09.3.- Data Cleansing-Outliers_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 > II - Data Processing Domain *** > # 09.3.- Data Cleansing - Outliers

Data Cleaning refers to identifying and correcting (or removing) errors in the dataset that may negatively impact a predictive model, replacing, modifying, or deleting the dirty or coarse data.

0.1 Tasks

- Basic operations
  - Text data analysis
  - Delete Needless/Irrelevant/Private Columns
  - Inconsistent Data.
  - Expected values
  - Zeroes
  - Columns with a Single Value
  - Columns with Very Few Values
  - Columns with Low Variance
  - Duplicates (rows/samples) & (columns/features) Data
- Missing Values
  - Missing Values Identification
    - Missing Values Per Sample
    - Missing Values Per Feature
    - Zero Missing Values
    - Other Missing Values
    - Null/NaN Missing Values
  - Delete Missing Values
    - Deleting Rows with Missing Values in Target Column
    - Deleting Rows with Missing Values
    - Deleting Features with some Missing Values
    - Deleting Features using Rate Missing Values
  - Basic Imputation
    - Imputation by Previous Row Value
    - Imputation by Next Row Value
  - Statistical Imputation
    - Selection of Imputation Strategy
    - Constant Imputation
    - Mean Imputation
    - Median Imputation
    - Most Frequent Imputation
    - Interpolation Imputation
  - Prediction Imputation (KNN Imputation)
    - Evaluating k-hyperparmeter in KNN Imputation
    - Applying KNN Imputation
  - Iterative Imputation
    - Evaluating Different Imputation Order
    - Applying Iterative Imputation
- Outliers
  - Outliers - Univariate
    - Visualizing Outliers
      - Distribution
      - Box Plots
      - Isolation Forest
    - Outliers Identification
      - Grubbs' Test
      - Z-Score
      - Standard Deviation Method
      - Interquartile Range Method
      - Tukey's method
      - Internally studentized residuals AKA z-score method
      - Median Absolute Deviation method
  - Outliers - MultiVariate
    - Visualizing Outliers
      - ScatterPlots
    - Outliers Identification
      - Mahalanobis Distance
      - Robust Mahalanobis Distance
      - DBSCAN Clustering
      - PyOD Library
  - Automatic Detection and Removal of Outliers
    - Compare Algorithms
    - LocalOutlierFactor
    - IsolationForest
    - Minimum Covariance Determinant

0.2 Consideraciones casos CitizenLab programados en R

- La mayoría 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
- 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: 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
8;LC_IDENTIFICATION=C'

0.4 Settings

0.4.1 Libraries to use

[2]:
```
library(readr)
library(dplyr)
# library(sf)
library(tidyr)
library(stringr)
library(ggplot2)
```

```
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
```

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_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, …)

[6]:
```
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.
```

Visualizo los datos.

Estructura de los datos:

[7]:
```
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:

```
[8]: 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 Outliers - Univariate

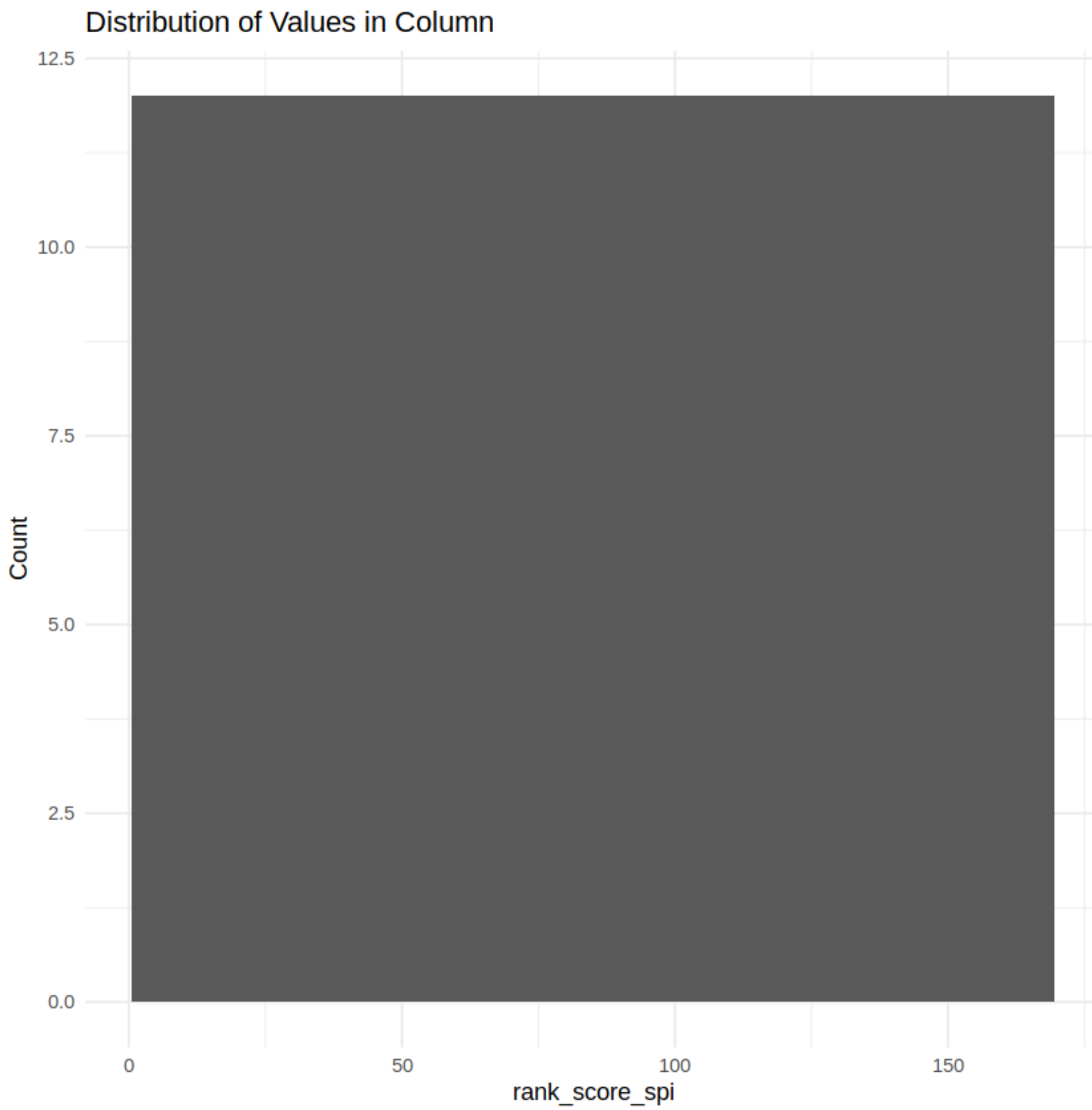
0.6.1 Visualizing Outliers

Distribution Selecting feature to analyze

```
[9]: # Selecting feature to analyze
column_name <- "rank_score_spi"
```

Operation

```
[10]: # Create a histogram plot
ggplot(data, aes(x = data[[column_name]])) +
  geom_histogram(binwidth = 1) +
  labs(x = column_name, y = "Count") +
  ggtitle("Distribution of Values in Column") +
  theme_minimal()
```



Box Plots Are great to summarize and visualize the distribution of variables easily and quickly.

[ ]:

```

```

Selecting feature to analyze

[11]:

```
# Selecting feature to analyze
column_name <- "rank_score_spi"
```

Operation

[12]:

```
# Analyze specifics features
# Create a box plot
```

```
ggplot(data, aes(x = "", y = data[[column_name]])) +
  geom_boxplot() +
  labs(x = "", y = column_name) +
  ggtitle("Box Plot of Column") +
  theme_minimal()
```

Isolation Forest Selecting feature to analyze

[13]:
```
# Selecting feature to analyze
```

Operation

[ ]:
```
```

0.6.2 Outliers Identification

Grubbs' Test Selecting feature to analyze

[14]:
```
# Selecting feature to analyze
column_name <- "rank_score_spi"
```

Operation

[15]:
```
library(outliers)
outliers <- grubbs.test(data[[column_name]], opposite = FALSE)
print(outliers)
```

```
        Grubbs test for one outlier

data:  data[[column_name]]
G = 1.72141, U = 0.99854, p-value = 1
alternative hypothesis: highest value 169 is an outlier
```

Z-Score Selecting feature to analyze

[16]:
```
# Selecting feature to analyze
column_name <- "rank_score_spi"
```

Operation

[ ]:
```

```

[17]:
```
# Define a threshold to identify an outlier.
# List of row numbers with outlier
# Choose the numeric column from your data

# Calculate the z-score
z_scores <- data %>% select(all_of(column_name)) %>% scale()

# Define a threshold for identifying outliers (e.g., z-score > 5 or z-score <
 ↪-5)
threshold <- 5

# Find the row numbers with z-scores exceeding the threshold
outlier_rows <- which(abs(z_scores) > threshold)

# Print the row numbers with outliers
print(outlier_rows)

data_cleaned <- subset(data, !(row.names(data) %in% outlier_rows))
```

```
integer(0)
```

Standard Deviation Method Selecting feature to analyze

[18]:
```
# Selecting all features to analyze
column_name <- "rank_score_spi"
```

Operation

[ ]:
```

```

[19]:
```
# # identify outliers with standard deviation
# # Choose the numeric column from your data
# column_name_df <- data[[column_name]]

# # Calculate the mean and standard deviation of the column
# column_mean <- mean(column_name_df)
# column_sd <- sd(column_name_df)

# # Define the threshold as a multiple of the standard deviation (e.g., 3 times␣
↪the standard deviation)
# threshold <- 3

# # Identify the outliers based on the threshold
# outlier_rows <- which(column_name_df > (column_mean + threshold * column_sd)␣
↪| column_name_df < (column_mean - threshold * column_sd))

# # Print the updated column with outliers removed
# print(outlier_rows)

# data_cleaned <- subset(data, !(row.names(data) %in% outlier_rows))
```

