Ciencia de datos, práctica 2

Juan Casado Ballesteros, Samuel García Gonzalez, Iván Anaya Martín

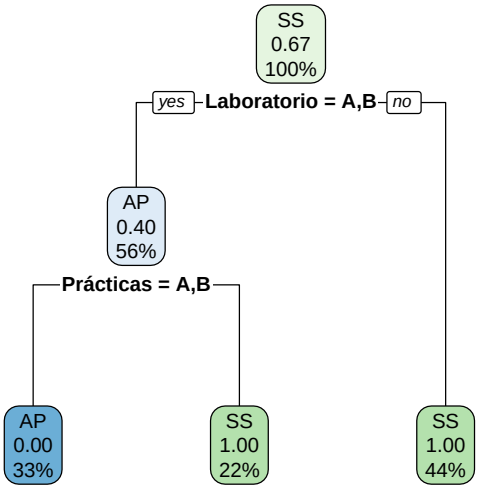
October 29, 2019

Abstract

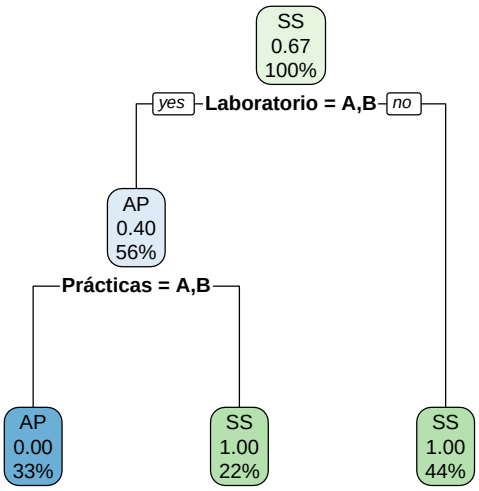
Contents

1 EJ1

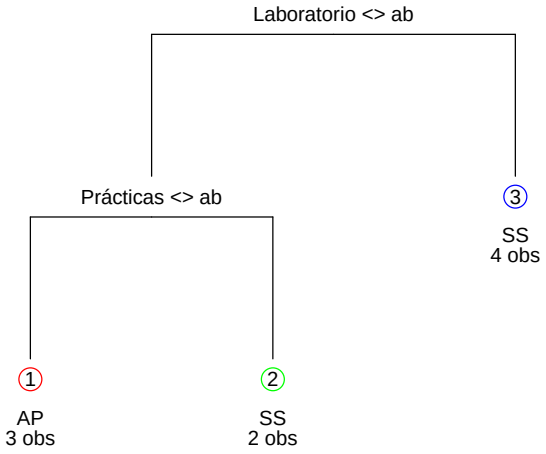
```
> muestra1 = data.frame(read.table("datos1.txt"))
> clas_1gini=rpart(Global~., data=muestra1,method="class",minsplit=1)
> rpart.plot(clas_1gini)
```




```
> clas_1entropia=rpart(Global~., data=muestra1,method="class",minsplit=1,parms=list(split=
> rpart.plot(clas_1entropia)
```




```
> clas_1tree=tree(Global~.,data=muestra1,mincut=1,minsize=2)
> draw.tree(clas_1tree)
```



2 EJ2

Creamos un .txt con los datos proporcionados sobre el radio y densidad de los planetas y lo leemos.

```
> datos2 <- read.table("datos2.txt")
> datos2
```

```
    Nombre Radio Densidad
1 Mercurio   2.4      5.4
2    Venus   6.1      5.2
3   Tierra   6.4      5.5
4    Marte   3.4      3.9
```

Calculamos la regresión sobre dichos datos para obtener la recta que más se aproxime a los puntos que tenemos.

```
> regresion2 <- lm(Densidad~Radio, data=datos2)
> regresion2_own <- regLine(datos2$Radio, datos2$Densidad)
```

Podemos ver los valores que adopta la ecucaión de la recta que se generará.
y = ax + b
b=

```
> regresion2$coefficients[1]
```

```
(Intercept)
   4.362396
```

```
> regresion2_own$coefficients[1]
```

```
[1] 4.362396
```

a=

```
> regresion2$coefficients[2]
```

```
    Radio
0.1393669
```

```
> regresion2_own$coefficients[2]
```

```
[1] 0.1393669
```

Cuando calculamos la recta de regresión sobre unos datos es necesario evaluar la calidad de esta. Debemos analizar cómo de bien se ajusta a nuestros datos. Podemos ver esta información mediante summary.

