**FACTORES**

**¿Qué es un factor?**

Un factor es un caso especial de vector que se usa únicamente para representar variables categóricas u ordinales.

En el conjunto de datos médicos que estamos construyendo, podríamos usar un factor para representar el género, porque usa dos categorías: MASCULINO Y FEMENINO.

**¿Es importante un factor?**

Si tenemos en mente usar algoritmos de “machine o deep learning”, para entrenar nuestros equipos en clasificación de datos, debemos de prestar especial atención a este tipo de estructura, ya que nuestros algoritmos tomarán para su entrenamiento generalmente un factor. Este paso es predeterminante y realmente importante en donde todo el proyecto puede salir mal o bien si no se presta especial atención.

**¿Por qué no usar vectores?**

Una ventaja de los factores es que las etiquetas de categoría se almacenan solo una vez.

Por ejemplo, en lugar de almacenar MALE, MALE, FEMALE, la computadora puede almacenar 1, 1, 2, lo que reduce el tamaño de la memoria necesaria para almacenar la misma información.

Además, muchos algoritmos de aprendizaje automático o machine learning, tratan los datos nominales y numéricos de manera diferente.

La codificación como factores a menudo es necesaria para informar a una función de R para tratar los datos categóricos de manera adecuada.

**Importante**

No se debe usar un factor para los vectores de caracteres que no son realmente categóricos. Si un vector almacena en su mayoría valores únicos como nombres o cadenas de identificación, manténgalo como un vector.

**Creando un Factor**

Para crear un factor a partir de un vector de caracteres, simplemente aplique la función factor ().

Ejemplo:

gender **<-** factor**(**c**(**"MALE", "FEMALE", "MALE"**))**

gender

Resultado:

[1] MALE FEMALE MALE

Levels: FEMALE MALE

**Observe**

Cuando se mostraron los datos de género de John Doe y Jane Doe, R imprimió información adicional sobre el factor de género. La variable de niveles comprende el conjunto de posibles categorías que el factor podría tomar, en este caso: MASCULINO, FEMENINO.

**Niveles Adicionales**

Cuando creamos factores, podemos agregar niveles adicionales, que pueden no aparecer en los datos.

Supongamos que agregamos otro factor para el tipo de sangre, como se muestra en el siguiente ejemplo:

blood **<-** factor**(**c**(**"O", "AB", "A"**)**, levels **=** c**(**"A", "B", "AB", "O"**))**

Cuando definimos el factor sanguíneo para los tres pacientes, especificamos un vector adicional de cuatro tipos de sangre posibles usando el parámetro o argumento de levels.

Como resultado, aunque nuestros datos solo incluyeron los tipos O, AB y A, los cuatro tipos se almacenan con el factor sanguíneo como lo indica la salida.

Almacenar el nivel adicional permite la posibilidad de agregar datos con otros tipos de sangre en el futuro.

También garantiza que, si tuviéramos que crear una tabla de tipos de sangre, sabríamos que existe el tipo B, a pesar de que no se registra en nuestros datos.

**Categorías y Variables Nominales**

La estructura de datos de factores también nos permite incluir información sobre el orden de las categorías de una variable nominal, lo que proporciona una manera conveniente de almacenar datos ordinales.

Mas adelante cuando estemos viendo la parte de machine learning, vamos a poder ver como el orden de las categorías nos puede ayudar en el caso de la especificialidad y sensibilidad de un algoritmo de clasificación en una matriz de confusión.

Ejemplo:

Vamos a suponer que tenemos los datos sobre la gravedad de los síntomas de un paciente, siendo estos codificados en un nivel creciente acorde a su gravedad. Es decir, desde leve, moderada y grave.

En este caso como ya tenemos el vector identificado como categórico, vamos a enumerar los niveles en orden ascendente de menor a mayor y finalmente vamos a establecer el argumento “ordered” igual a verdadero, de esta manera queda ordenada de manera implícita la categoría de menor a mayor.

Aquí el código:

symptoms **<-** factor**(**c**(**"SEVERE", "MILD", "MODERATE"**)**,levels **=** c**(**"MILD", "MODERATE", "SEVERE"**)**, ordered **=** **TRUE)**

**Prueba Lógica**

Una característica útil que incluso es aprovechada por los algoritmos de clasificación en los factores ordenados es que las pruebas lógicas funcionarán como uno espera.

Ejemplo:

Vamos a probar si los síntomas del supuesto paciente son más que moderados.

symptoms **>** "MODERATE"

Resultado:

> symptoms > "MODERATE"

[1] TRUE FALSE FALSE

**Recordando Nuevamente**

Los algoritmos de aprendizaje automático (machine learnig) capaces de modelar datos categóricos esperarán y asumen que los factores están ordenados, así que asegúrese desde un inicio codificar sus datos en consecuencia.

Este simple paso representa en muchas ocasiones si está mal realizado, el fracaso del entrenamiento en un proyecto.

El manejo de los factores es tan importante que R tiene una librería específicamente para su manejo y se llama forcats, la cual pueden descargar del siguiente enlace:

https://posit.co/wp-content/uploads/2022/10/factors-1.pdf

**PRACTICA**

**Introducción:** Las variables de factor en R pueden ser alucinantes. A menudo, puede evitarlos y usar vectores de caracteres en su lugar, simplemente no olvide establecer stringsAsFactors = FALSE.

Sin embargo, son muy útiles en algunas circunstancias, especialmente si nos vamos a sumergir en Machine Learning, modelado estadístico y presentaciones de datos en gráficos, geometrías y tablas.

Usted puede confiar plenamente en los factores, sin embargo, mal interpretarlos puede causar un serio problema.

Estos ejercicios se presentan con el fin de realizar un aprendizaje autodidáctico efectivo, no olvide siempre estoy a las ordenes en caso tenga alguna duda de como ir resolviendo cada uno de los mismos.

Estos ejercicios tienen un nivel de dificultad de 4.5 en escala de 1-10, en algunas partes el sentido es más lógico y en otras se requiere investigar dentro de las ayudas que tiene R.

Al dominar los factores, dominará los datos categóricos, el cual le dará una clara ventaja al momento de analizar datos.

**Ejercicio 1.**

Si

x <- 1, 2, 3, 3, 5, 3, 2, 4, NA

¿Cuáles son los “levels” del factor (x)?

**Tip: cree el factor primero**

a. 1, 2, 3, 4, 5

b. NA

c. 1, 2, 3, 4, 5, NA

**Ejercicio 2.**

Si

x <- c(11, 22, 47, 47, 11, 47, 11)

Generamos la siguiente expresión en R:

factor(x, levels=c(11, 22, 47), ordered=TRUE)

¿Cuál es el 3er elemento de la salida?

a. 11

b. 22

c. 47

**Ejercicio 3.**

Si

z <- c("p", "a" , "g", "t", "b")

¿Cuál de las siguientes expresiones R reemplazará el tercer elemento en la variable z con "b"?

factor(z[3]) <- "b"

levels(z[3]) <- "b"

z[3] <- "b"

**Ejercicio 4.**

Si

z <- factor(c("p", "q", "p", "r", "q"))

Y los niveles de z son: "p", "q" ,"r", escriba una expresión en R que pueda cambiar el nivel de “p” a “w” para que z sea igual al resultado siguiente: "w", "q" , "w", "r" , "q".

**Tip:** Es una respuesta corta, observe la posición de p con respecto a los otros, utilice la función levels(), promocionando el nombre de la variable y la posición de “p”.

**Ejercicio 5.**

Aquí usaremos una función de R llamada sample(), que toma un ejemplo de un vector y set.seed para una muestra aleatoria.

Si

set.seed(1)

s1 <- factor(sample(letters, size=5, replace=TRUE))

s2 <- factor(sample(letters, size=5, replace=TRUE))

Escriba una expresión en R para concatenar s1 y s2 en un solo factor s3 con 10 elementos.

**Ejercicio 6.**

Vamos a considerar otro data set por defecto de R, en este caso es “iris”, como podemos observar podemos llamar un dataset en memoria con la función data().

data("iris")

Escribamos una expresión que nos permita cortar la variable Sepal.Length y crear el siguiente factor con sus 5 niveles “levels”.

(4.3, 5.02] (5.02, 5.74] (5.74, 6.46] (6.46, 7.18] (7.18, 7.9]

32 41 42 24 11

*“Esto es más avanzado les puede tomar algo de tiempo, pero aprendemos dos cosas, a buscar la ayuda y a pensar una solución adecuada.”*

**Tip:**Buscaremos en la ayuda de R la función table(), y cut() de esta forma:

?table()

?cut()

**Tip2:** Además nuestra primera función a utilizar es table() que contendrá la otra función cut().

**Ejercicio 7.**

Consideraremos nuevamente el dataset IRIS, como el anterior ejercicio.

Escriba una expresión R que genere una tabla de frecuencia de dos vías con dos filas y tres columnas. Las filas deben relacionarse con Sepal.length (less than 5: TRUE or FALSE) y columnas con Species, con el siguiente resultado:

setosa versicolor virginica

FALSE 30 49 49

TRUE 20 1 1

**Tip:** nuevamente utilizamos la función table

**Ejercicio 8.**

Considere el siguiente factor:

responses <- factor(c("Agree", "Agree", "Strongly Agree", "Disagree", "Agree"))

el cual le da la siguiente salida:

[1] Agree Agree Strongly Agree Disagree Agree

Levels: Agree Disagree Strongly Agree

Mas tarde encontró que existe un nuevo nivel "Strongly Disagree".

Escriba una expresión en R la cual incluya el nivel "Strongly Disagree", como un nuevo atributo del factor, además devuelva exactamente el siguiente resultado:

[1] Agree Agree Strongly Agree Disagree Agree

Levels: Strongly Agree Agree Disagree Strongly Disagree

**Ejercicio 9.**

Si

x <- data.frame(q=c(2, 4, 6), p=c("a", "b", "c"))

Escriba una instrucción en “R” que reemplace los niveles a,b,c con las etiquetas (labels), "fertiliser1", "fertliser2", "fertiliser3".

**Ejercicio 10.**

Si

x <- factor(c("high", "low", "medium", "high", "high", "low", "medium"))

Escriba una expresión en R que proporcione valores numéricos únicos para varios niveles de x con la siguiente salida:

levels value

1 high 1

2 low 2

3 medium 3

**Tip:** En el ejercicio 9 y 10 utilizamos la función de data.frame() aunque no las hemos visto todavía si las puede resolver es un plus.