09.1.- Data Cleansing-Basic_05_servicios_completo_v_01

June 16, 2023

#

CUxx_Nombre del caso de uso

Citizenlab Data Science Methodology > II - Data Processing Domain *** > # 09.1.- Data Cleansing - Basic

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: xxxxxxxxxx
- Output File: No aplica

0.4 Settings

0.4.1 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.

[8]:
```
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'

0.4.2 Libraries to use

[9]:
```
library(readr)
library(dplyr)
library(sf)
library(tidyr)
#library(stringr)
```

0.4.3 Paths

[10]:
```
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

[11]:
```
# file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[12]:
```
iFile <- "CU_34_08_05_servicios_completo.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_34_08_05_servicios_completo.csv
```

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

[13]:
```
data <- read_csv(file_data)
```

```
Rows: 274792 Columns: 34
  Column specification
```

```
Delimiter: ","
chr   (5): Servicio, CMUN, CDIS, CSEC, NSEC
dbl  (27): Futbol, nservicios, capacidad, tmed, prec, velmedia,
presMax, cca…
lgl   (1): is_train
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.
```

Visualizo los datos.

Estructura de los datos:

[14]:
```
data |> glimpse()
```

```
Rows: 274,792
Columns: 34
$ Fecha            <date> 2022-01-12, 2022-01-31, 2022-01-28,
```

```
2022-01-06, 2022…
$ Servicio         <chr> "Delivery", "Taxi", "Taxi",
"Delivery", "Delivery", "…
$ CMUN             <chr> "079", "079", "903", "079", "007",
"022", "079", "079…
$ CDIS             <chr> "14", "01", "01", "04", "04", "01",
"16", "01", "16",…
$ CSEC             <chr> "050", "048", "006", "080", "012",
"004", "041", "033…
$ Futbol           <dbl> 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0,…
$ nservicios       <dbl> 58, 5, 0, 14, 60, 50, 13, 4, 3, 9,
68, 12, 0, 1, 14, …
$ capacidad        <dbl> 80, 69, 56, 80, 70, 56, 80, 69, 69,
69, 80, 80, 69, 6…
$ tmed             <dbl> 7.366319, 8.823406, 7.854915,
4.226603, 4.982656, 7.2…
$ prec             <dbl> -0.009468616, 0.000000000,
0.000000000, 0.010181896, …
$ velmedia         <dbl> 1.5999961, 1.5114967, 2.2536168,
1.0279945, 1.0387037…
$ presMax          <dbl> 954.7939, 948.9795, 940.2553,
945.1884, 948.9570, 943…
$ ccaa             <dbl> 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 1…
$ CPRO             <dbl> 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 2…
$ t1_1             <dbl> 1094, 1251, 2232, 746, 1080, 2256,
692, 1270, 2229, 8…
$ t2_1             <dbl> 0.5585, 0.4508, 0.5273, 0.5509,
0.5352, 0.4965, 0.536…
$ t2_2             <dbl> 0.4415, 0.5492, 0.4727, 0.4491,
0.4648, 0.5035, 0.463…
$ t3_1             <dbl> 45.4360, 41.6091, 44.2016, 47.1729,
48.5361, 43.2877,…
$ t4_1             <dbl> 0.1298, 0.0823, 0.1263, 0.1233,
0.1278, 0.1658, 0.114…
$ t4_2             <dbl> 0.6408, 0.7970, 0.6644, 0.6166,
0.5574, 0.5988, 0.637…
$ t4_3             <dbl> 0.2294, 0.1207, 0.2092, 0.2601,
0.3148, 0.2354, 0.248…
$ t5_1             <dbl> 0.1517, 0.2974, 0.0896, 0.0389,
0.1870, 0.1250, 0.189…
$ t6_1             <dbl> 0.2852, 0.4037, 0.1523, 0.0751,
0.2639, 0.2101, 0.329…
$ t7_1             <dbl> 0.0378, 0.0392, 0.0718, 0.1116,
0.0372, 0.0813, 0.029…
$ t8_1             <dbl> 0.0263, 0.0305, 0.0615, 0.1070,
```

