Usando Weka desde R, inicio

Nahuel Bargas

2020-11-21

Crossover

- Weka y R nacieron en el mismo país, Nueva Zelanda.
- R(R Core Team (2018)) es un sistema para realizar análisis estadísticos,econométricos, minería de datos y gráficos de gran calidad, entre otras posibilidades. Gracias a la adición de paquetes con características adicionales, su espectro de alcance tiene varias ramificaciones. Es de código abierto y cuenta con una gran cantidad de autores a lo largo del mundo que contribuyern a su mantenimiento y actualización .

Conociendo a Weka

Características

- **Weka**(Witten and Frank (2005)) está escrito en Java, y permite realizar una serie de tareas relacionadas con el aprendisaje autómatico, comprendiendo el *procesamiento de la información* ,*algoritmos de clasificación*, *regresión*, *análisis de clústeres* y *selección de atributos*.

- Su interface gráfica es aména, y permite realizar todo el análisis sin escribir ninguna línea de código, aunque si se quiere tener un registro de cada paso dado, se puede utilizar el API de JAVA que provee Weka.

- Al igual que **R**, complementa sus funciones básicas con otros programas que añaden interoperabilidad con otros softwares(cómo el propio R, Python, Spark, Hadoop, Jython, Groovy, entre otros) y mayor cantidad de algoritmos.

- Puede instalar la versión de Weka que desee pinchando aquí

- Luego de configurar el programa correctamente, eche un vistazo en la documentación que se encuentra en la carpeta de instalación, allí podrá observar con detalle el funcionamiento y las especificaciones de los algoritmos, filtros, etc.

Su funcionamiento desde R

- Gracias al wrapper **RWeka**(Hornik, Buchta, and Zeileis (2009)) y al paquete **rJava**(Urbanek (2019)), desde la consola de R podemos interactuar con Weka, aplicar filtros, evaluar los modelos y graficar.

Atención Debe generarse una variable de sistema(¡En Windows!) *WEKA_HOME* que apunte a la carpeta wekafiles, el lugar en el cuál el sistema guarda la información de los paquetes instalados de WEKA.

- Por ejemplo, para verificar los paquetes instalados en sus sistema por WEKA, usaríamos:

```
RWeka::WPM("list-packages","installed") # Si nunca ha utilizado/instalado un paquete
#           externo de WEKA, el comando no arrojará ningún resultado positivo.
```

Si bien no es estrictamente necesario, recomiendo instalar Weka para corroborar los resultados que obtenemos en R, y poder utilizar funciones que no se encuentran incorporadas en RWeka.

Conociendo los archivos "arff"

El archivo "vidrio.arff"

- 'Vidrio.arff' es casi idéntico a glass.arff, salvo que posee una traducción al castellano del atributo clase.
- Pero un momento, ¿Qué es un archivo "arff"?.
- Es un tipo de formato por el cuál WEKA relaciona distintos tipos de atributos.
- Los atributos son las variables, el conjunto de datos que nos ayudarán a armar el modelo.
- Los atributos pueden ser valores numéricos(tanto enteros cómo reales), nominales o presentarse en formato de cadena de texto.
- Si abrimos el archivo 'arff' con un bloc de notas, podemos observar que habitualmente se realizan comentarios sobre la base, que significa cada variable, la fuente de datos y demás, interponiendo el signo '%'.
- Posteriormente, se puede determinar la relación(habitualmente el nombre de la base), los atributos y los valores que toman los mismos.
- Las líneas siquientes a la expresión @data completan la base de datos conformando cada una las instancias, el número de observaciones por atributo.
- El dato faltante se indica con el signo de interrrogación (?).

El archivo "vidrio.arff -2- "

- El archivo "vidrio.arff" detalla que la base de datos consta de 10 atributos(incluyendo la clase) y 214 instancias.
- Nuestra variable de interés, que señala la clase, es 'Tipo'. Existen siete tipos de vidrios en la muestra, aquellos usados para vajilla, para faros delanteros, para contenedores, a los que se les sometió a un proceso de flotado y se destino a ventanas para la construcción y vehículos, y aquellos que tuvieron el mismo destino pero no requirieron dicho proceso.
- El resto de atributos corresponden al índice de refracción y a diversos elementos representados por sus nombres químicos.
- Los atributos no tienen datos faltantes.

El archivo "vidrio.arff -3-"

- El archivo "vidrio.arff" es tan generoso que nos provee de los valores máximo, mínimo, la media y el desvío estandar que posee cada atríbuto en un comentario, pero abramos la base en R y calculemos dichos valores por nuestra cuenta:

```
datos<-RWeka::read.arff("archivos/vidrio.arff")

resumen <- function(x) c(min(x), mean(x), max(x), sd(x))

kable(data.frame("Val"=rbind("Mín.","Media","Máx.","DS."),
       apply(datos[,1:9],2,resumen)),digits=2, align="c")
```

Val	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
Mín.	1.51	10.73	0.00	0.29	69.81	0.00	5.43	0.00	0.00
Media	1.52	13.41	2.68	1.44	72.65	0.50	8.96	0.18	0.06
Máx.	1.53	17.38	4.49	3.50	75.41	6.21	16.19	3.15	0.51
DS.	0.00	0.82	1.44	0.50	0.77	0.65	1.42	0.50	0.10

¿ Se ve mejor la tabla si aplicamos Stargazer?

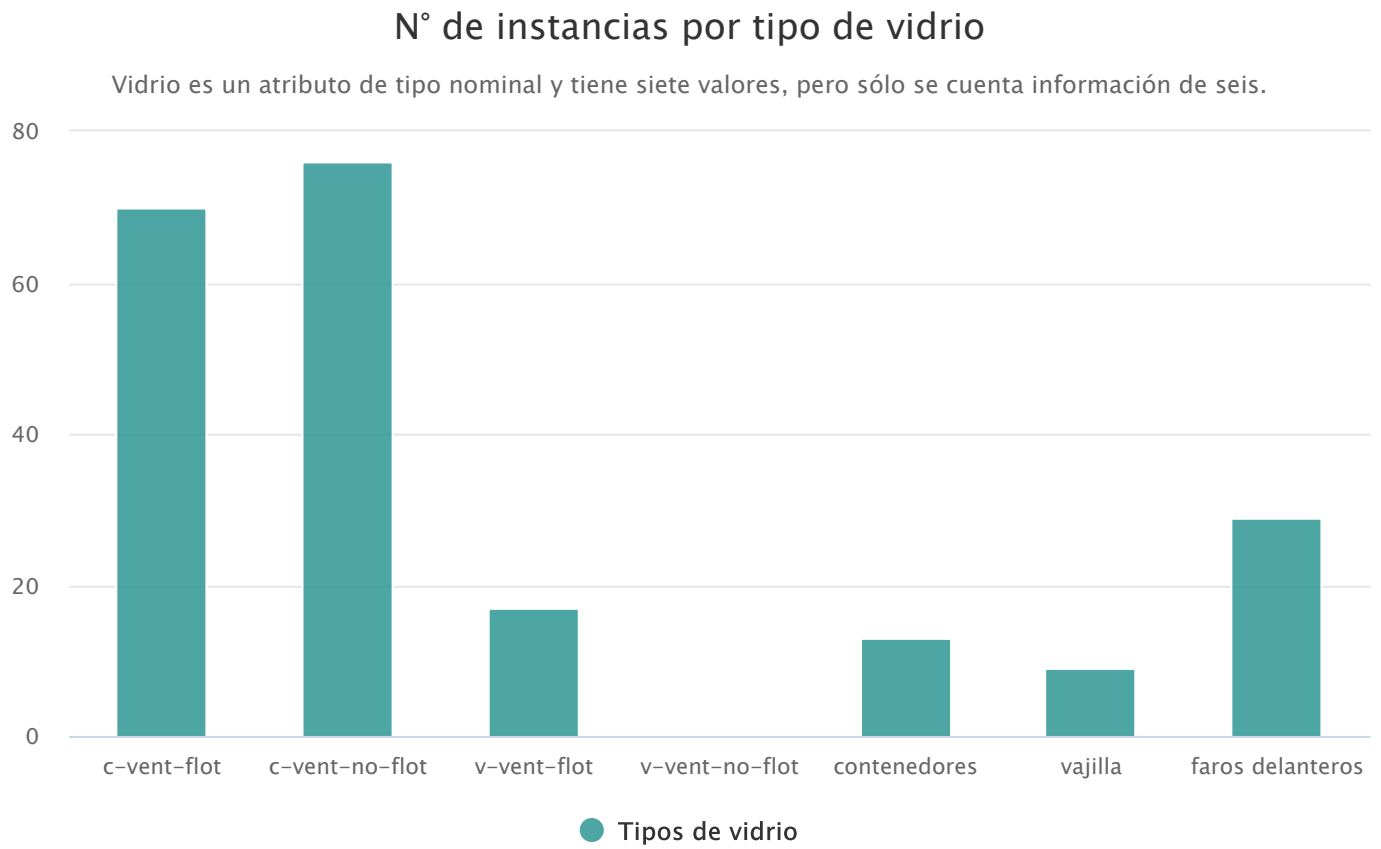
- Por lo menos,nos ahorramos unos pasos...

```
stargazer(datos, type="html", summary.stat=c("min","mean","max","sd"))
```

