14.- Feature Data Transform_CU_18_20_infra_meteo_v_01

June 13, 2023

#

CU18_Infraestructuras_eventos

Citizenlab Data Science Methodology > III - Feature Engineering Domain *** > # 14.- Feature Data Transform

Feature Data Transform is the process that allows change (if is required) the type and/or distribution of data features (e.g. scaling, normalizing o standardizing data features).

0.1 Tasks

Perform Basic Data Transforms

Perform Categorical Variable Transformation
- Encode Transformation
- One-hot encoding
- Ordinal encoding
- Dummy encoding
- Evaluate a Logistic Regression model
- Consider Embedding if text mining context

Perform Numeric Variable Transformation
- Scale Transformation
- Normalization
- Standardization
- IQR Robust Scaler Transform
- Evaluate a KNN model
- Distribution Transformation
- Discretization
- Uniform
- Clustered(k-Means)
- Quantile
- Normal Quantile
- Uniform Quantile
- Evaluate a KNN model
- Evaluate a KNN model
- Power transforms (Make Distributions More Gaussian)
- Box-Cox Transform
- Yeo-Johnson Transform
- Evaluate a KNN model

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=es_ES.UTF-8;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. Ucomment 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)
```

A spec_tbl_df: 5 × 10

id_inf <dbl>	fecha <date>	capacidad <dbl>	demanda <dbl>	evento_infra <dbl>	evento_zona <dbl>	tmed <dbl>	p <dbl>
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 Basic Data Transforms

0.6.1 Data Selecting

[9]:
```
data |> select(1)
```

id_inf
<dbl>
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128

A tibble: 377727 × 1

0.6.2 Data Filtering

[12]:
```
# data |> filter(ano = 2019)
```

0.6.3 Insert New Column

[13]:
```
data |>
    mutate(x = TRUE)
```

id_inf <dbl>	fecha <date>	capacidad <dbl>	demanda <dbl>	evento_infra <dbl>	evento_zona <dbl>	tmed <dbl>	pr <dbl>
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
6	2019-01-01	1026	1139	0	1	5.878542	0
7	2019-01-01	1007	897	0	0	6.043628	0
8	2019-01-01	976	1086	1	0	6.716287	0
9	2019-01-01	1037	1150	1	1	6.541842	0
10	2019-01-01	972	861	0	1	7.274224	0
11	2019-01-01	940	837	1	0	6.651713	0
12	2019-01-01	1000	1109	1	0	5.859384	0
13	2019-01-01	993	881	0	0	5.870729	0
14	2019-01-01	1014	901	0	1	6.269831	0
15	2019-01-01	990	884	0	1	6.037552	0
16	2019-01-01	989	1094	1	1	6.184667	0
17	2019-01-01	993	1103	0	1	7.205027	0
18	2019-01-01	980	1091	1	0	6.614984	0
19	2019-01-01	985	1093	1	1	6.805965	0
20	2019-01-01	974	1084	1	0	5.999183	0
21	2019-01-01	1017	910	0	0	6.429136	0
22	2019-01-01	953	848	1	1	6.425044	0
23	2019-01-01	982	871	0	0	6.700994	0
24	2019-01-01	1020	1132	1	0	6.748697	0
25	2019-01-01	1042	1155	1	1	5.879139	0
26	2019-01-01	1054	1169	1	1	7.455467	0
27	2019-01-01	1004	1116	1	0	5.222163	0
28	2019-01-01	973	868	0	1	5.105823	0
29	2019-01-01	1041	929	1	1	5.410022	0
A tibble: 377727 × 11 30	2019-01-01	978	872	0	0	5.727610	0
1109	2019-12-31	1026	1139	1	1	6.303931	0
1110	2019-12-31	1002	1111	1	0	6.573342	0
1111	2019-12-31	989	879	0	0	6.885949	0
1112	2019-12-31	973	865	0	1	6.856195	0
1113	2019-12-31	939	836	0	0	6.341240	0
1114	2019-12-31	1056	941	1	1	6.639703	0
1115	2019-12-31	939	1041	1	0	6.565606	0
1116	2019-12-31	1034	1145	0	1	6.309946	0
1117	2019-12-31	997	889	1	1	6.803234	0
1118	2019-12-31	1041	1154	1	1	6.586504	0
1119	2019-12-31	1013	1122	1	1	6.768358	0
1120	2019-12-31	999	890	0	0	6.580942	0
1121	2019-12-31	1060	940	0	0	6.506678	0
1122	2019-12-31	959	1064	1	1	6.630664	0
1123	2019-12-31	974	1080	1	1	6.238223	0
1124	2019-12-31	971	1080	1	1	6.079000	0
1125	2019-12-31	1037	1151	1	1	6.728930	0
1126	2019-12-31	963	1069	1	0	6.513794	0
1127	2019-12-31	1037	1152	1	1	6.465050	0
1128	2019-12-31	966	1071	1	0	6.682220	0

0.6.4 Delete Column

[15]:
```
# data |> select(-x)
```

0.6.5 Replace Values

[ ]:
```
# data |> mutate(x = case_when(...))
```

0.6.6 Rank Data

Operation

[ ]:
```
# data |> mutate(rank = order(n_vacunas))
```

0.7 Categorical Variable Transformation

0.7.1 Encode Transformation

[ ]:
```
# data |> mutate(x = if_else(x = 'Yes', 1, 0))
```

Ordinal Encoding Transform

[ ]:
```
# data |> mutate(x = fct_reorder(...))
```

One Hot Encoding Transform

Dummy Variable Encoding Transform

0.7.2 Embedding Transformation

Specific encode for text mining context. No code here.

0.8 Numeric Variable Transformation: Scale

0.8.1 Data to Transform

Evaluating Normalization Tranform

Evaluating Standarization Transform

0.8.2 Normalization Transform

Select columns

[ ]:
```
# vars <- c()
```

Operation

[ ]:
```

```

0.8.3 Standarization Transform

Select columns

[ ]:
```
# vars <- c()
```

Operation

[ ]:
```
# data |> mutate(across(vars), scale)
```

0.9 Numeric Variable Transformation: Distribution

0.9.1 Discretization Transform

Evaluating Discretization Transformations

Uniform Discretization Transform Select columns

[ ]:
```
# retrieve just the numeric input values
# vars <- c(...)
```

Operation

[ ]:
```
# data |> mutate(across(vars), cut(breaks = 10))
```

0.9.2 Power Transform

Data to Transform

Evaluating Box-Cox tranform

Evaluating Yeo-Johnson tranform

Box-Cox Transform Select columns

Operation

[ ]:
```

```

Yeo-Johnson Transform Select columns

Operation

0.10 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

[16]:
```
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 14

[17]:
```
caso <- "CU_18"
proceso <- '_014'
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 sufijo2 si es necesario
- Cambiar datos por datos_xx si es necesario

[18]:
```
# file_save <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
 ↪extension)
# path_out <- paste0(oPath, file_save)
# write_csv(data_to_save_xxxx, 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/.

[19]:
```
# file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
# path_out <- paste0(oPath, file_save)
# write_csv(data_to_save_xxxx, 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

[20]:
```
# 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
- 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

[ ]:
```
```