad p(2) = 0.71875$$



- **Impureza del árbol**

$$I_G(T) = 0.1570$$
$$I_E(T) = 0.2775$$

- **Se obtiene menor impureza con esta partición**

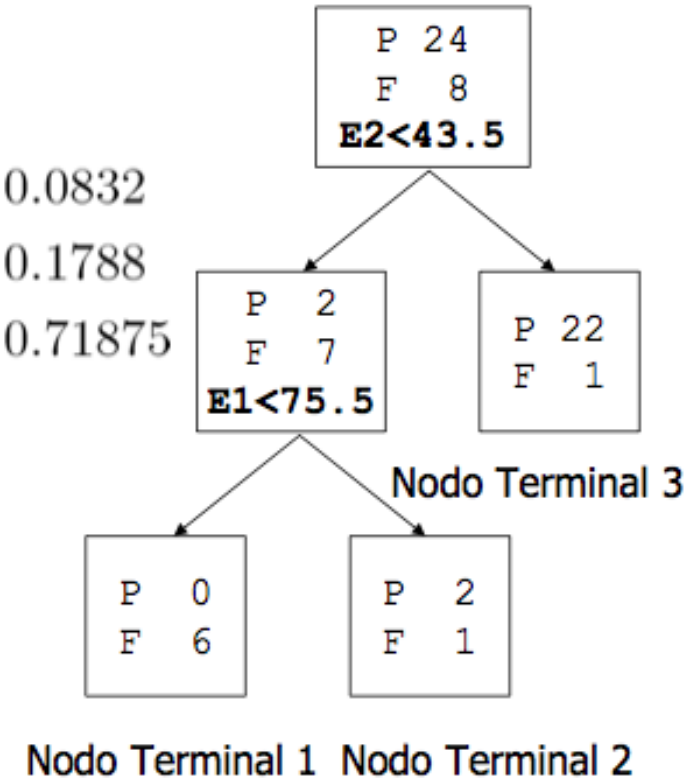
Ejemplo: Cálculo de Impureza

- **Impureza por nodo**

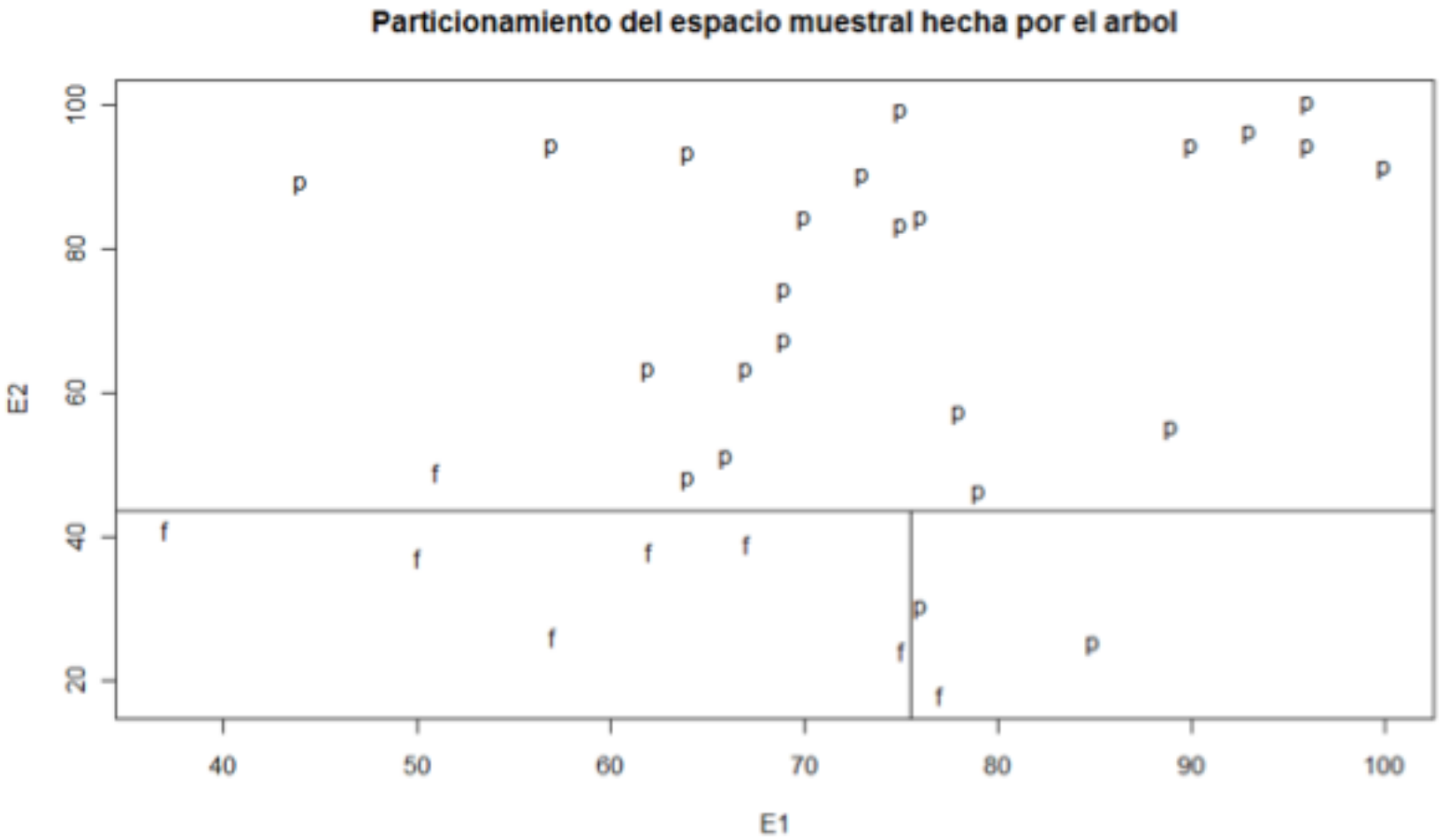
$$i_G(1) = 0 \qquad i_G(2) = 0.4444 \qquad i_G(3) = 0.0832$$
$$i_E(1) = 0 \qquad i_E(2) = 0.6365 \qquad i_E(3) = 0.1788$$
$$p(1) = 0.1875 \qquad p(2) = 0.09375 \qquad p(3) = 0.71875$$

