

## 18.- Feature Construction\_CU\_18\_20\_infra\_meteo\_v\_01

June 13, 2023

#

CU18\_Infraestructuras\_eventos

Citizenlab Data Science Methodology > III - Feature Engineering Domain \*\*\* > # 18.- Feature Construction

Feature Construction is the process related to create new features from your existing ones to improve model performance.

### 0.1 Tasks

Feature Construction - Create Interaction Features - Create derived variables - Combine Sparse Classes - Explore Binning for Feature Construction

### 0.2 Consideraciones casos CitizenLab programados en R

- Algunas de las tareas de este proceso se han realizado en los notebooks del proceso 05 Data Collection porque eran necesarias para las tareas ETL. En esos casos, en este notebook se referencia al notebook del proceso 05 correspondiente
- Otras tareas típicas de este proceso se realizan en los notebooks del dominio IV al ser más eficiente realizarlas en el propio pipeline de modelización.
- Por tanto en los notebooks de este proceso de manera general se incluyen las comprobaciones necesarias, y comentarios si procede
- Las tareas del proceso se van a aplicar solo a los archivos que forman parte del despliegue, ya que hay muchos archivos intermedios que no procede pasar por este proceso
- El nombre de archivo del notebook hace referencia al nombre de archivo del proceso 05 al que se aplica este proceso, por eso pueden no ser correlativa la numeración
- Las comprobaciones se van a realizar teniendo en cuenta que el lenguaje utilizado en el despliegue de este caso es R

### 0.3 File

- Input File: CU\_18\_09.3\_20\_diario\_infra
- Output File: No aplica

#### 0.3.1 Encoding

Con la siguiente expresión se evitan problemas con el encoding al ejecutar el notebook. Es posible que deba ser eliminada o adaptada a la máquina en la que se ejecute el código.

```
[1]: Sys.setlocale(category = "LC_ALL", locale = "es_ES.UTF-8")
```

```
'LC_CTYPE=es_ES.UTF-8;LC_NUMERIC=C;LC_TIME=es_ES.UTF-8;LC_COLLATE=es_ES.UTF-8;LC_MONETARY=es_ES.UTF-8;LC_MESSAGES=en_US.UTF-8;LC_PAPER=es_ES.UTF-8;LC_NAME=C;LC_ADDRESS=C;LC_TELEPHONE=C;LC_MEASUREMENT=C;LC_IDENTIFICATION=C'
```

## 0.4 Settings

### 0.4.1 Libraries to use

```
[2]: library(readr)
library(dplyr)
library(tidyr)
library(forcats)
library(lubridate)
```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

filter, lag

The following objects are masked from 'package:base':

intersect, setdiff, setequal, union

Attaching package: 'lubridate'

The following objects are masked from 'package:base':

date, intersect, setdiff, union

### 0.4.2 Paths

```
[3]: iPath <- "Data/Input/"
oPath <- "Data/Output/"
```

## 0.5 Data Load

OPCION A: Seleccionar fichero en ventana para mayor comodidad

Data load using the {tcltk} package. Uncomment the line if using this option

```
[4]: # file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

```
[5]: iFile <- "CU_18_09.3_20_diario_infra.csv"
file_data <- paste0(iPath, iFile)

if(file.exists(file_data)){
  cat("Se leerán datos del archivo: ", file_data)
} else{
  warning("Cuidado: el archivo no existe.")
}
```

Se leerán datos del archivo: Data/Input/CU\_18\_09.3\_20\_diario\_infra.csv

**Data file to dataframe** Usar la función adecuada según el formato de entrada (xlsx, csv, json, ...)

```
[6]: data <- read_csv(file_data)
```

Rows: 377727 Columns: 10

Column specification

Delimiter: ","

dbl (9): id\_inf, capacidad, demanda, evento\_infra, evento\_zona, tmed,  
prec,...  
date (1): fecha

Use `spec()` to retrieve the full column specification for this data.

Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

Estructura de los datos:

```
[7]: data |> glimpse()
```

Rows: 377,727

Columns: 10

\$ id\_inf <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,  
13, 14, 15, 16, 17...

\$ fecha <date> 2019-01-01, 2019-01-01, 2019-01-01,  
2019-01-01, 2019-01-...

\$ capacidad <dbl> 993, 996, 1036, 1020, 992, 1026, 1007,

