**Contextualitzeu els exemples de les següents preguntes respecte al projecte que heu definit a la PAC1. Si ho desitgeu, pots podeu redefinir o afinar el projecte.**

1. **Creieu que els arbres de decisió són el mètode més adequat per aconseguir els objectius que us havíeu proposat? Justifiqueu la resposta tot raonant-la.**

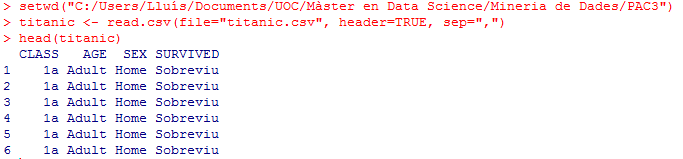
**Com podria ser l'arbre resultant?**

**Podríeu donar, tres exemples de regles que es poguessin derivar de l'arbre de decisió del vostre projecte?**

1. **En aquest exercici aneu a seguir els passos del cicle de vida d'un projecte de mineria de dades pel cas d'un algorisme de classificació i més concretament un arbre de decisió. Ho fareu amb el fitxer titanic.csv. Que trobareu a la wiki. Aquest fitxer conté un registre per cada passatger que viatjava en el Titànic. En les variables es caracteritza si era home o dona, adult o nen, en quina categoria viatjava o si era membre de la tripulació.**

* **Estudieu les dades, per exemple: Número de registres dels fitxer? Distribucions de valors per variables? Hi ha camps mal informats o buits?**
* **Prepareu les dades. En aquest cas ja estan en el format correcte i no cal discretitzar ni generar atributs nous. Cal triar quines són les variables que s'utilitzaran per construir el model i quina és la variable que classifica. En aquest cas la variable per la que classificarem és el camp de si el passatger va sobreviure o no.**
* **Instal·leu, si manca, el paquet C5.0 a R. Aquest paquet, documentat a la wiki , és una implementació moderna de l'algorisme ID3 de Quinlan. Té els principis teòrics de l'ID3 més la poda automàtica. Amb aquest paquet genereu un model de mineria.**
* **Quina és la qualitat del model?**
* **Genereu l'arbre gràfic**
* **Extraieu les regles del model**
* **En funció del model, l'arbre i les regles: Quin és el coneixement que trèiem?**
* **Proveu el model generat presentant-li nous registres. Classifica prou bé?**

Per començar, especifico la ruta al directori de treball. A continuació carrego el fitxer *titanic.csv* a l’entorn de R, en un dataframe amb nom *titanic*. Comprovo que s’ha carregat correctament amb la funció *head*, que mostra en consola les primeres files del fitxer.

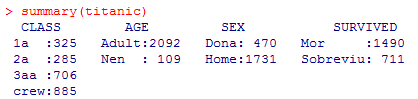


Passo a mirar les dimensions del dataframe *titanic*:



Aquest resultat ens diu que el dataframe *titanic* té 2201 files i 4 columnes. Això en diu que aquest dataframe té 2201 registres i hi ha 4 atributs.

Per tenir una visió general de les variables empraré la funció *summary*, que mostra per a cada variable el valor mínim, el màxim, la mitjana, els tres quartils i el número de NA’s, si es tracta de variables numèriques, i la freqüència d’aparició en el cas de les variables de text:



Així, veiem que tenim 4 atributs, que són:

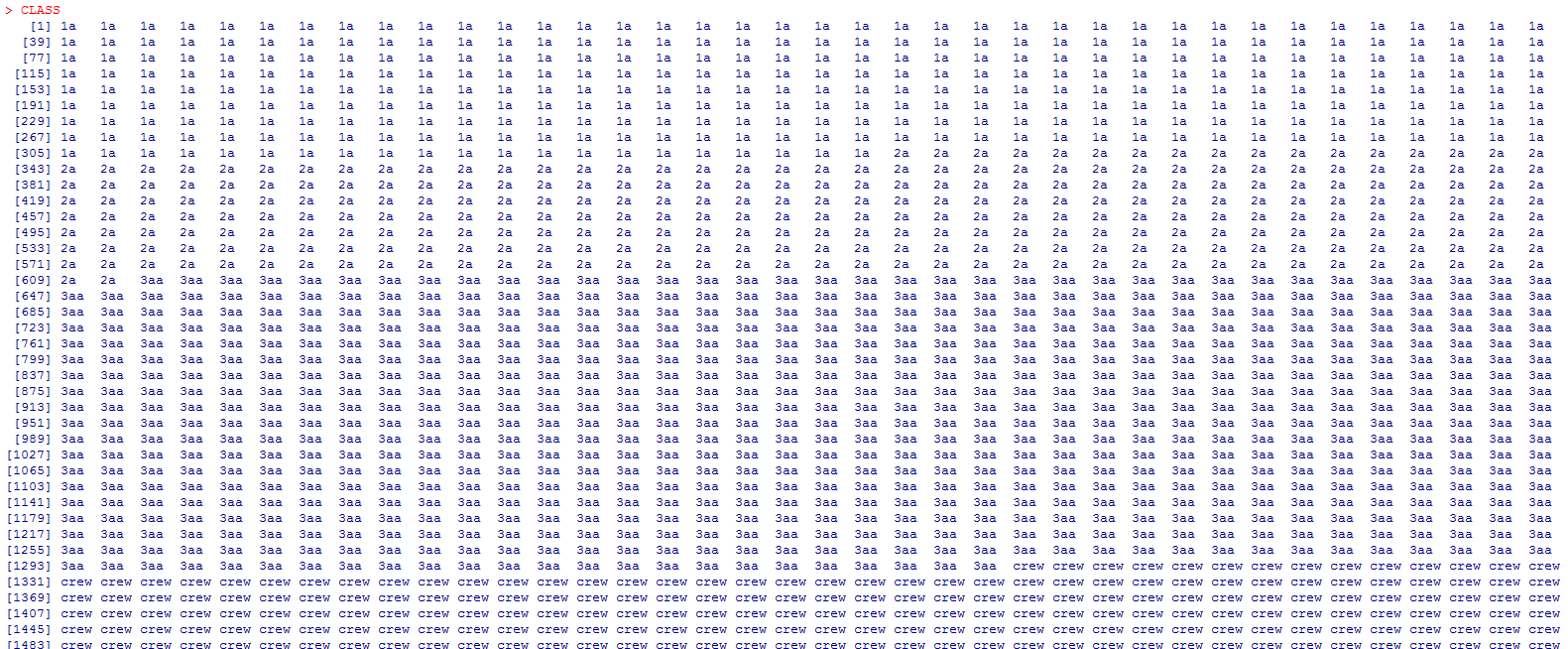
* *CLASS*, amb 4 possibles valors: *1a*, *2a*, *3aa* i *crew*.
* *AGE*, amb 2 possibles valors: *Adult* i *Nen*.
* *SEX*, amb 2 possibles valors: *Dona* i *Home*.
* *SURVIVED*, amb 2 possibles valors: *Mor* i *Sobreviu*.

En cap dels 4 atributs hi apareixen valors NA.

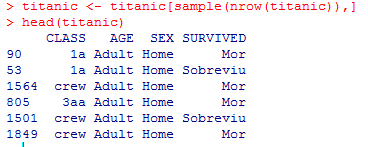
Per tal de poder realitzar l’anàlisi d’una manera més còmode utilitzaré la funció *attach* per poder accedir a les diferents variables donant només els seus noms, sense haver de fer referència cada vegada al dataframe *titanic*:



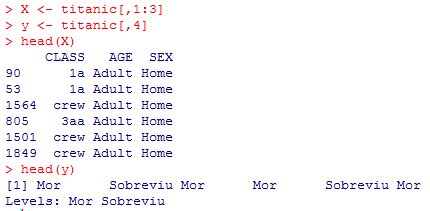
Faig una prova amb una variable, *CLASS*, per exemple:



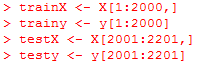
Pel que hem vist en aplicar la funció *head*, sembla que les dades estan ordenades. Convé que tinguem les dades sense ordenar, i per tant podem desordenar-les amb la funció *sample*. Comprovem que estan desordenades visualitzant les primeres files amb *head*:



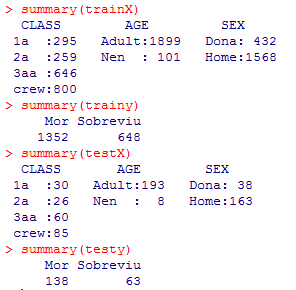
Ara separem separar les dades per tenir per un costat les features, que són les variables que s'utilitzaran per construir el model, i el target, que és la variable que classifica. En el nostre cas la variable per la que classificarem és el camp de si el passatger va sobreviure o no:



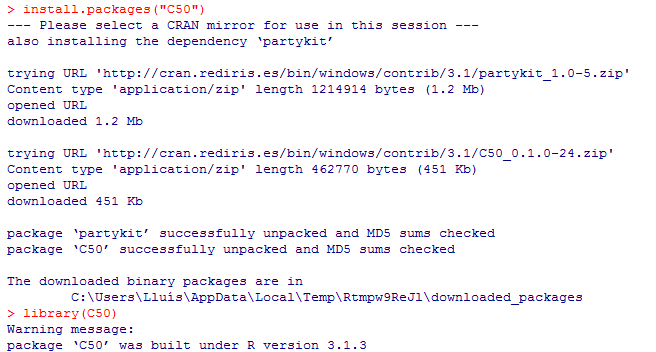
Ara creem un conjunt d’entrenament i un de test:



Fem comprovacions sobre aquests conjunts que hem construït:



A continuació, instal·lem el paquet C50 i accedim a ell:



A continuació podem construir un model, per poder investigar un sumari de la seva sortida. Utilitzaré la notació que es suggereix al document del wiki de l’assignatura, emprant “::” per ajudar a recordar quines funcions provenen de quins paquets:

> model <- C50::C5.0(trainX, trainy)

> summary(model)

Call:

C5.0.default(x = trainX, y = trainy)

C5.0 [Release 2.07 GPL Edition] Mon Nov 06 17:34:53 2017

-------------------------------

Class specified by attribute `outcome'

Read 2000 cases (4 attributes) from undefined.data

Decision tree:

SEX = Home: Mor (1568/337)

SEX = Dona:

:...CLASS in {1a,2a,crew}: Sobreviu (246/20)

CLASS = 3aa: Mor (186/85)

Evaluation on training data (2000 cases):

Decision Tree

----------------

Size Errors

3 442(22.1%) <<

(a) (b) <-classified as

---- ----

1332 20 (a): class Mor

422 226 (b): class Sobreviu

Attribute usage:

100.00% SEX

21.60% CLASS

Time: 0.0 secs

Veiem que es crea el següent arbre:

Decision tree:

SEX = Home: Mor (1568/337)

SEX = Dona:

:...CLASS in {1a,2a,crew}: Sobreviu (246/20)

CLASS = 3aa: Mor (186/85)

Això vol dir que primer classifica per *SEX*, i si és *SEX=Home* classifica que *Mor*, amb 1568 encerts i 337 no encerts. Si és *SEX=Dona*, fa la distinció en dos conjunts. En un d’aquests conjunts es considera que *CLASS={1a, 2a, crew}* i en aquest cas classifica que *Sobreviu*, amb 246 encerts i 20 desencerts. A l’altre conjunt es considera que *CLASS)=3aa* i en aquest cas classifica que *Mor*, amb 186 encerts i 85 desencerts.

Més avall aquest sumari ens diu que s’ha avaluat sobre dades d’entrenament, amb 2000 casos. L’arbre té un tamany de 3 i en total hi ha 442 errors de classificació, que representa un 22.1% del total.

Passem ara a fer un boosting, que és un procés en que s’afegeixen aprenents dèbils de tal maner que aprenents nous tenen en compte els resultats obtinguts per aprenents més antics. D’aquesta manera es por incrementar la precisió del model. Això ho podem fer amb la funció *C5.0()*, i amb el paràmetre *tails* podem triar les iteracions. Anem a provar-ho:

> model <- C50::C5.0( trainX, trainy, trials=10 )

> summary(model)

Call:

C5.0.default(x = trainX, y = trainy, trials = 10)

C5.0 [Release 2.07 GPL Edition] Mon Nov 06 17:52:21 2017

-------------------------------

Class specified by attribute `outcome'

Read 2000 cases (4 attributes) from undefined.data

----- Trial 0: -----

Decision tree:

SEX = Home: Mor (1568/337)

SEX = Dona:

:...CLASS in {1a,2a,crew}: Sobreviu (246/20)

CLASS = 3aa: Mor (186/85)

----- Trial 1: -----

Decision tree:

SEX = Dona: Sobreviu (439.7/115.5)

SEX = Home: Mor (1560.3/549.7)

----- Trial 2: -----

Decision tree:

CLASS = 1a: Sobreviu (293.6/84.8)

CLASS in {2a,3aa,crew}:

:...AGE = Nen: Sobreviu (116.3/41.2)

AGE = Adult:

:...SEX = Dona: Sobreviu (300.1/132.8)

SEX = Home: Mor (1289.9/505.2)

----- Trial 3: -----

Decision tree:

CLASS = 1a: Sobreviu (299.3/110.4)

CLASS in {2a,3aa,crew}: Mor (1700.7/782.7)

----- Trial 4: -----

Decision tree:

SEX = Dona: Sobreviu (449.3/179.4)

SEX = Home:

:...AGE = Nen: Sobreviu (80.4/28)

AGE = Adult:

:...CLASS in {1a,2a,3aa}: Mor (689.2/274.9)

CLASS = crew: Sobreviu (781/372.7)

----- Trial 5: -----

Decision tree:

CLASS = 1a: Sobreviu (293/114)

CLASS in {2a,3aa,crew}: Mor (1707/791.2)

----- Trial 6: -----

Decision tree:

SEX = Dona: Sobreviu (438.6/172)

SEX = Home: Mor (1557.4/738.7)

----- Trial 7: -----

Decision tree:

CLASS in {1a,2a}: Sobreviu (617.9/243.4)

CLASS in {3aa,crew}: Mor (1125.1/180.4)

----- Trial 8: -----

Decision tree:

SEX = Dona: Sobreviu (589.3/255.8)

SEX = Home: Mor (1127.7/200.4)

----- Trial 9: -----

Decision tree:

CLASS = 1a: Sobreviu (471.1/170)

CLASS in {2a,3aa,crew}: Mor (1231.9/231.9)

Evaluation on training data (2000 cases):

Trial Decision Tree

----- ----------------

Size Errors

0 3 442(22.1%)

1 2 458(22.9%)

2 4 512(25.6%)

3 2 575(28.8%)

4 4 888(44.4%)

5 2 575(28.8%)

6 2 458(22.9%)

7 2 616(30.8%)

8 2 458(22.9%)

9 2 575(28.8%)

boost 438(21.9%) <<

(a) (b) <-classified as

---- ----

1332 20 (a): class Mor

418 230 (b): class Sobreviu

Attribute usage:

100.00% CLASS

100.00% SEX

93.60% AGE

Time: 0.0 secs

Veiem que en total es classifiquen malament 438 casos, que representen el 21.9% del total.

Sense boosting havíem obtingut 442 errors de classificació, que representa un 22.1% del total. Veiem que hi ha una lleugera millora, però molt petita.

En total classifica com a Mor 1332+20=1352 casos, dels quals hi ha 1332 encerts i 20 errors, i classifica com a Sobreviu 418+230=648, del quals hi ha 230 encerts i 418 errors.

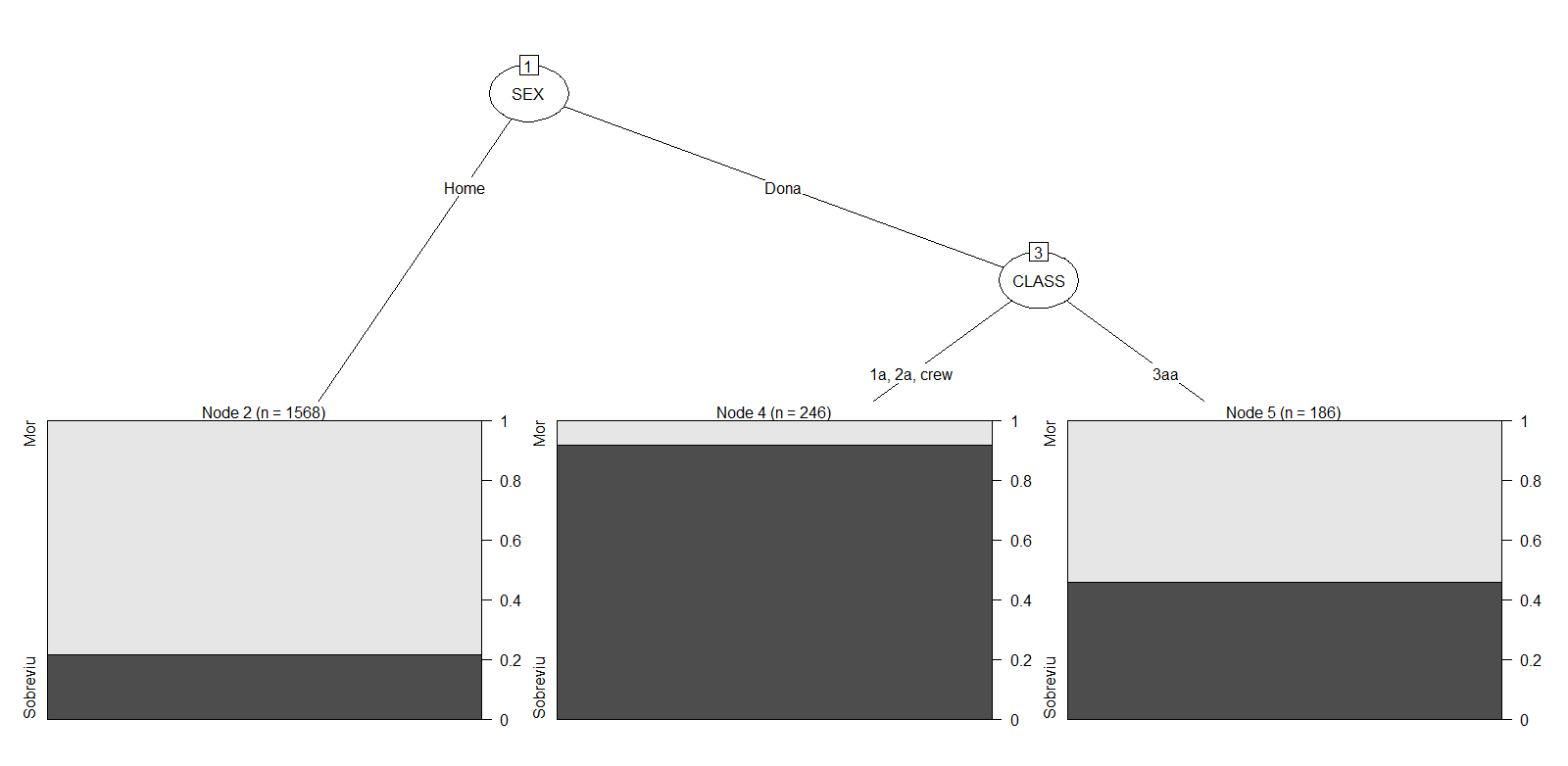
Podem fer prediccions aplicant-hi el model a la funció *predict()*, amb les dades de test. El paràmetre *class* especifica que volem les etiquetes de classe a la sortida, en comptes que la probabilitat de que l’etiqueta de classe sigui una o altra. El que obtenim és el vector *p* amb encerts i errors:



Veiem que obtenim un encert del 83.08%.

Passem ara a crear la representació gràfica de l’arbre:





Veiem que l’arbre classificador comença discriminant segons l’atribut *SEX*. Si és *SEX=Home* tenim una fulla en que majoritàriament el resultat és que *Mor*. Si *SEX=Dona* apareix un altre discriminant, en el qual apareixen dues fulles: en el cas en que *CLASS={1a, 2a, crew}* tenim que majoritàriament el resultat és que *Sobreviu*, i en el cas en que *CLASS=3aa* tenim que no discrimina gaire bé, ja que tenim gairebé els mateixos casos en que els resultats són que *Mor* i *Sobreviu*.

1. **Repetiu l’exercici 2 amb algun altre conjunt de dades. Poden ser dades reals del vostre àmbit laboral o d’algun repositori de dades a Internet. Mireu per exemple:** [**http://www.ics.uci.edu/~mlearn/MLSummary.html**](http://www.ics.uci.edu/~mlearn/MLSummary.html) **Seguiu el guió proposat a la pregunta anterior. Podeu afegir nous punts si ho considereu adequat (per exemple provar el boosting, variar el prunning o provar una altra mena d’arbre...) Recordeu també que el cicle de vida dels projectes de mineria contempla retrocedir per tornar a generar el model amb dades modificades o paràmetres de l'algorisme variats si el resultat no és prou bo.**

Miro al repositori de l’UCI i veig que entre els molts conjunts de dades que s’hi troben hi ha el conjunt de dades Iris, un conjunt que s’ha convertit en un clàssic com a exemple en diversos llibres que tracten sobre Ciència de Dades, i amb el que ja m’hi he trobat algunes vegades. Això em decideix a triar aquest conjunt de dades.

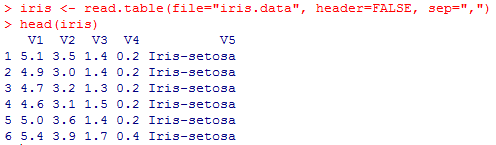
El conjunt de dades conté 3 classes de 50 instàncies cadascuna, on cada classe correspon a un tipus de planta iris.