- **Impureza del árbol**

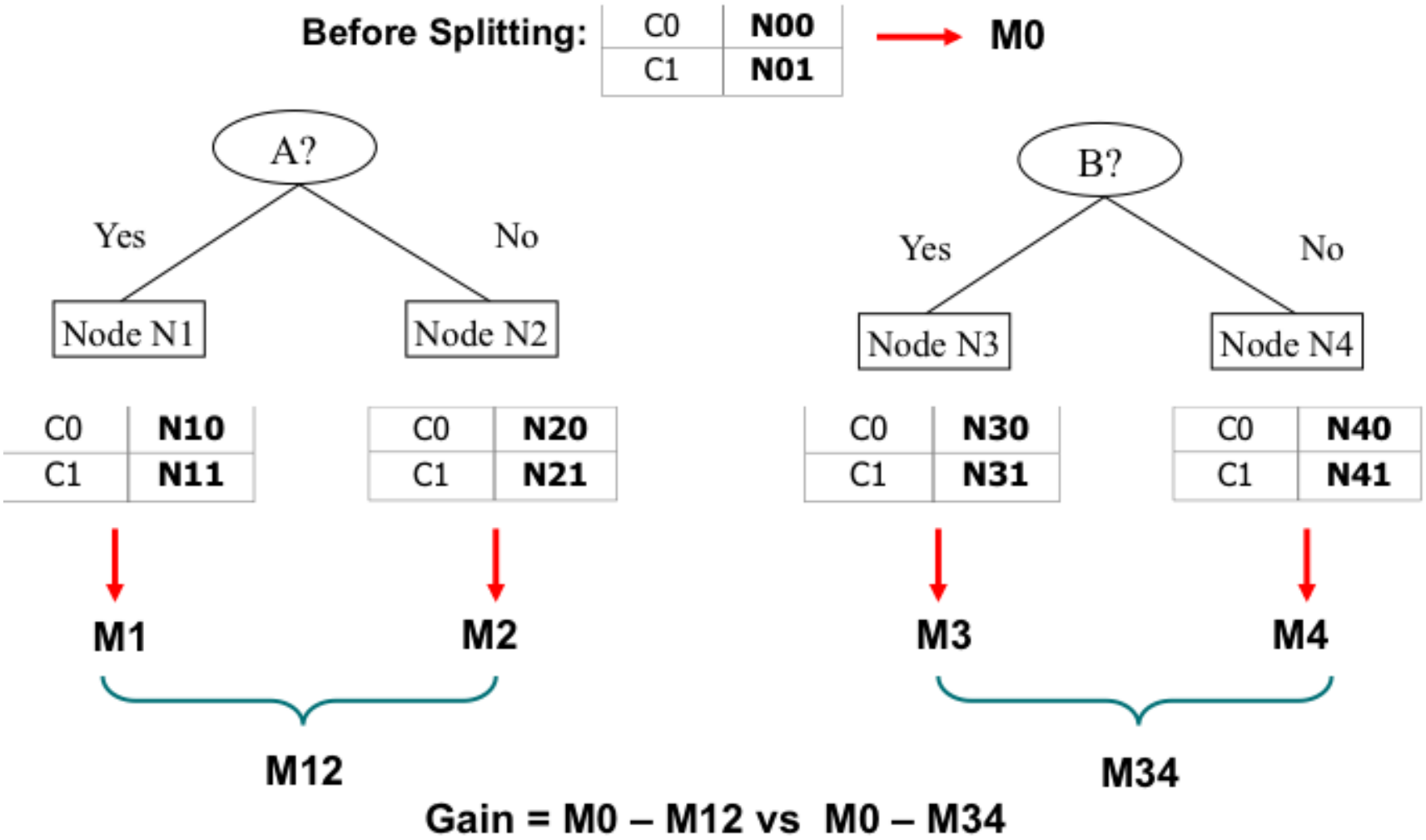
$$I_G(T) = 0.1014$$
$$I_E(T) = 0.1882$$



Ejemplo: Cálculo de Impureza



Mejor Partición



La Librería `rpart`

- En el `R` se puede utilizar la librería `rpart` con la función `rpart` cuya sintaxis es:

  ```
  rpart(formula, data,method)
  ```

- Argumentos:
  - formula: formula indicando las variable respuesta y las predictoras.
  - data: conjunto de datos a ser utilizado.
  - se usa ‘‘`class`’’ para árboles de decisión.

Opciones de control derpart

- **minsplit**: fija el número mínimo de observaciones en un nodo para que este sea dividido. Esta opción por defecto es 20.
- **minbucket**: indica el número mínimo de observaciones en cualquier nodo terminal. Por defecto esta opción es el valor redondeado de minsplit/3.
- **cp**: parámetro de complejidad. Indica que si el criterio de impureza no es reducido en mas de cp*100 % entonces se para. Por defecto cp=.01. Es decir, la reducción en la impureza del nodo terminal debe ser de al menos 1 % de la impureza inicial.
- **maxdepth**: condiciona la profundidad máxima del arbol. Por defecto está establecida como 30.

La Librería rpart

- Para graficar el árbol se utiliza:

```
plot(objeto,margin=0.25)
text(objeto,use.n=T)
```

- Argumentos:
  - objeto: salida de rpart.
  - margin: margen del gráfico del árbol.
  - use.n : si es T adiciona al gráfico cuantos elementos hay de cada clase.

La Librería `rpart`

- Para realizar predicciones se utiliza:

  `predict(object,newdata,type)`

- Argumentos:
  - predict(object,newdata,type)
  - newdata: conjunto de datos con los cuales se va predecir
  - type: tipo de salida (‘‘`class`’’ retorna la clase, ‘‘`prob`’’ retorna matriz de probabilidades de las clases)

Ejemplo BUPA

- 345 instancias
- 7 atributos
  - V1 volumen corpuscular
  - V2 fosfatosa alcalina
  - V3 alamine aminotransferase
  - V4 aspartate aminotransferase
  - V5 gamma-glutamyl transpeptidase
  - V6 número de bebidas alcohólicas
  - V7 1(hígado enfermo) 2(hígado sano)

Conjunto de datos BUPA

```
# Leer conjunto de datos
bupa<-read.table("datos/bupa.txt",header=T,sep=",")

# Declarar V7 como un factor
bupa[,7]<-as.factor(bupa[,7])

# Cargar libreria rpart
library(rpart)
```

Ejemplo BUPA

```
# Ejemplo 1: considerando minbucket=50 (minsplit=150)
#Estimar el árbol
arbol1=rpart(V7~V3+V5,data=bupa,method="class",
             minbucket=50)
```

Árboles de Decisión

```
arbol1

node), split, n, loss, yval, (yprob)
      * denotes terminal node
1) root 345 145 2 (0.4202899 0.5797101)
  2) V5< 20.5 140  61 1 (0.5642857 0.4357143) *
  3) V5>=20.5 205  66 2 (0.3219512 0.6780488) *
```

Estructura del Árbol

- Esta salida nos presenta la estructura del árbol:

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node
```

- node : número del nodo
- split : punto de corte
- n : número de observaciones en el nodo
- loss : número de clasificados incorrectamente
- yval : la clase para el nodo
- (yprob): probabilidad de cada clase
- * : denota que es un nodo terminal

Ejemplo BUPA: Estructura del Árbol

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node
1) root 345 145 2 (0.4202899 0.5797101)
```

- Es el nodo raíz, representa a todas las 345 observaciones.
- La clase mayoritaria es 2 (hígado sano)
- El valor de 145 nos indica cuantas de las 345 serian incorrectamente clasificadas.
- El valor yprob nos indica que el 42 % de las observaciones corresponden a un hígado enfermo y el 58 % a uno sano.