2.1 Residuos

Diferencias entre cada valor de y real y cada valor de y obtenido mediante la función de regresión.

```
> summary(regresion2)$residuals
```

```
          1           2           3           4
 0.70312301 -0.01253452  0.24565541 -0.93624389
```

2.2 Coeficientes

Coeficientes estimados para y error estándar para cada uno de ellos.

```
> summary(regresion2)$coefficients
```

```
             Estimate Std. Error   t value   Pr(>|t|)
(Intercept) 4.3623964  1.2049754 3.6203201 0.06854492
Radio       0.1393669  0.2466205 0.5651067 0.62893696
```

2.3 Error estándar

Podemos comprobar que coincide con nuestra implementación. Cuanto más próximo a 0 sea el error estándar mejor será la recta de regresión.

```
> summary(regresion2)$sigma
```

```
[1] 0.8460019
```

```
> errorEstandar(datos2$Radio, datos2$Densidad, regresion2)
```

```
    Radio
0.8460019
```

2.4 Correlación cuadrada

Podemos comprobar que coincide con nuestra implementación. Este valor está entre 0 y 1 siendo mejor cuanto más próximo a 1 sea (idealmente a partir de 0.8).

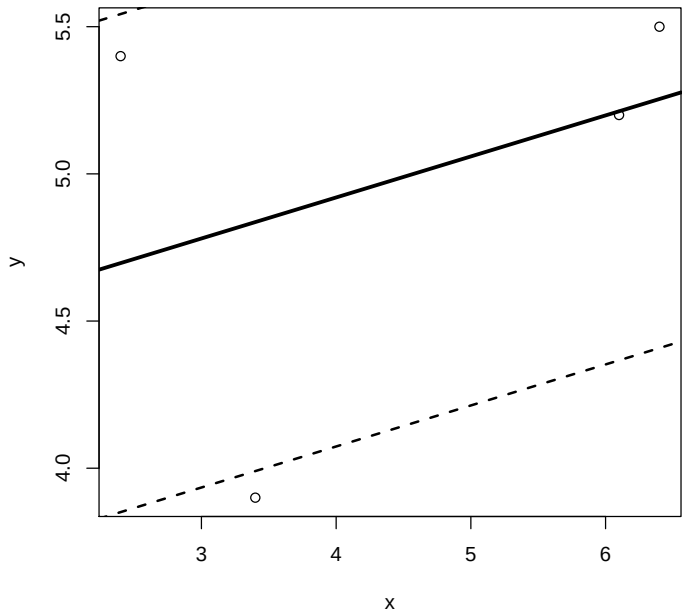
```
> summary(regresion2)$r.squared
```

```
[1] 0.1376878
```

```
> correlacionCuadrada(datos2$Radio, datos2$Densidad)
```

```
    Radio
0.1376878
```

Para finalizar dibujaremos una gráfica en la que se representarán lod datos junto a la recta de regresión. Paralela a la recta de regresión dibujaremos las rectas que marcan el error estándar entorno a la recta de regresión. En trazado gris grueso la que marca la región en la que estarán el 66% de los datos y en gris fino la que marca el 95%.