Tenim dos fitxers:

* iris.data: Fitxer en el que es troben les dades, un registre per cada fila i amb els atributs separats per comes.
* iris.names: Fitxer amb informació sobre el conjunt de dades. Entre aquesta informació, destaquen les següents característiques:
  + Nombre de registres: 150, amb 50 de cadascuna de les 3 classes.
  + Nombre d’atributs: 4, numèrics, am 3 features i un target.
  + Informació dels atributs:
    - Longitud del sèpal, en cm.
    - Amplada del sèpal, en cm.
    - Longitud del pètal, en cm.
    - Amplada del pètal, en cm.sepal length in cm
  + Target: Pot ser de 3 tipus diferents:
    - Iris Setosa.
    - Iris Versicolour.
    - Iris Virginica.
  + No hi ha valors perduts.
  + Informació estadística.

Passem ara a fer una anàlisi similar a la que hem fet a l’exercici anterior.

Començo carregant el fitxer *iris.data* a l’entorn de R, en un dataframe amb nom *iris*. Comprovo que s’ha carregat correctament amb la funció *head*, que mostra en consola les primeres files del fitxer.

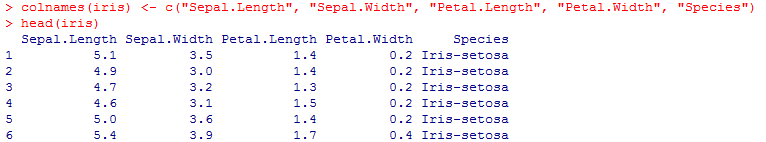


Passo a mirar les dimensions del dataframe *titanic*:

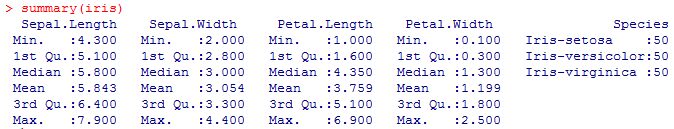


El dataframe *iris* té 150 files i 5 columnes. Això en diu que aquest dataframe té 150 registres i hi ha 5 atributs.

Passem ara a nombrar les columnes amb uns noms més intuïtius:

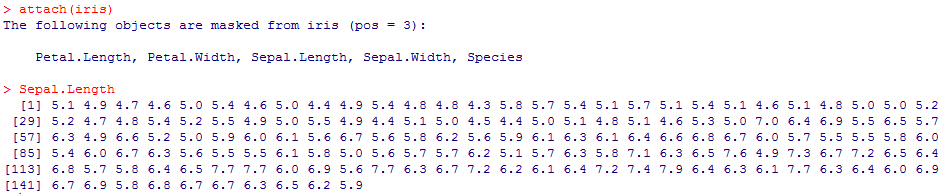


Mirem ara el sumari d’aquest dataset amb la funció *summary()*:

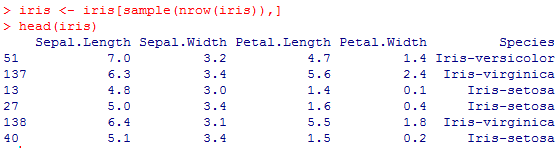


Veiem que, efectivament, en cap dels atributs hi apareixen valors NA.

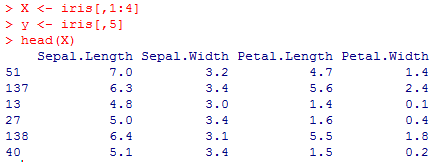
Per tal de poder realitzar l’anàlisi d’una manera més còmode utilitzaré la funció *attach* per poder accedir a les diferents variables donant només els seus noms, sense haver de fer referència cada vegada al dataframe *iris*. Faig una prova amb una variable, *Sepal.Length*, per exemple:



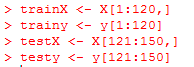
Pel que hem vist en aplicar la funció *head*, sembla que les dades estan ordenades. Convé que tinguem les dades sense ordenar, i per tant podem desordenar-les amb la funció *sample*. Comprovem que estan desordenades visualitzant les primeres files amb *head*:



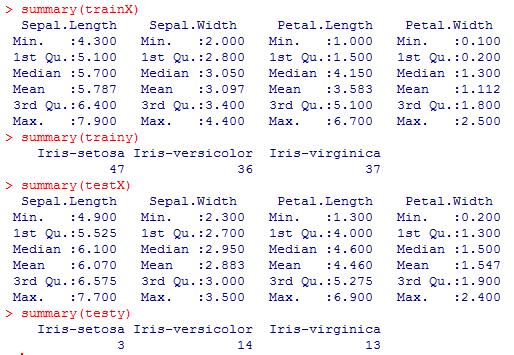
Ara separem les dades per tenir per un costat els *feature*s, que són les variables que s'utilitzaran per construir el model, i el *target*, que és la variable que classifica. En el nostre cas la variable per la que classificarem és el camp *Species*.