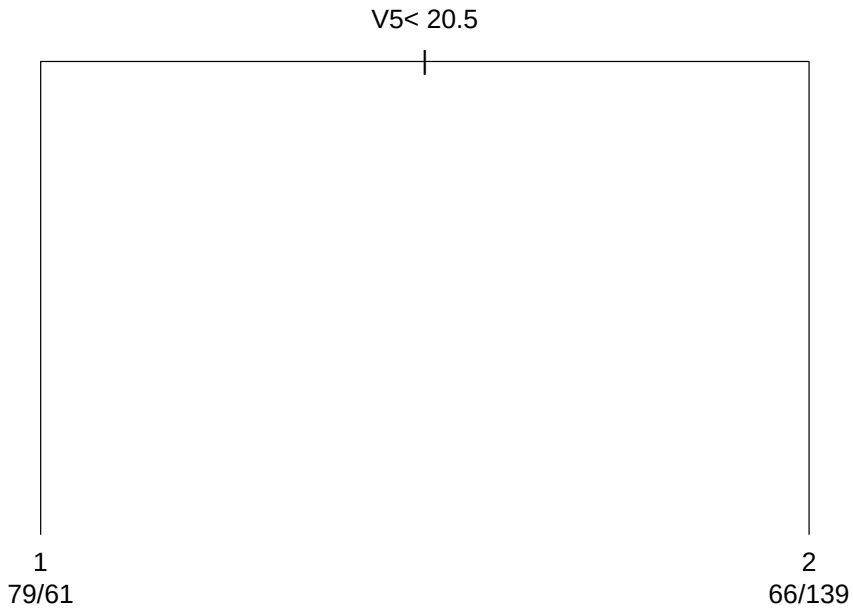
Ejemplo BUPA: Estructura del Árbol

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node
1) root 345 145 2 (0.4202899 0.5797101)
  2) V5< 20.5 140  61 1 (0.5642857 0.4357143) *
  3) V5>=20.5 205  66 2 (0.3219512 0.6780488) *
```

- El nodo raíz ha sido dividido en dos subnodos. La división se basa en la variable V5 y el punto de corte es de 20.5
- En el nodo 2 el punto de corte es dado por V5¡ 20.5. De las 140 observaciones el 43.6 % estrían mal clasificadas.
- El nodo 3 contiene a las restantes 205 observaciones. En este caso la decisión es clasificar como 2(hígado sano). Hay un 32.2 % de error en este caso.

Ejemplo BUPA: Estructura del Árbol

```
# Graficando el arbol
plot(arbol1,margin=.25)
text(arbol1,use.n=T)
```

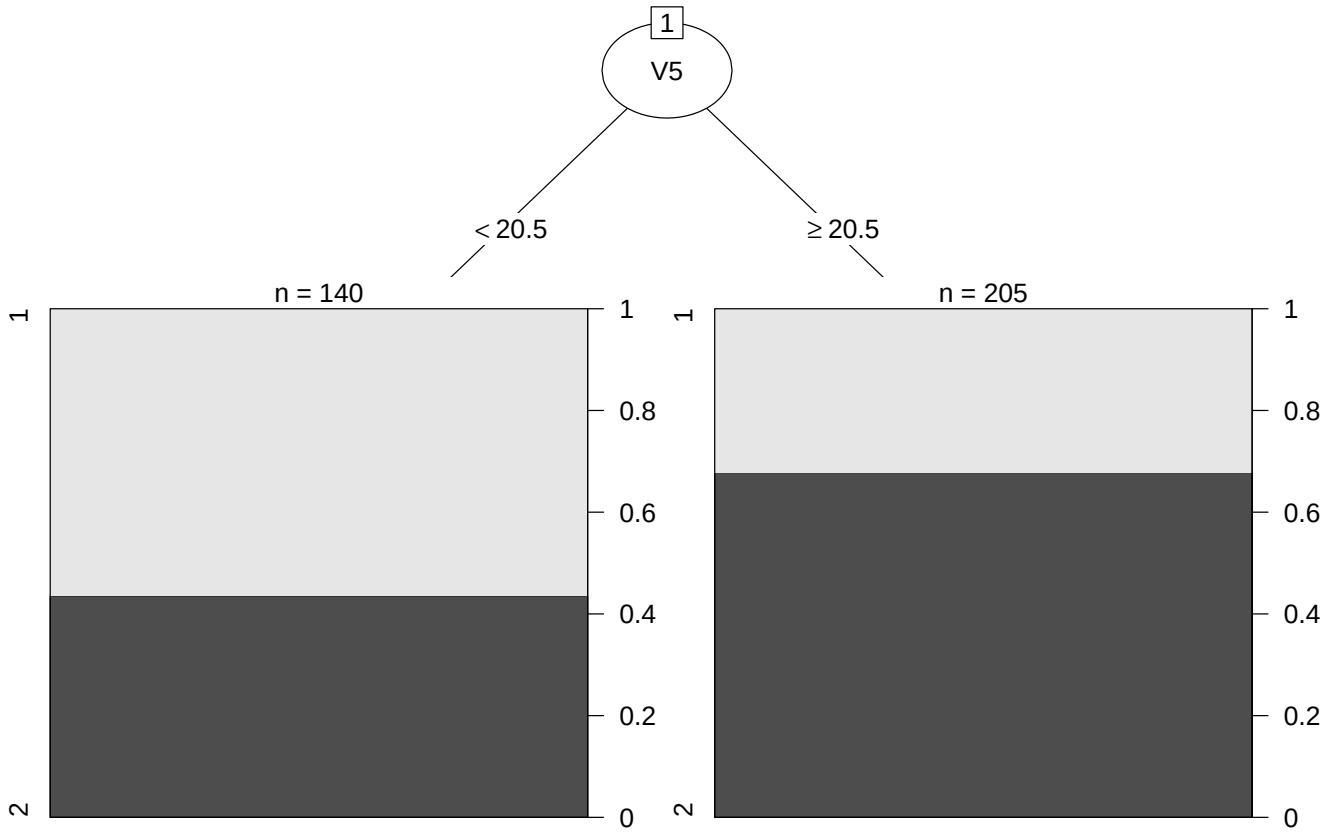


Gráficos para Árboles de Decisión

- Para obtener mejores gráficos se pueden utilizar las librerías:
  - `partykit`
  - `rattle`

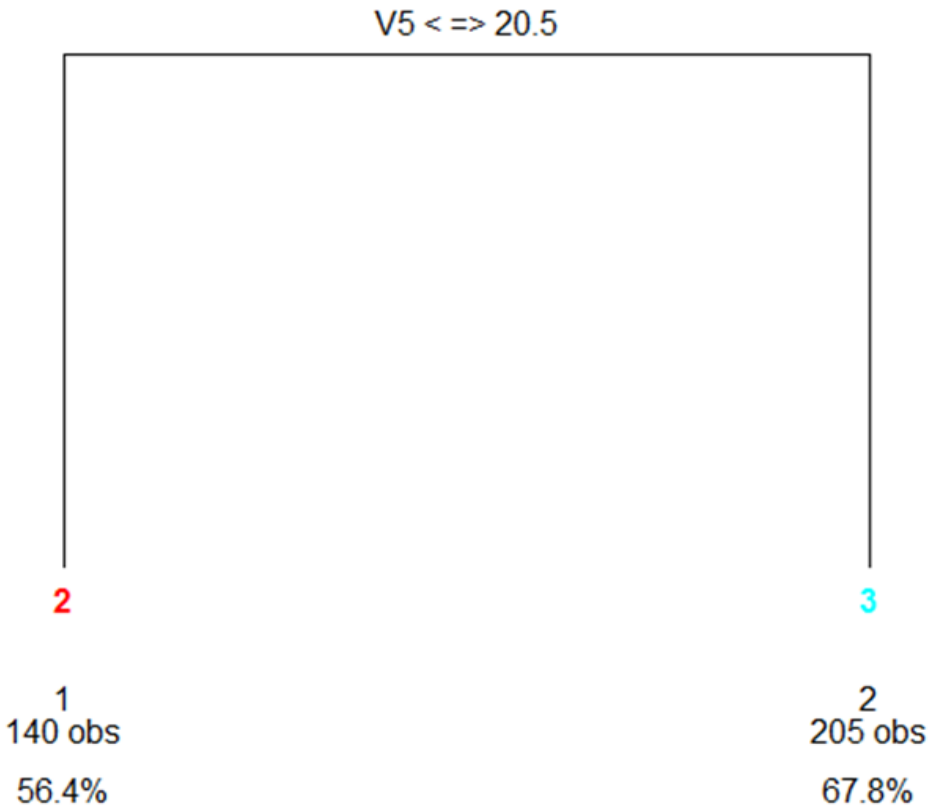
Ejemplo BUPA: Gráficos

```
library(partykit)
plot(as.party(arbol1), tp_args = list(id = FALSE))
```



Ejemplo BUPA: Gráficos

```
library(rattle)
drawTreeNodes(arbol1)
```

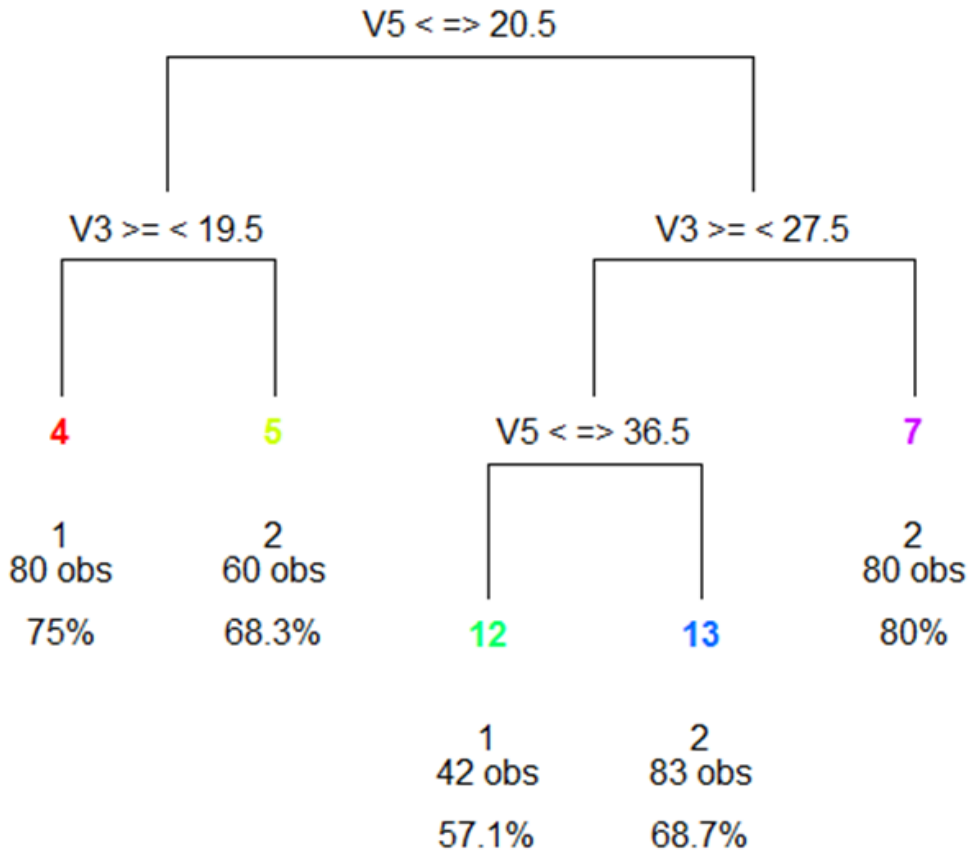


Árboles de Decisión

- Ejemplo 2: minbucket=20 (minsplit=60) para obtener un árbol con más ramas.