Interquartile Range Method Selecting factor k

[20]:
```
# Selecting factor k
column_name <- "rank_score_spi"
threshold <- 1.5
```

Operation

[21]:
```
# identify outliers with standard deviation
# Choose the numeric column from your data
# column_name_df <- data[[column_name]]

# # Calculate the first quartile (Q1) and third quartile (Q3)
# Q1 <- quantile(column_name_df, 0.25)
# Q3 <- quantile(column_name_df, 0.75)
```

```
# # Calculate the IQR (Interquartile Range)
# IQR <- Q3 - Q1

# # Identify the outliers based on the threshold
# outlier_rows <- which(column_name_df < (Q1 - threshold * IQR) |
↪column_name_df > (Q3 + threshold * IQR))
# print(outlier_rows)
# data_cleaned <- subset(data, !(row.names(data) %in% outlier_rows))
```

Tukey's method

[22]:
```
#Tukey's method
# column_name <- "presMax"

# column_name_df <- data[[column_name]]
# # Calculate the first quartile (Q1) and third quartile (Q3)
# Q1 <- quantile(column_name_df, 0.25)
# Q3 <- quantile(column_name_df, 0.75)

# # Calculate the interquartile range (IQR)
# IQR <- Q3 - Q1

# # Define the multiplier for Tukey's method (e.g., 1.5 times the IQR)
# multiplier <- 1.5

# # Calculate the lower and upper bounds for outliers
# lower_bound <- Q1 - multiplier * IQR
# upper_bound <- Q3 + multiplier * IQR

# # Identify the outliers based on the bounds
# outlier_rows <- which(column_name_df < lower_bound | column_name_df >
↪upper_bound)

# # Print the updated column with outliers removed
# print(outlier_rows)
# data_cleaned <- subset(data, !(row.names(data) %in% outlier_rows))
```

Internally studentized residuals AKA z-score method

[23]:
```
#Internally studentized method (z-score)
```

Median Absolute Deviation method

[24]:
```
#MAD method
# column_name <- "presMax"

# column_name_df <- data[[column_name]]
# # Calculate the median absolute deviation (MAD)
```

```
# mad <- median(abs(column_name_df - median(column_name_df, na.rm = TRUE)), na.
↪rm = TRUE)

# # Define a threshold for identifying outliers (e.g., 3 times the MAD)
# threshold <- 3 * mad

# # Identify the outliers based on the MAD
# outliers <- which(abs(column_name_df - median(column_name_df, na.rm = TRUE))␣
↪> threshold)

# # Print the updated column with outliers removed
# print(outliers)
# data_cleaned <- subset(data, !(row.names(data) %in% outliers))
```

0.7 Outliers - MultiVariate

0.7.1 Visualizing Outliers

ScatterPlots: a common way to plot multivariate outliers is the scatter plot.

ScatterPlots A common way to plot multivariate outliers is the scatter plot.

[ ]:

```

```

Selecting feature to analyze

[25]:

```
# Selecting feature to analyze
```

Operation

[ ]:

```

```

0.7.2 Outliers Identification

Mahalanobis Distance

[ ]:

```

```

Selecting feature to analyze

[26]:

```
# Selecting features to analyze
```

Operation

[27]:

```
# Analyze selected features
```

[28]:

```
# Analyze all dataset
```

Robust Mahalanobis Distance

[29]:
```
#Robust Mahalonibis Distance
```

Selecting feature to analyze

[30]:
```
# Selecting features to analyze
```

Operation

[31]:
```
# Analyze selected features
```

[32]:
```
# Analyze all dataset
```

DBSCAN Clustering Selecting feature to analyze

[33]:
```
# Selecting feature to analyze
```

Operation

[ ]:
```

```

[34]:
```
# specify & fit model
```

[35]:
```
# visualize outputs
```

[36]:
```
# outliers dataframe
```

[37]:
```
# Index of rows with outliers
```

[38]:
```
# Outliers Dataframe
```

[ ]:
```

```

0.8 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

[39]:
```
data_to_save <- data_cleaned
```

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

0.8.1 Proceso 09.3

[40]:
```
caso <- "CU_53"
proceso <- '_09.3'
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

[41]:
```
# 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/.

[42]:
```
# 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
```

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

[43]:
```
# path_in <- paste0(iPath, file_save)
# file.copy(path_out, path_in, overwrite = TRUE)
```

0.9 REPORT

A continuación se realizará un informe de las acciones realizadas

0.10 Main Actions Carried Out

- Si eran necesarias se han realizado en el proceso 05 por cuestiones de eficiencia

0.11 Main Conclusions

- Los datos están limpios para el despliegue

0.12 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

[ ]:

```

```