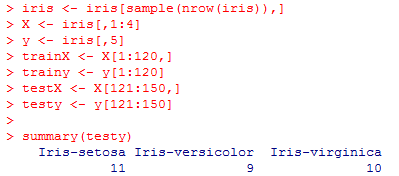
Ara creem un conjunt d’entrenament i un de test:



Fem comprovacions sobre aquests conjunts que hem construït:



Amb aquests resultats veiem com la barreja aleatòria no ha estat gaire bona, ja que en el conjunt d’entrenament hi ha molts registres que es corresponen a Irsi-setosa i en el conjunt de test només n’hi ha 3. Per tant, torno a repetir el procés fins que obting uns conjunts més ben repartits:



A continuació podem construir un model amb el paquet C5.0, per poder investigar un sumari de la seva sortida:

> model\_iris <- C50::C5.0(trainX, trainy)

> summary(model\_iris)

Call:

C5.0.default(x = trainX, y = trainy)

C5.0 [Release 2.07 GPL Edition] Mon Nov 06 19:48:04 2017

-------------------------------

Class specified by attribute `outcome'

Read 120 cases (5 attributes) from undefined.data

Decision tree:

Petal.Length <= 1.9: Iris-setosa (39)

Petal.Length > 1.9:

:...Petal.Width > 1.7: Iris-virginica (38/1)

Petal.Width <= 1.7:

:...Petal.Length <= 5: Iris-versicolor (39)

Petal.Length > 5: Iris-virginica (4/1)

Evaluation on training data (120 cases):

Decision Tree

----------------

Size Errors

4 2( 1.7%) <<

(a) (b) (c) <-classified as

---- ---- ----

39 (a): class Iris-setosa

39 2 (b): class Iris-versicolor

40 (c): class Iris-virginica

Attribute usage:

100.00% Petal.Length

67.50% Petal.Width

Time: 0.0 secs

Veiem que es crea el següent arbre:

Decision tree:

Petal.Length <= 1.9: Iris-setosa (39)

Petal.Length > 1.9:

:...Petal.Width > 1.7: Iris-virginica (38/1)

Petal.Width <= 1.7:

:...Petal.Length <= 5: Iris-versicolor (39)

Petal.Length > 5: Iris-virginica (4/1)

Això vol dir que primer classifica per *Petal.Length*.Si és *Petal.Length<=1.9* classifica *Iris-setosa*, amb 39 encerts i cap error. Si és *Petal.Length>1.9*, entra en una altra branca, que classifica segons *Petal.Width*. Si és Petal.Width>1.7 classifica com *Iris-virginica*, amb 38 encerts i 1 error. Per *Petal.Width*<=1.7 entra en una altra branca, que classifica segons *Petal.Length*. Si és *Petal.Length*<=5 classifica com *Iris-versicolor*, amb 39 encerts i cap error, i si és *Petal.Length*>5 classifica com *Iris-virginica*, amb 4 encerts i 1 error.

L’arbre té un tamany de 4 i en total hi ha 2 errors de classificació, que representa un 1.7% del total.

Passem ara a fer un boosting:

> model\_iris <- C50::C5.0( trainX, trainy, trials=10 )

> summary(model\_iris)

...

Evaluation on training data (120 cases):

Trial Decision Tree

----- ----------------

Size Errors

0 4 2( 1.7%)

1 4 5( 4.2%)

2 4 14(11.7%)

3 3 5( 4.2%)

4 3 4( 3.3%)

5 5 3( 2.5%)

6 4 9( 7.5%)

7 5 6( 5.0%)

8 4 2( 1.7%)

9 6 1( 0.8%)

boost 1( 0.8%) <<

(a) (b) (c) <-classified as

---- ---- ----

39 (a): class Iris-setosa

41 (b): class Iris-versicolor

1 39 (c): class Iris-virginica

Veiem que en total es classifica malament un sol cas, que representa el 0.8% del total.

Sense boosting havíem obtingut 2 errors de classificació, que representa un 1.7% del total. Veiem que hi ha una lleugera millora.