Statistic	Min	Mean	Max	St. Dev.
RI	1.511	1.518	1.534	0.003
Na	10.730	13.408	17.380	0.817
Mg	0.000	2.685	4.490	1.442
Al	0.290	1.445	3.500	0.499
Si	69.810	72.651	75.410	0.775
K	0.000	0.497	6.210	0.652
Ca	5.430	8.957	16.190	1.423
Ba	0.000	0.175	3.150	0.497
Fe	0.000	0.057	0.510	0.097

El archivo "vidrio.arff -4-"

¿ Cuantos tipos de vidrio hay en la muestra?



Un ejemplo práctico

Árbol de decisión: J48

- Uno de los algoritmos más populares en Weka para encarar un problema de clasificación es J48, el cuál genera un árbol de decisión de la familia del programa **C4.5(Quinlan (1993))**, la última versión abierta de un software tan exitoso que ahora tiene una licencia comercial.

- Evaluemos el modelo de decisión dividiendo la base de entrenamiento en dos partes, quedándonos, arbitrariamente, con el 70% para armar el árbol y el porcentaje restante para testearlo.

```
set.seed(1)

gen<-sample(214,150)

train<-datos[gen,]
test<-datos[-gen,]


m1<-J48(Tipo~.,data=train)
e1<-evaluate_Weka_classifier(m1, newdata = test)

e1
```

```
=== Summary ===

Correctly Classified Instances          40               62.5    %
Incorrectly Classified Instances        24               37.5    %
Kappa statistic                          0.482
Mean absolute error                      0.1142
Root mean squared error                  0.3087
Relative absolute error                 55.0597 %
Root relative squared error             96.7429 %
Total Number of Instances               64

=== Confusion Matrix ===

  a  b  c  d  e  f  g   <-- classified as
 13  4  5  0  0  0  0 |  a = c-vent-flot
  5 14  1  0  2  0  0 |  b = c-vent-no-flot
  0  3  1  0  0  0  0 |  c = v-vent-flot
  0  0  0  0  0  0  0 |  d = v-vent-no-flot
  0  0  0  0  1  0  0 |  e = contenedores
  0  1  0  0  0  0  0 |  f = vajilla
  3  0  0  0  0  0 11 |  g = faros delanteros
```

Árbol de decisión: J48 -2-

- Esta vez usemos otro método de evaluación, tomando la base completa, dividiendola en diez partes con la misma cantidad de observaciones, de los cuáles una por vez se usara para testear el modelo formado por los nueve restantes,lo que se conoce cómo 'Cross-Validation' con **k=10**, siendo k el número de subparticiones('folds').

- La ventaja de llevar a cabo el mencionado procedimiento proviene de la reducción de la varianza en la estimación del error sobre los datos testeados (James, Witten, Hastie, et al. (2017b)), que, en nuestro caso, comprendería el número de observaciones mal clasificadas(37.5% en el ejemplo anterior).

```
m2<-J48(Tipo~.,data=datos)
e2<-evaluate_Weka_classifier(m2,numFolds=10,seed=1)

e2
```

```
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances         143               66.8224 %
Incorrectly Classified Instances        71               33.1776 %
Kappa statistic                          0.55
Mean absolute error                      0.1197
Root mean squared error                  0.3129
Relative absolute error                 48.536  %
Root relative squared error             89.2762 %
Total Number of Instances              214

=== Confusion Matrix ===

  a  b  c  d  e  f   <-- classified as
 50 15  3  0  1  1 |  a = c-vent-flot
 16 47  6  2  3  2 |  b = c-vent-no-flot
  5  5  6  0  1  0 |  c = v-vent-flot
  0  2  0 10  0  1 |  d = contenedores
  1  1  0  0  7  0 |  e = vajilla
  3  2  0  0  1 23 |  f = faros delanteros
```

Características J48

¿Podar o no podar?

- ¿Que tál si no nos preocupa el tamaño final del árbol de decisión?

```
m2U<-J48(Tipo~.,data=datos,
    control=Weka_control(U=TRUE))
# control= Nos permite cambiar los parámetros que toma el clasificador

e2U<-evaluate_Weka_classifier(m2U,numFolds=10,seed=1)

e2U
```

```
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances         144               67.2897 %
Incorrectly Classified Instances        70               32.7103 %
Kappa statistic                          0.5571
Mean absolute error                      0.1168
Root mean squared error                  0.3082
Relative absolute error                 47.3498 %
Root relative squared error             87.9266 %
Total Number of Instances              214

=== Confusion Matrix ===

  a  b  c  d  e  f   <-- classified as
 50 15  3  0  1  1 |  a = c-vent-flot
 16 47  6  2  3  2 |  b = c-vent-no-flot
  5  4  7  0  1  0 |  c = v-vent-flot
  0  2  0 10  0  1 |  d = contenedores
  1  1  0  0  7  0 |  e = vajilla
  3  2  0  0  1 23 |  f = faros delanteros
```

Características J48 -2-

- Con el árbol sin podar, obtuvimos mejores resultados en términos de clasificación, pero debemos saber que al no podar, el modelo generado se ajustará demasiado a los datos de entrenamiento, por lo que podríamos perder potencia, una mayor tasa de error.
- La tasa de error en éste problema de clasificación sería la proporción de observaciones que no pertenecen a la clase más común del nodo.
- De la matriz de confusión subyace que todos los elementos por fuera de la diagonal principal se encuentran mal clasificados.

Características J48 -3-

Comparemos

- Podemos reducir el tamaño del árbol aumentando el número de instancias(o filas) por cada 'hoja' o 'nodo final', o modificando el parámetro de intervalo de confianza que utiliza el clasificador.

```
m2P8<-J48(Tipo~.,data=datos,control=Weka_control(M=8))
#<< En la documentación, M es el número de instancias por 'hoja'.

e2P8<-evaluate_Weka_classifier(m2P8,numFolds=10,seed=1)

e2P8
```

- Empeoramos en términos de clasificación, pero tenemos un árbol más chico, más simple a la hora de la interpretación.
- En terminos de predictivos, hay otros mecanismos supervisados que nos pueden otorgar mejores resultados.

```
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances         141               65.8879 %
Incorrectly Classified Instances        73               34.1121 %
Kappa statistic                          0.5419
Mean absolute error                      0.138
Root mean squared error                  0.2942
Relative absolute error                 55.9499 %
Root relative squared error             83.931  %
Total Number of Instances              214

=== Confusion Matrix ===

  a  b  c  d  e  f   <-- classified as
 47 19  2  0  1  1 |  a = c-vent-flot
 15 45  5  2  7  2 |  b = c-vent-no-flot
  7  4  5  0  1  0 |  c = v-vent-flot
  0  1  0 10  1  1 |  d = contenedores
  0  1  0  0  8  0 |  e = vajilla
  1  1  0  0  1 26 |  f = faros delanteros
```