```

976, 1037, 972, 94...
$ demanda      <dbl> 883, 888, 922, 1134, 1103, 1139, 897,
1086, 1150, 861, 83...
$ evento_infra <dbl> 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0,
0, 0, 1, 0, 1, 1, ...
$ evento_zona  <dbl> 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
1, 1, 1, 1, 0, 1, ...
$ tmed         <dbl> 6.953211, 6.196420, 6.483569, 5.875797,
6.212680, 5.87854...
$ prec         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, ...
$ velmedia     <dbl> 0.6433886, 0.3417523, 0.4132169,
0.1820178, 0.2118110, 0...
$ presMax      <dbl> 952.9357, 950.2191, 950.8051, 951.6768,
953.5118, 952.168...

```

Muestra de los primeros datos:

```
[8]: data |> slice_head(n = 5)
```

	id_inf <dbl>	fecha <date>	capacidad <dbl>	demanda <dbl>	evento_infra <dbl>	evento_zona <dbl>	tmed <dbl>	p <dbl>
A spec_tbl_df: 5 × 10	1	2019-01-01	993	883	1	1	6.953211	0
	2	2019-01-01	996	888	0	0	6.196420	0
	3	2019-01-01	1036	922	0	0	6.483569	0
	4	2019-01-01	1020	1134	1	0	5.875797	0
	5	2019-01-01	992	1103	1	1	6.212680	0

## 0.6 Creating Interaction Features

## 0.7 Creating derived variables

## 0.8 Combining Sparse Classes

## 0.9 Binning for Feature Construction

## 0.10 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

```
[9]: data_to_save <- data
```

Estructura de nombre de archivos:

- Código del caso de uso, por ejemplo “CU\_04”
- Número del proceso que lo genera, por ejemplo “\_06”.
- Resto del nombre del archivo de entrada
- Extensión del archivo

Ejemplo: "CU\_04\_06\_01\_01\_zonasgeo.json, primer fichero que se genera en la tarea 01 del proceso 05 (Data Collection) para el caso de uso 04 (vacunas) y que se ha transformado en el proceso 06

Importante mantener los guiones bajos antes de proceso, tarea, archivo y nombre

### 0.10.1 Proceso 18

```
[10]: caso <- "CU_18"  
      proceso <- '_18'  
      tarea <- "_20"  
      archivo <- ""  
      proper <- "_diario_infra"  
      extension <- ".csv"
```

OPCION A: Uso del paquete "tcltk" para mayor comodidad

- Buscar carpeta, escribir nombre de archivo SIN extensión (se especifica en el código)
- Especificar sufixo2 si es necesario
- Cambiar datos por datos\_xx si es necesario

```
[11]: # file_save <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper, ↵  
      ↵extension)  
      # path_out <- paste0(oPath, file_save)  
      # write_csv(data_to_save_XXXXX, path_out)  
  
      # cat('File saved as: ')  
      # path_out
```

OPCION B: Especificar el nombre de archivo

- Los ficheros de salida del proceso van siempre a Data/Output/.

```
[12]: # file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)  
      # path_out <- paste0(oPath, file_save)  
      # write_csv(data_to_save_XXXXX, path_out)  
  
      # cat('File saved as: ')  
      # path_out
```

**Copia del fichero a Input** Si el archivo se va a usar en otros notebooks, copiar a la carpeta Input

```
[13]: # path_in <- paste0(iPath, file_save)  
      # file.copy(path_out, path_in, overwrite = TRUE)
```

## 0.11 REPORT

A continuación se realizará un informe de las acciones realizadas

## 0.12 Main Actions Carried Out

- Si eran necesarias se han realizado en el proceso 05 por cuestiones de eficiencia
- En el proceso 05 se realizó un cluster, y los pasos en este notebook no aplican
- O bien se hacen en el dominio IV o V para integrar en el pipeline de modelización

## 0.13 Main Conclusions

- Los datos están listos para la modelización y despliegue

## 0.14 CODE TO DEPLOY (PILOT)

A continuación se incluirá el código que deba ser llevado a despliegue para producción, dado que se entiende efectúa operaciones necesarias sobre los datos en la ejecución del prototipo

Description

- No hay nada que desplegar en el piloto, ya que estos datos son estáticos o en todo caso cambian con muy poca frecuencia, altamente improbable durante el proyecto.

CODE

[ ]: