14.- Feature Data Transform_CU_53_02_spi_v_01

June 13, 2023

#

CU53_impacto de las políticas de inversión en sanidad, infraestructuras y promoción turística en el SPI

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_53_09.2_02_spi
- Output File: CU_53_14_02_spi

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_MEASUREMENT8;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

[4]:
```
iFile <- "CU_53_09.2_02_spi.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_53_09.2_02_spi.csv
```

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

[5]:
```
data <- read_csv(file_data)
```

```
Rows: 2028 Columns: 18
  Column specification



Delimiter: ","
dbl (17): rank_score_spi, score_spi, score_bhn, score_fow, score_opp,
score_…
lgl  (1): is_train

 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:

[6]:
```
data |> glimpse()
```

```
Rows: 2,028
Columns: 18
$ rank_score_spi <dbl> 80, 97, 46, 84, 99, 150, 74, 105, 36,
143, 154, 69, 168…
$ score_spi      <dbl> 67.59, 60.10, 73.96, 62.86, 61.43,
45.57, 66.56, 59.45,…
$ score_bhn      <dbl> 79.16, 74.55, 81.88, 79.45, 77.84,
47.15, 80.41, 66.16,…
$ score_fow      <dbl> 65.40, 51.25, 70.69, 61.22, 57.63,
45.21, 62.82, 54.62,…
$ score_opp      <dbl> 58.22, 54.49, 69.32, 47.92, 48.83,
44.34, 56.46, 57.56,…
$ score_nbmc     <dbl> 86.67, 72.88, 86.33, 83.91, 87.72,
54.66, 92.38, 72.21,…
$ score_ws       <dbl> 86.44, 83.35, 88.07, 77.71, 78.15,
47.82, 78.47, 66.32,…
$ score_sh       <dbl> 87.69, 77.17, 89.59, 85.11, 86.61,
36.59, 85.21, 75.91,…
$ score_ps       <dbl> 55.85, 64.81, 63.55, 71.08, 58.87,
49.53, 65.57, 50.21,…
$ score_abk      <dbl> 74.20, 47.04, 89.07, 65.15, 55.79,
50.36, 81.61, 68.71,…
$ score_aic      <dbl> 74.19, 37.15, 68.14, 51.25, 78.17,
33.84, 61.95, 56.61,…
$ score_hw       <dbl> 53.55, 64.58, 61.41, 62.00, 45.35,
36.99, 61.64, 41.87,…
$ score_eq       <dbl> 59.66, 56.22, 64.13, 66.47, 51.22,
```