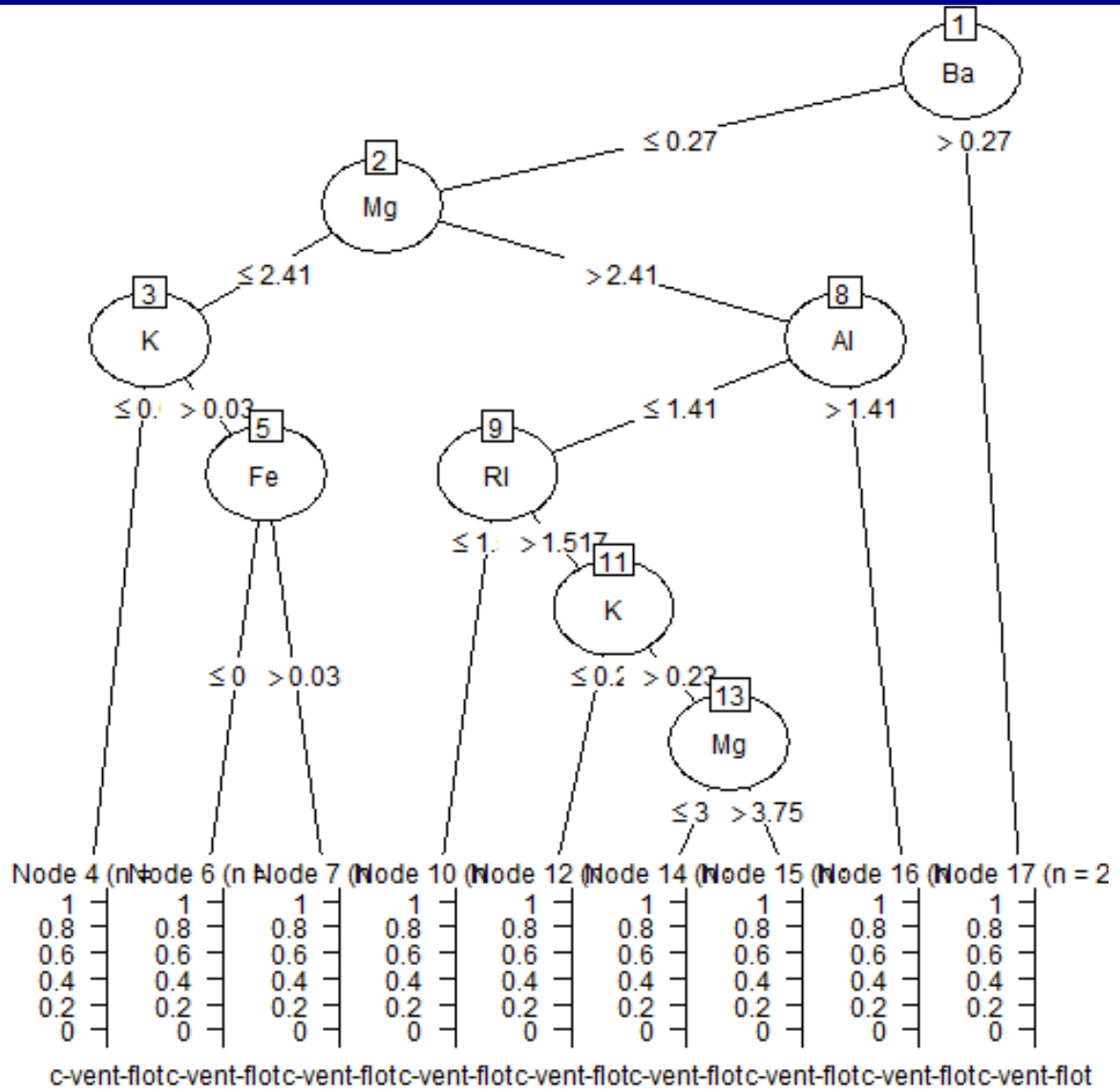
Visualización árbol de decisión

- La forma más rápida es vía el paquete "partykit":

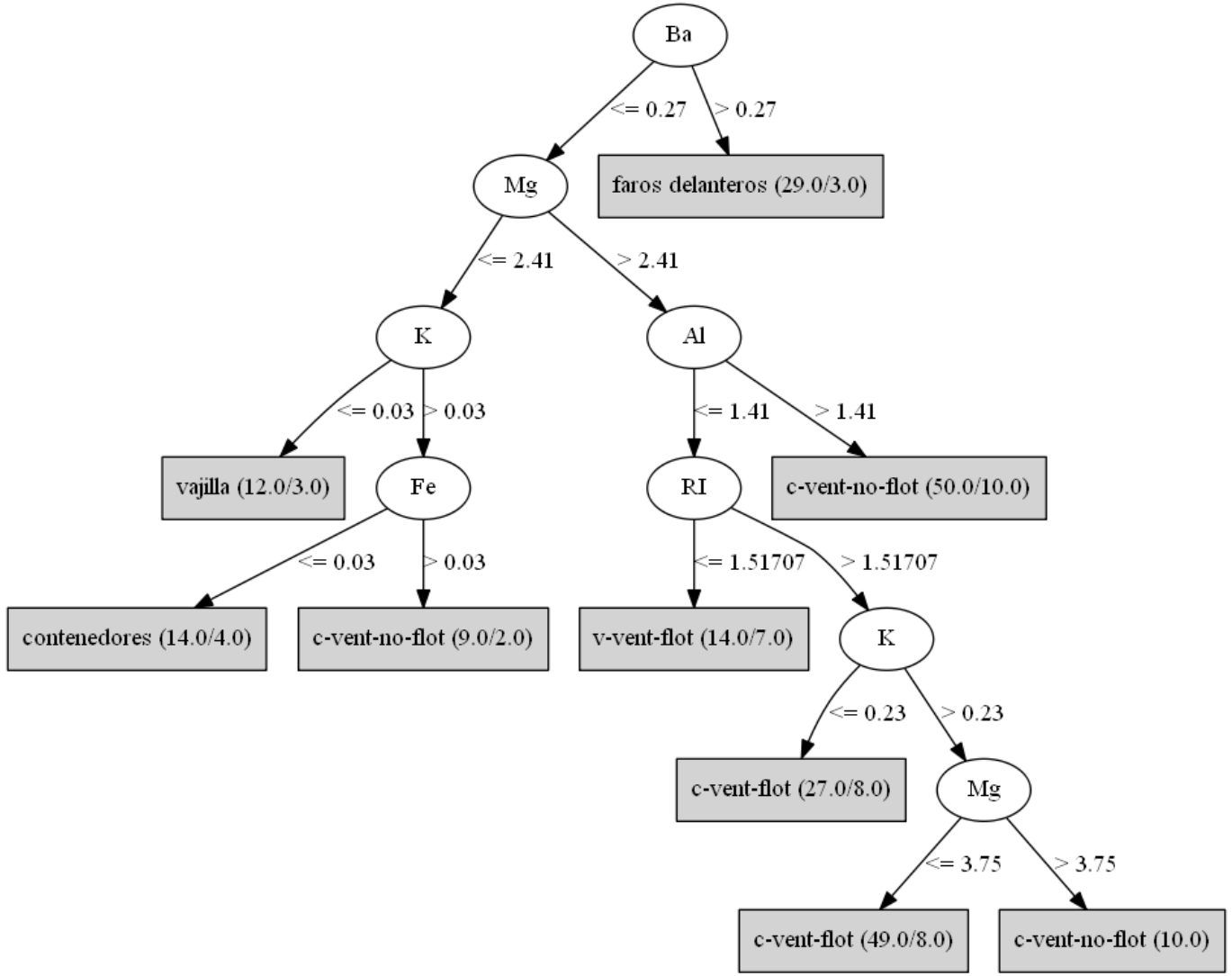
```
plot(m2P8) # visualmente poco atractivo.
```

- Al final de cada nodo finales u hojas, se observa el número de instancias totales que comprenden cada terminal y la cantidad que fue incorrectamente clasificada.
- El **bario** es el primer atributo que toma el modelo a la hora de realizar la clasificación.
- Otra opción sería transformar el modelo en un archivo **dot** mediante la función write_to_dot y copiando el archivo **dot** a 'Graphviz':

```
write_to_dot(m2P8) # y utilizarlo en Graphviz
```



Árbol de decisión usando 'partykit'



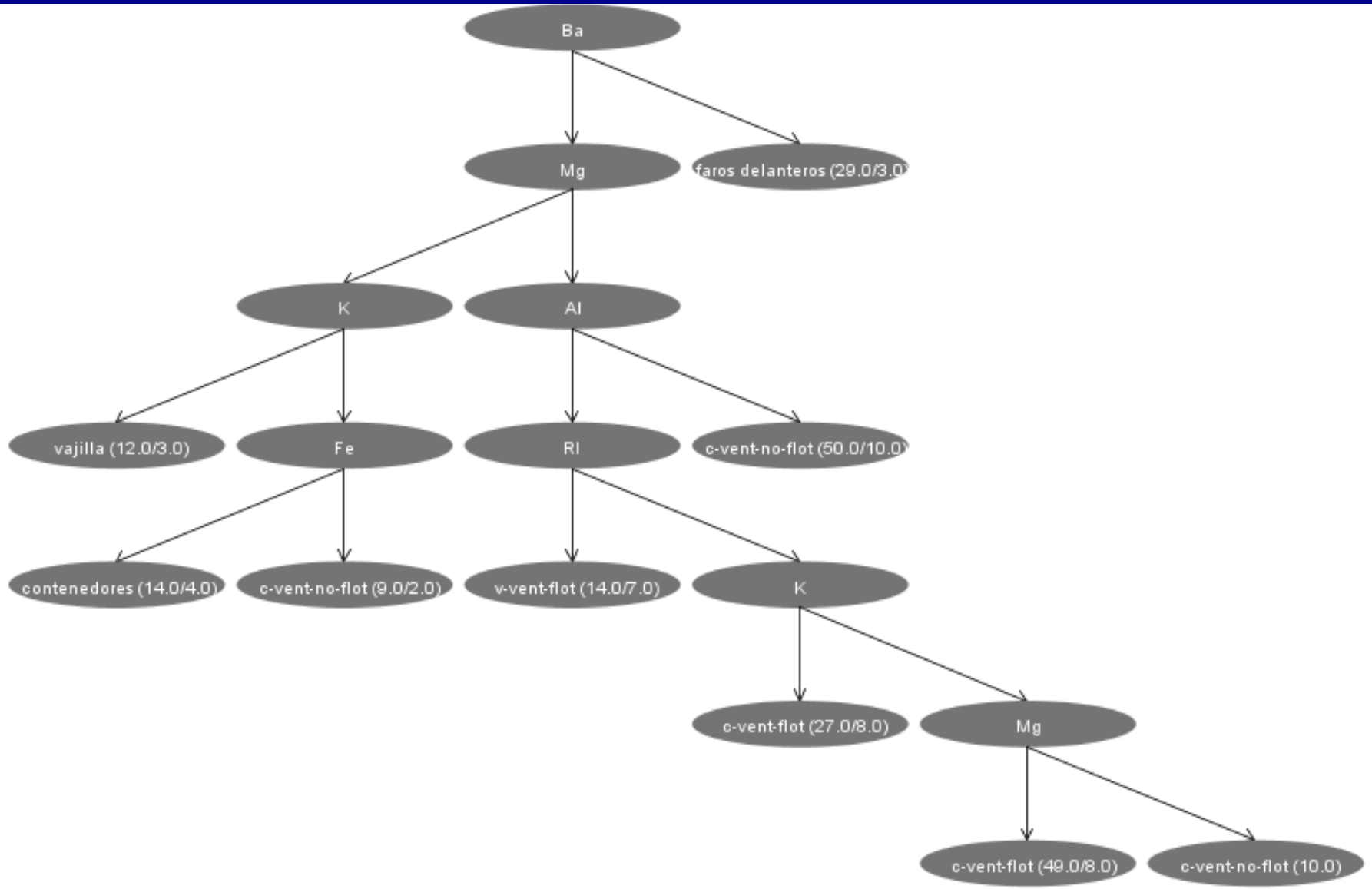
Árbol de decisión usando RWeka::write_to_dot(m2P8) y copiando el archivo dot a 'Graphviz'

```
library(rJava)  #Ejemplo tomado de la viñeta de RWeka

graphVisualizer <-
function(file, width = 400, height = 400,
         title = substitute(file), ...)
{    ## Build the graph visualizer
    visualizer <- .jnew("weka/gui/graphvisualizer/GraphVisualizer")
    reader <- .jnew("java/io/FileReader", file)
    .jcall(visualizer, "V", "readDOT",
           .jcast(reader, "java/io/Reader"))
    .jcall(visualizer, "V", "layoutGraph")
    ## and put it into a frame.
    frame <- .jnew("javax/swing/JFrame",
                   paste("graphVisualizer:", title))
    container <- .jcall(frame, "Ljava/awt/Container;", "getContentPane")
    .jcall(container, "Ljava/awt/Component;", "add",
           .jcast(visualizer, "java/awt/Component"))
    .jcall(frame, "V", "setSize", as.integer(width), as.integer(height))
    .jcall(frame, "V", "setVisible", TRUE)
}

write_to_dot(m2P8,"m2P8.dot")

graphVisualizer("m2P8.dot") # que desplegará el árbol resultante una nueva ventana
```



Árbol de decisión vía Weka-GUI y Java

Bagging

- Vamos a acabar ésta serie de ejemplos con métodos que nos permitirán aumentar la capacidad predictiva, a costa de una perdida en la interpretación, ya que no nos podremos quedar con un único árbol.
- **Bagging** utiliza bootstrap, construyendo modelos de muestras con repetición de la base de entrenamiento y combinando las predicciones de cada uno a través un 'voto mayoritario' que señala una predicción general ,obteniendo de esta forma, una menor varianza en la estimación(James, Witten, Hastie, et al. (2017b)).
- Corramos el "meta-clasificador" con J48 cómo *base-learner* y diez interacciones, éste último valor viene por default.
- Obtendremos una mejor tasa de clasificación con respecto al caso del modelo m2.

```
mbag <- Bagging(Tipo~.,data=datos,
        control=Weka_control(W=list(J48)))

ebag<- evaluate_Weka_classifier(mbag,numFolds=10,seed=1)
```

```
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances         159               74.2991 %
Incorrectly Classified Instances        55               25.7009 %
Kappa statistic                          0.6509
Mean absolute error                      0.122
Root mean squared error                  0.2541
Relative absolute error                 49.4821 %
Root relative squared error             72.482  %
Total Number of Instances              214

=== Confusion Matrix ===

  a  b  c  d  e  f   <-- classified as
 55 12  1  0  1  1 |  a = c-vent-flot
  9 55  5  3  2  2 |  b = c-vent-no-flot
  5  4  6  0  1  1 |  c = v-vent-flot
  0  1  0 11  0  1 |  d = contenedores
  0  1  0  0  8  0 |  e = vajilla
  2  3  0  0  0 24 |  f = faros delanteros
```

Random Forest

- Es un método similar a Bagging pero, a diferencia, se otorga aleatoriedad al algoritmo, no a la base de entrenamiento.

- Con J48, se seleccionaba el mejor atributo para dividir las ramas. Aquí, entre algunas opciones,que están entre las mejores, se elige aleatoriamente una.

- Vamos a construir el árbol con 100 interacciones, eligiendo entre 4 atributos o características principales por cada decisión que toma el modelo y con una "profundidad/ altura" ilimitada.

```
rforest<-make_Weka_classifier("weka/classifiers/trees/RandomForest")
#creamos una conexión de R con el clasificador de Weka

mforest<- rforest(Tipo~.,data=datos,
        control=Weka_control(K=4))

eforest<- evaluate_Weka_classifier(mforest,numFolds=10,seed=1)
```

```
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances         172               80.3738 %
Incorrectly Classified Instances        42               19.6262 %
Kappa statistic                          0.7301
Mean absolute error                      0.1157
Root mean squared error                  0.2282
Relative absolute error                 46.934  %
Root relative squared error             65.0862 %
Total Number of Instances              214

=== Confusion Matrix ===

  a  b  c  d  e  f   <-- classified as
 61  7  2  0  0  0 |  a = c-vent-flot
  8 62  2  2  1  1 |  b = c-vent-no-flot
  8  3  6  0  0  0 |  c = v-vent-flot
  0  1  0 11  0  1 |  d = contenedores
  0  1  0  0  8  0 |  e = vajilla
  1  3  0  1  0 24 |  f = faros delanteros
```

Resúmen

- A lo largo de las slides he tocado diversos temas, con la profundidad que permite el espacio:
  - ¿ Qué es Weka?
  - ¿Cómo se relaciona con R? ¿Cómo se lo implementa?
  - Estructura de los archivos ".arff"
  - Problemas de clasificación y un ejemplo práctico con el algoritmo J48.
  - Visualización de los árboles generados.
  - Random Forest y Bagging.
- Para ahondar más en los aspectos teóricos, puede revisar las referencias bibliográficas que se detallan a continuación.
- Los temas relacionados con WEKA son muchos, todavía queda un largo camino por recorrer. Aquí les he presentado un pequeño pantallazo.
- Encontré de mucha utilidad la serie de *MOOC* de Weka que se proveen en FutureLearn , así que los interesados en conocer más sobre lo que el software brinda, no duden en realizarlos.

Rerefencias

Hornik, K, C. Buchta, and A. Zeileis (2009). "Open-Source Machine Learning: R Meets Weka". In: *Computational Statistics* 24.2, pp. 225-232. DOI: 10.1007/s00180-008-0119-7.

James, G, D. Witten, T. Hastie, et al. (2017b). *ISLR: Data for an Introduction to Statistical Learning with Applications in R*. R package version 1.2. URL: https://CRAN.R-project.org/package=ISLR.

Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Urbanek, S. (2019). *rJava: Low-Level R to Java Interface*. R package version 0.9-11. URL: https://CRAN.R-project.org/package=rJava.

Witten, I. H. and E. Frank (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd. San Francisco: Morgan Kaufmann.

- Crédito de la imágen del áve weka : "https://phys.org/news/2019-09-weka-sandwich-stealing-scallywags-ecosystem.html" , recuperada el 16/11/2020.

Contacto

- Pueden contactarme para cualquier sugerencia o comentario a mi correo electrónico. O también vía:

- El código para reproducir las slides lo podrán encontrar en el repositorio y si quieren una copia en pdf, visiten éste enlace .

¡ Muchas Gracias !