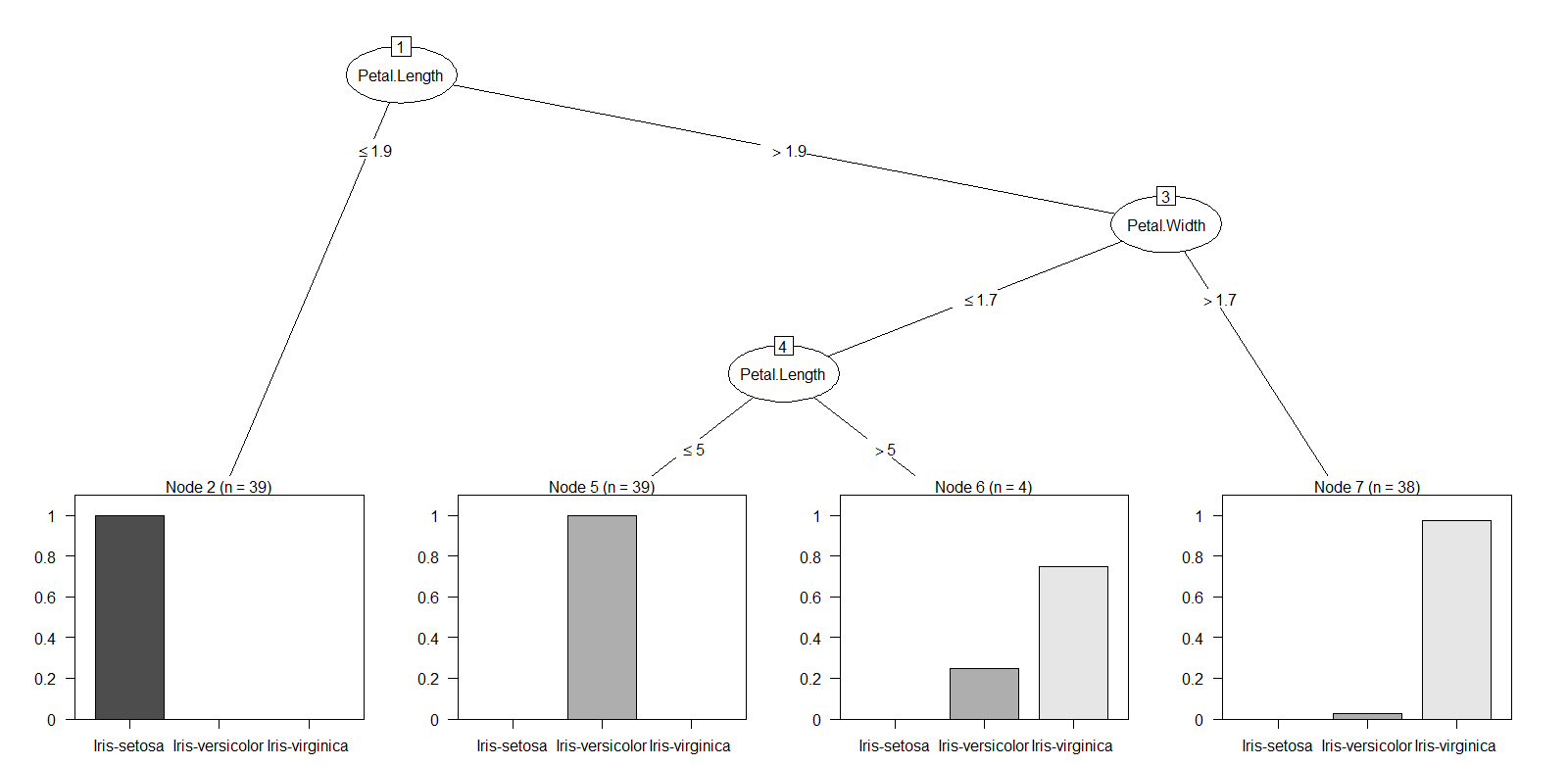
Podem fer prediccions aplicant-hi el model a la funció *predict()*, amb les dades de test:



Veiem que obtenim un encert del 93.33%.

Passem ara a crear la representació gràfica de l’arbre:





**BIBLIOGRAFIA**

* Hernández Orallo, J.; Ramírez Quintana, M. J.; Ferri Ramírez, C.: *Introducción a la Minería de Datos*. Pearson Prentice Hall. Madrid. 2004.
* Mor i Pera, E.; Sangüesa i Solé, R.; Molina Félix, L. C.: *Mineria de dades*. Material docent de la UOC. UOC – Universitat Oberta de Catalunya.
* Decision Trees in R using the C50 package | Connor\_Johnson

*http://connor-johnson.com/2014/08/29/decision-trees-in-r-using-the-c50-package/*

* Package “C50”

*https://cran.r-project.org/web/packages/C50/C50.pdf*

* UCI Machine Learning Repository

*http://archive.ics.uci.edu/ml/index.php*