```
arbol2=rpart(V7~V3+V5,data=bupa,method="class",
minbucket=20)
drawTreeNodes(arbol2)
```

Árboles de Decisión

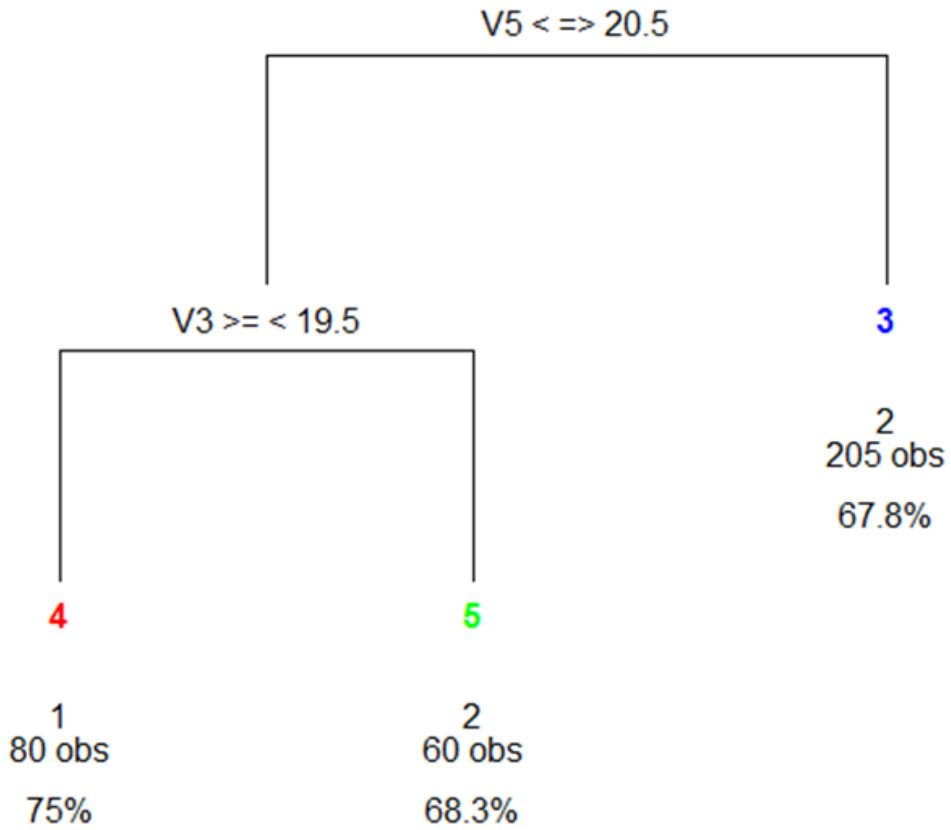


Árboles de Decisión

- Ejemplo 3: Controlando el crecimiento del árbol con el parámetro de complejidad (cp=0.05).

```
arbol3=rpart(V7~V3+V5,data=bupa,method="class", cp=0.05)
drawTreeNodes(arbol3)
```

Árboles de Decisión

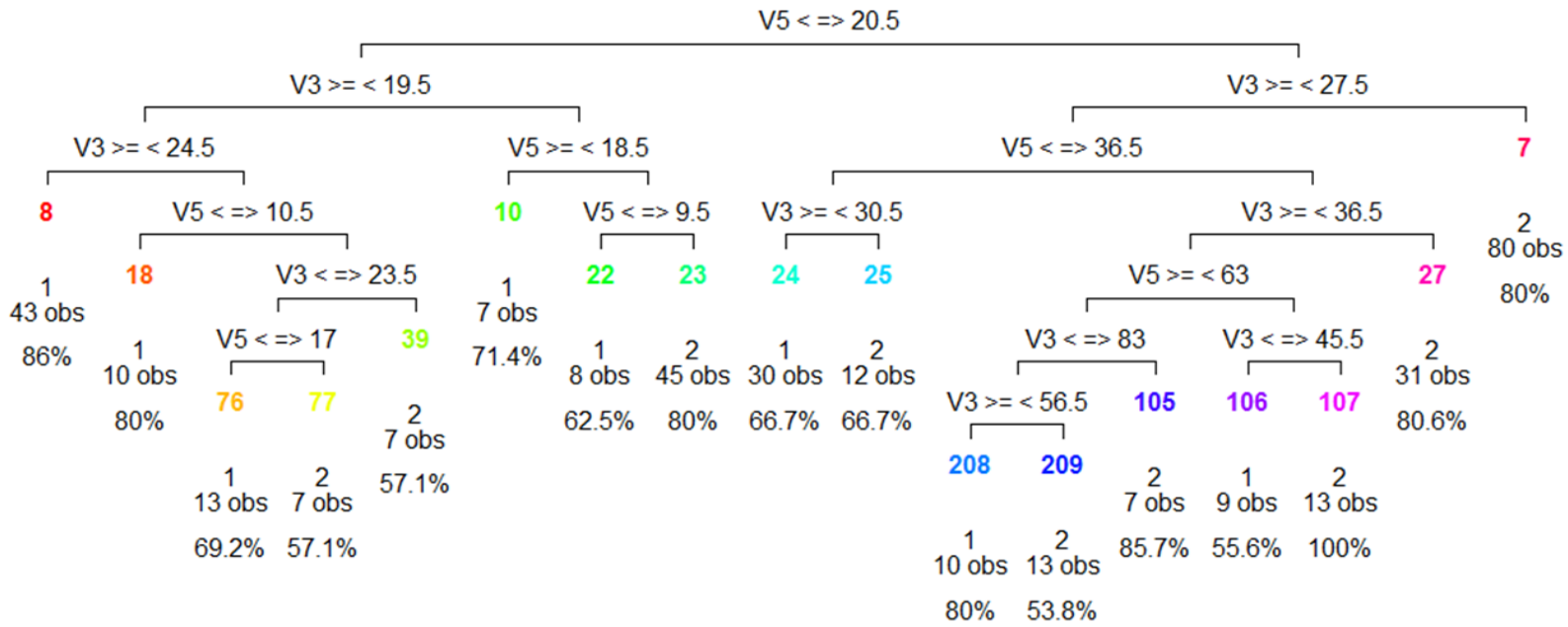


Árboles de Decisión

- Ejemplo 4: cp=0.001 para obtener un árbol con más ramas.