```
59.66, 46.07, 51.28,…
$ score_pr       <dbl> 81.60, 71.05, 90.28, 61.56, 60.41,
69.20, 70.02, 74.13,…
$ score_pfc      <dbl> 60.29, 64.77, 67.65, 56.51, 58.62,
40.61, 62.49, 59.83,…
$ score_incl     <dbl> 40.24, 56.12, 68.48, 48.70, 35.57,
41.81, 36.89, 55.73,…
$ score_aae      <dbl> 50.73, 26.03, 50.87, 24.90, 40.72,
25.72, 56.45, 40.54,…
$ is_train       <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE, TRUE, TRUE, T…
```

Muestra de los primeros datos:

[7]:
```
data |> slice_head(n = 5)
```

A spec_tbl_df: 5 × 18

rank_score_spi <dbl>	score_spi <dbl>	score_bhn <dbl>	score_fow <dbl>	score_opp <dbl>	score_nbmc <dbl>	score_ <dbl>
80	67.59	79.16	65.40	58.22	86.67	86.44
97	60.10	74.55	51.25	54.49	72.88	83.35
46	73.96	81.88	70.69	69.32	86.33	88.07
84	62.86	79.45	61.22	47.92	83.91	77.71
99	61.43	77.84	57.63	48.83	87.72	78.15

0.6 Basic Data Transforms

0.6.1 Data Selecting

[8]:
```
data |> select(1)
```

rank_score_spi <dbl>
80
97
46
84
99
150
74
105
36
143
154
69
168
141
164
1
64
75
125
86
27
81
144
102
30
126
25
78
129
123

A tibble: 2028 × 1

56
81
12
16
19
19
20
23
24
38
39
108
103
99
90
86
84
92
92
91

0.6.2 Data Filtering

[9]:
```
# data |> filter(ano = 2019)
```

0.6.3 Insert New Column

[10]:
```
data |>
    mutate(x = TRUE)
```

rank_score_spi <dbl>	score_spi <dbl>	score_bhn <dbl>	score_fow <dbl>	score_opp <dbl>	score_nbmc <dbl>	score_ws
80	67.59	79.16	65.40	58.22	86.67	86.44
97	60.10	74.55	51.25	54.49	72.88	83.35
46	73.96	81.88	70.69	69.32	86.33	88.07
84	62.86	79.45	61.22	47.92	83.91	77.71
99	61.43	77.84	57.63	48.83	87.72	78.15
150	45.57	47.15	45.21	44.34	54.66	47.82
74	66.56	80.41	62.82	56.46	92.38	78.47
105	59.45	66.16	54.62	57.56	72.21	66.32
36	79.93	88.01	77.07	74.71	91.17	94.88
143	44.25	54.20	46.20	32.34	66.70	55.90
154	46.58	44.86	48.63	46.24	55.91	30.89
69	67.71	80.42	69.13	53.59	91.56	82.05
168	32.39	28.96	33.29	34.91	36.38	21.97
141	48.89	63.31	48.66	34.69	74.23	63.23
164	37.97	56.86	35.38	21.67	60.12	54.29
1	90.53	91.18	90.79	89.61	93.87	98.79
64	70.85	79.02	71.16	62.36	85.50	83.44
75	66.15	68.20	62.45	67.78	73.34	71.66
125	47.13	44.10	46.95	50.33	57.05	41.28
86	65.13	75.15	61.19	59.06	77.30	83.87
27	82.26	92.60	83.01	71.16	95.39	98.89
81	63.98	76.05	61.00	54.90	84.32	79.66
144	48.89	52.80	54.34	39.54	74.38	56.89
102	57.05	78.17	50.37	42.62	91.17	80.63
30	79.96	89.24	75.67	74.96	95.15	89.88
126	54.08	54.01	57.65	50.59	66.19	55.23
25	83.02	87.77	81.22	80.07	96.34	90.46
78	66.45	83.38	61.98	54.01	94.83	83.59
129	47.46	42.28	43.51	56.60	48.33	41.90
123	54.34	67.05	53.59	42.39	74.20	64.51
56	71.89	79.38	69.84	66.44	88.35	81.83
81	63.19	83.38	57.98	48.21	91.25	94.06
12	86.65	89.32	87.27	83.38	92.87	96.17
16	86.89	89.12	87.75	83.79	92.94	94.28
19	85.42	87.59	82.68	86.00	93.02	93.87
19	85.21	87.42	82.39	85.83	93.07	93.44
20	84.99	87.28	83.27	84.42	93.10	93.35
23	85.14	87.22	83.95	84.26	93.13	93.26
24	85.04	87.42	84.15	83.54	93.18	94.01
38	77.64	84.48	70.93	77.52	89.20	95.08
39	77.67	84.17	70.81	78.02	89.30	93.78
108	59.89	85.47	57.32	36.88	89.70	93.73
103	61.50	85.80	57.62	41.07	89.97	92.78
99	63.99	86.05	59.60	46.30	90.45	91.68
90	63.82	74.66	65.38	51.44	87.37	79.54
86	63.43	78.04	60.71	51.53	84.45	82.19
84	65.18	79.22	64.56	51.77	85.29	84.22
92	62.07	76.50	56.97	52.73	90.33	69.58
92	63.03	77.07	60.16	51.85	90.76	71.19
91	64.36	77.05	63.10	52.94	91.22	71.28

A tibble: 2028 × 19

0.6.4 Delete Column

[11]:
```
# data |> select(-x)
```

0.6.5 Replace Values

[12]:
```
# data |> mutate(x = case_when(...))
```

0.6.6 Rank Data

Operation

[13]:
```
# data |> mutate(rank = order(n_vacunas))
```

0.7 Categorical Variable Transformation

0.7.1 Encode Transformation

[14]:
```
# data |> mutate(x = if_else(x = 'Yes', 1, 0))
```

Ordinal Encoding Transform

[15]:
```
# 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

[16]:
```
vars <- sapply(data, is.numeric)
vars["rank_score_spi"] <- FALSE
```

Operation

[32]:
```
data[vars] <- lapply(data[vars], scale)

data[vars] <- lapply(data[vars], function(column) {
    as.numeric(scale(column))
})
```

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

[33]:
```
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

[34]:
```
caso <- "CU_53"
proceso <- '_14'
tarea <- "_02"
archivo <- ""
proper <- "_spi"
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

[35]:
```
# 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/.

[36]:
```
file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out <- paste0(oPath, file_save)
```

```
write_csv(data_to_save, path_out)

cat('File saved as: ')
path_out
```

```
File saved as:
```

'Data/Output/CU_53_14_02_spi.csv'

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

[37]:
```
path_in <- paste0(iPath, file_save)
file.copy(path_out, path_in, overwrite = TRUE)
```

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

[ ]:
```
```