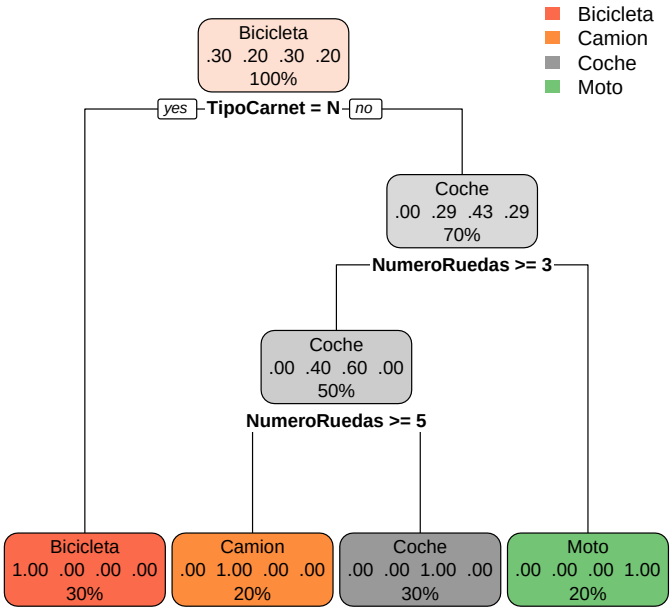


Como podemos ver la recta se ajusta muy mal a los datos que tenemos quedando las rectas que marcan el error estandas fuera del gráfico. La correlación cuadrada obtenida es muy baja. En parte esto se debe a que tenemos muy pocos datos.

3 EJ3

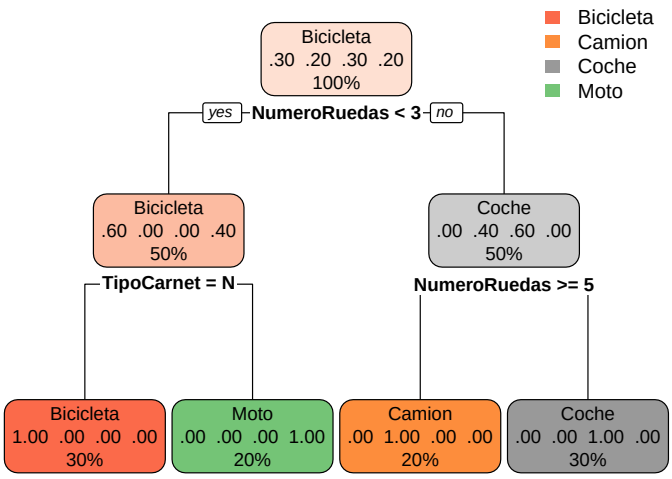
En esta parte realizamos el algoritmo Hunt con la librería rpart sobre los datos de los vehículos. Lo primero sera realizar el ejercicio usando Gini como metodo para calculart la impureza. La segunda parte la calcularemos usando la entropia. Por último utilizaremos la libreria tree para repetir estos cálculos aunque solo podremos utilizar Gini para calcular la impuraza. En ambos casos mostraremos los árboles obtenidos con las librerías adecuadas.

```
> muestra3 = data.frame(read.table("datos3.txt"))
> clas_3gini=rpart(TipoVehiculo~., data=muestra3,method="class",minsplit=1)
> rpart.plot(clas_3gini)
```



Ahora utilizaemos la entropia. Para ello añadimos el parametro parms=list(split="information"). Por defecto esta se clasifica por Gini.

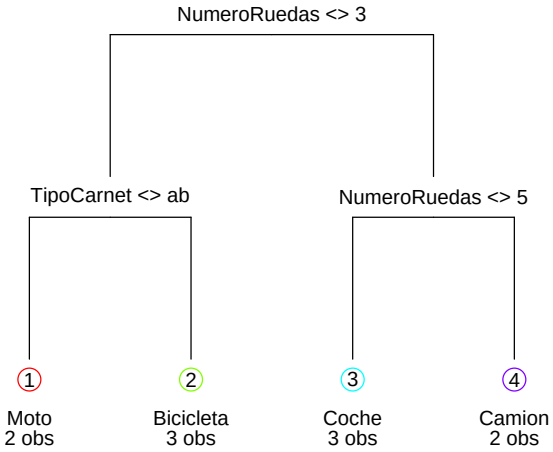
```
> clas_3entropia=rpart(TipoVehiculo~., data=muestra3,method="class",minsplit=1,parms=list(
> rpart.plot(clas_3entropia)
```



Vemos que el árbol obtenido es distinto. La primera diferencia que se observa, en el caso del arbol Gini su profundidad es de 3 mientras que con Entropia es de 2. Es preferible por tanto la clasificación utilizando entropia pues el tiempo de búsqueda en el árbol será menor.

Ahora realizamremos los mismo cálculos con tree.

```
> clas_3tree=tree(TipoVehiculo~.,data=muestra3,mincut=1,minsize=2)
> draw.tree(clas_3tree)
```



El arbol generado es igual que el obtenido con rpart con aplicando gini. Ya que por defecto tree implementa el gini. En esta libreria no esta implentado el calculo por entropia.