```
arbol4=rpart(V7~V3+V5,data=bupa,method="class", cp=0.001)
drawTreeNodes(arbol4)
```

Árboles de Decisión

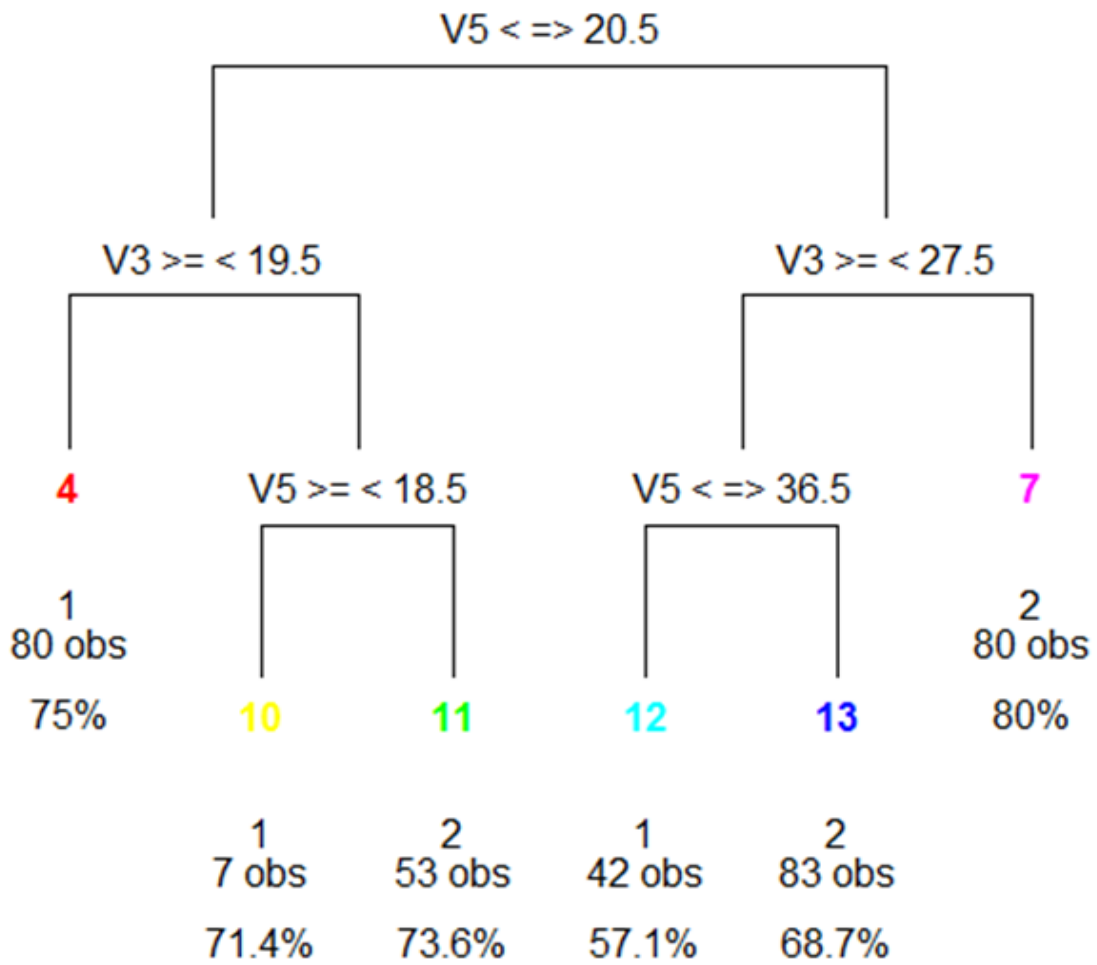


Árboles de Decisión

- Ejemplo 5: Controlando el crecimiento del árbol por número máximo de niveles (maxdepth=3).

```
arbol5=rpart(V7~V3+V5,data=bupa,method="class",
maxdepth=3)
drawTreeNodes(arbol5)
```

Árboles de Decisión

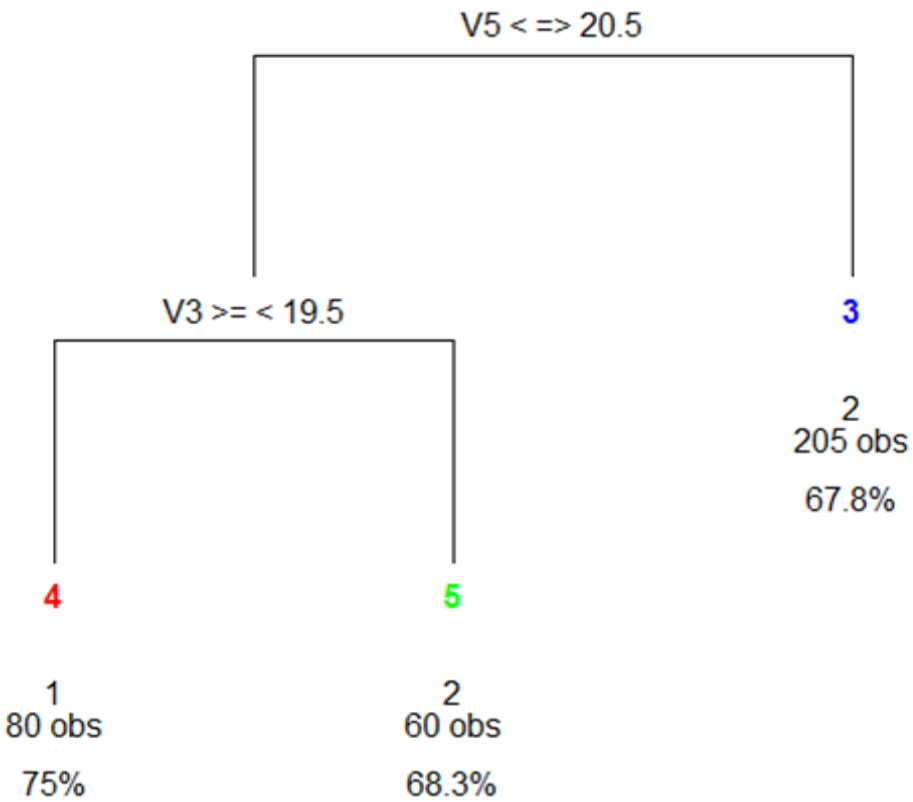


Recortar el árbol

- Hacer crecer un árbol demasiado puede crear problemas de sobreajuste (overfitting).
- La función `prune` de la librería `rpart` realiza el recorte de un arbol.
- La opción `cp`, denominado parámetro de complejidad, indica que se descartará cualquier partición que no disminuya la impureza por un factor de `cp`.

```
arbol6=prune(arbol4,cp=.1)
drawTreeNodes(arbol6)
```

Recortar el árbol



Recortar el árbol

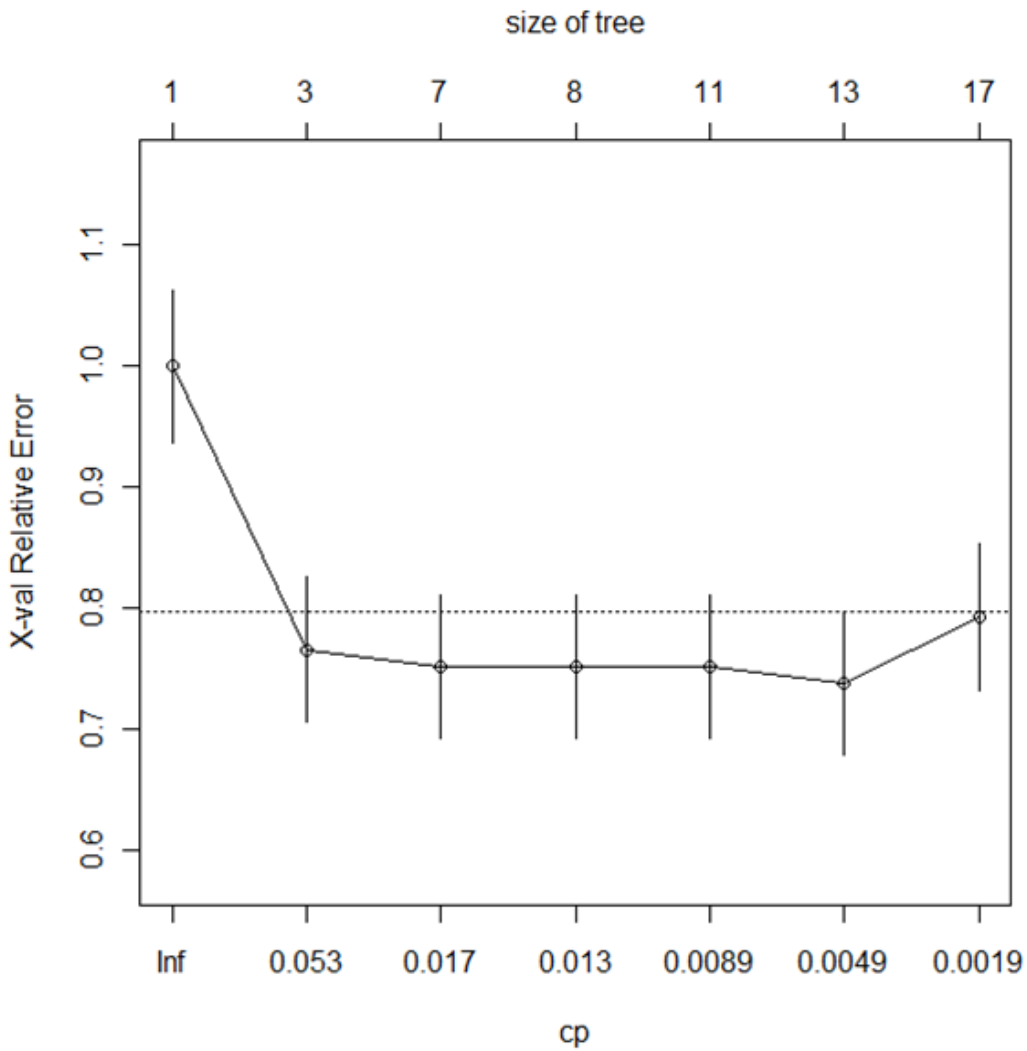
- Para decidir que parámetro de complejidad escoger se utiliza las funciones:

```
printcp(arbol4)
Root node error: 145/345 = 0.42029

          CP nsplit rel error  xerror     xstd
1 0.1379310      0   1.00000 1.00000 0.063230
2 0.0206897      2   0.72414 0.76552 0.059840
3 0.0137931      6   0.63448 0.75172 0.059551
4 0.0114943      7   0.62069 0.75172 0.059551
5 0.0068966     10   0.58621 0.75172 0.059551
6 0.0034483     12   0.57241 0.73793 0.059252
7 0.0010000     16   0.55862 0.79310 0.060386
```

Recortar el árbol

```
plotcp(arbol4)
```



Predicción

- Para realizar predicciones utilizando el arbol6:

```
#Calcular los valores predichos
pred<-predict(arbol6,bupa[,c(3,5)],type="class")

#Calcular el error de mala clasificación
error=mean(pred!=bupa$V7)
error
[1] 0.3043478

#Calcular la matriz de confusión
table(pred,bupa$V7)

 pred   1   2
   1  60  20
   2  85 180
```

Prioris y Matriz de pérdidas

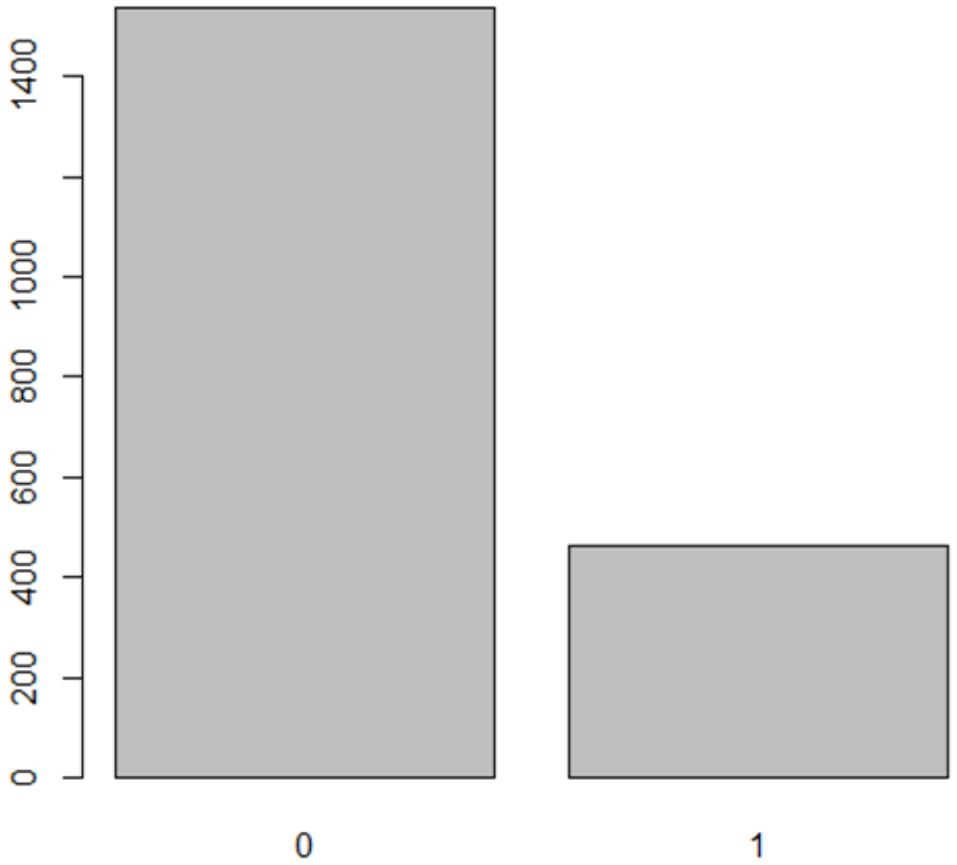
- En el caso que tengamos un conjunto de datos en que una clase este mal representada la función `rpart` permite incorporar probabilidades a priori y también una matriz de perdidas.

Ejemplo

```
library(rpart)
library(rattle)
library(partykit)

data(audit)
audit$TARGET_Adjusted<-as.factor( audit$TARGET_Adjusted)
barplot(table(audit$TARGET_Adjusted))
```

Ejemplo



Ejemplo

```
fit <- rpart(TARGET_Adjusted ~ .,data=audit[,-12],
method="class")
drawTreeNodes(fit)
plot(as.party(fit), tp_args = list(id = FALSE))

pred<-predict(fit,audit[,-c(12,13)],type="class")
# table(audit$TARGET_Adjusted,pred)
library(gmodels)
CrossTable(audit$TARGET_Adjusted,pred)
```

Ejemplo

audit$TARGET_Adjusted \ pred	0	1	Row Total
0	1432	105	1537
	29.866	124.480	
	0.932	**0.068**	**0.768**
	0.888	0.271	
	0.716	0.052	
1	181	282	463
	99.144	413.229	
	0.391	**0.609**	**0.232**
	0.112	0.729	
	0.090	0.141	
Column Total	1613	387	2000
	0.806	0.194	

Ejemplo

```
fit1 <- rpart(TARGET_Adjusted ~ .,data=audit[,-12],
parms=list(prior=c(.5,.5)))
drawTreeNodes(fit1)
plot(as.party(fit1), tp_args = list(id = FALSE))

pred<-predict(fit1,audit[,-c(12,13)],type="class")
# table(pred,audit$TARGET_Adjusted)
CrossTable(audit$TARGET_Adjusted,pred)
```

Ejemplo

audit$TARGET_Adjusted \ pred	0	1	Row Total
0	1022	515	1537
	43.244	51.073	
	0.665	**0.335**	**0.768**
	0.944	0.562	
	0.511	0.258	
1	61	402	463
	143.556	169.543	
	0.132	**0.868**	**0.232**
	0.056	0.438	
	0.030	0.201	
Column Total	1083	917	2000
	0.541	0.459	

Ejemplo

```
loss <- matrix(c(0, 1, 2, 0), byrow=TRUE, ncol=2)
loss
fit2 <- rpart(TARGET_Adjusted ~ ., data=audit[,-12],
parms=list(loss=loss))
drawTreeNodes(fit2)
plot(as.party(fit2), tp_args = list(id = FALSE))

pred<-predict(fit2,audit[,-c(12,13)],type="class")
# table(pred,audit$TARGET_Adjusted)
CrossTable(audit$TARGET_Adjusted,pred)
```

Ejemplo

pred			
audit$TARGET_Adjusted	0	1	Row Total
0	1395	142	1537
	31.818	113.139	
	0.908	**0.092**	**0.768**
	0.894	0.323	
	0.698	0.071	
1	166	297	463
	105.625	375.584	
	0.359	**0.641**	**0.232**
	0.106	0.677	
	0.083	0.148	
Column Total	1561	439	2000
	0.780	0.220	

Árboles por Inferencia Condicional I

- Uno de los principales problemas de los algoritmos tradicionales es que pueden sufrir de sobreajuste y pueden exhibir sesgo al seleccionar variables con muchas particiones posibles.
- Otro problema es que no usan ningún concepto de inferencia estadística que permita distinguir entre una ganancia significativa de la información medida (Mingers, 1989)
- De acuerdo a Hothorn, Hornik y Zeileis (2005) el método de árboles por inferencia condicional (implementado en la función ctree de la librería party) es insesgado en el sentido que no favorece a variables que tengan mayores puntos de corte, como si lo hacen los métodos usuales para construcción de árboles de decisión basados en la maximización de una medida de información como el RPART implementado en la función rpart de la librería rpart.

Árboles por Inferencia Condicional

Pasos:

- Hacer un test global de independencia entre cada variable independiente con la variable respuesta.
- Seleccionar la variable con la mayor asociación. Esta asociación es medida por el pvalor.
- Encontrar un punto de corte en la variable seleccionada.
- Repetir hasta que no se puedan abrir más nodos.

Árboles por Inferencia Condicional

Pasos:

- En el R esta implementado en la librería `party` con la función `ctree`.
- Aplicando al conjunto de datos bupa:

```
bupa$V7<- as.factor(bupa$V7)
library(party)
arbolc1<-ctree(V7~V3+V5,data=bupa)
arbolc1
```

Árboles por Inferencia Condicional

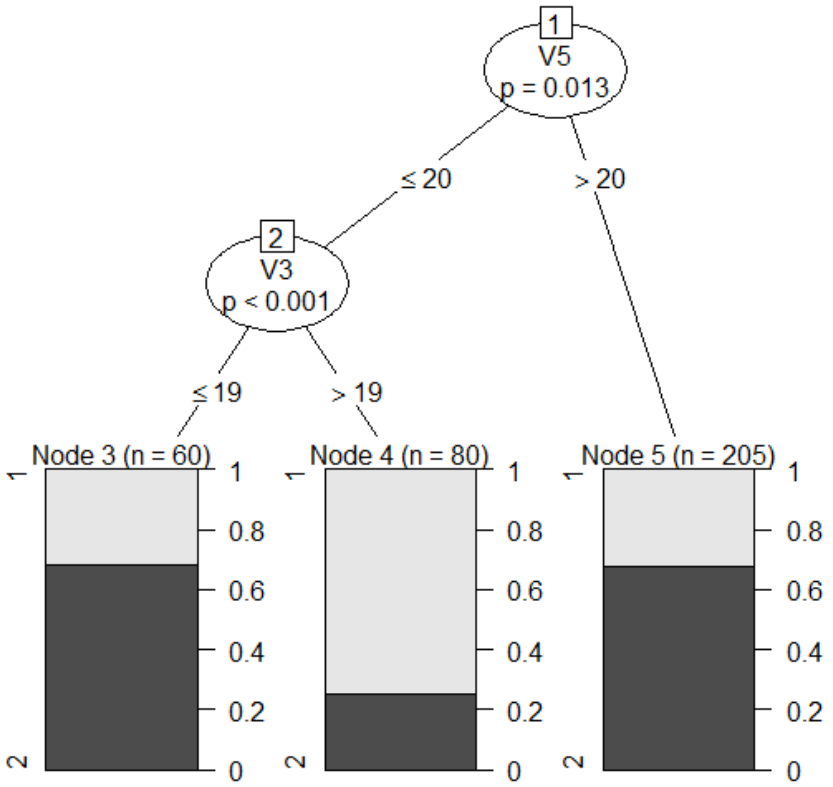
```
Conditional inference tree with 3 terminal nodes

Response:  V7
Inputs:  V3, V5
Number of observations:  345

1) V5 <= 20; criterion = 0.987, statistic = 7.372
  2) V3 <= 19; criterion = 1, statistic = 22.679
    3)*  weights = 60
  2) V3 > 19
    4)*  weights = 80
1) V5 > 20
  5)*  weights = 205
```

Árboles por Inferencia Condicional

```
plot(arbolc1)
```



Árboles por Inferencia Condicional

```
arbolc2<-ctree(V7~V3+V5,data=bupa,
controls = ctree_control(mincriterion = 0.70))
arbolc2
plot(arbolc2)
 Conditional inference tree with 4 terminal nodes

Response:  V7
Inputs:  V3, V5
Number of observations:  345

1) V5 <= 20; criterion = 0.987, statistic = 7.372
  2) V3 <= 19; criterion = 1, statistic = 22.679
    3)*  weights = 60
  2) V3 > 19
    4) V3 <= 24; criterion = 0.793, statistic = 2.563
      5)*  weights = 37
    4) V3 > 24
      6)*  weights = 43
1) V5 > 20
  7)*  weights = 205
```

Árboles por Inferencia Condicional