```
0.0191, 0.0765, 0.021…
$ t9_1             <dbl> 0.2700, 0.5017, 0.5292, 0.6590,
0.2081, 0.6190, 0.323…
$ t10_1            <dbl> 0.1764, 0.1611, 0.0822, 0.0742,
0.1848, 0.0489, 0.137…
$ t11_1            <dbl> 0.4464, 0.5218, 0.4979, 0.4771,
0.3981, 0.4543, 0.489…
$ t12_1            <dbl> 0.5420, 0.6220, 0.5426, 0.5153,
0.4883, 0.4777, 0.567…
$ NSEC             <chr> "Madrid - 14.050", "Madrid -
01.048", "Tres Cantos - …
$ area             <dbl> 38753.96, 15289.89, 124539.78,
89206.78, 24473.30, 34…
$ elevation        <dbl> 658, 635, 719, 710, 693, 710, 702,
635, 690, 710, 690…
$ densidad_hab_km2 <dbl> 28229.3737, 81818.7738, 17921.9842,
8362.5928, 44129.…
$ is_train         <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE, TRUE, TRUE,…
```

Muestra de los primeros datos:

[15]:
```
data |> slice_head(n = 5)
```

A spec_tbl_df: 5 × 34

Fecha <date>	Servicio <chr>	CMUN <chr>	CDIS <chr>	CSEC <chr>	Futbol <dbl>	nservicios <dbl>	capacidad <dbl>	tme <d
2022-01-12	Delivery	079	14	050	1	58	80	7.3
2022-01-31	Taxi	079	01	048	0	5	69	8.8
2022-01-28	Taxi	903	01	006	0	0	56	7.8
2022-01-06	Delivery	079	04	080	0	14	80	4.2
2022-01-21	Delivery	007	04	012	1	60	70	4.9

0.6 Text data analysis

Select columns

[16]:
```
# Select column
```

Operation

[17]:
```
# Analizar datos de texto y verificar su corrección
# e.g. faltas ortografía, etc
```

[18]:
```
# pasar a mayúsculas todas las columnas de texto
```

0.7 Delete Columns Needless/Irrelevant/Private

Select columns

[19]:
```
# Select columns
```

Operation

[20]:
```
# Eliminamos columnas que consideramos irrelevantes o innecesarias
```

[21]:
```
# Eliminamos columnas que consideramos irrelevantes o innecesarias
```

0.8 Inconsistent Data

Select columns and value

[22]:
```
# Select column and value
# e.g. age > 100
```

Operation

[23]:
```
# Inconsistent data is unique to each data set and
# must be searched manually
```

0.9 Expected values

[24]:
```
# Check for expected value
```

0.10 Zeros

[25]:
```
# Check for zeroes in data
zero_counts <- colSums(data == 0, na.rm = TRUE)

# Print variables with zero counts
for (variable in names(zero_counts[zero_counts > 0])) {
  num_zeroes <- zero_counts[variable]
  print(paste("Variable:", variable, "- Number of zeroes:", num_zeroes))
}
```

```
[1] "Variable: Futbol - Number of zeroes: 222963"
[1] "Variable: nservicios - Number of zeroes: 31578"
[1] "Variable: prec - Number of zeroes: 195008"
[1] "Variable: t5_1 - Number of zeroes: 62"
[1] "Variable: is_train - Number of zeroes: 54959"
```

0.11 Single Value

[26]:
```
# Obtener el número de valores diferentes en cada columna
unique_counts <- summarise(data, across(everything(), ~length(unique(.))))

# Imprimir los resultados
print(unique_counts)
```

```
# A tibble: 1 × 34
  Fecha Servicio  CMUN  CDIS  CSEC Futbol nservicios capacidad   tmed  prec
  <int>    <int> <int>
<int> <int>  <int>
<int>     <int>  <int>
<int>
1    32        3   179    21   221      3         63        25
137368 39881
#   24 more variables: velmedia <int>, presMax <int>, ccaa <int>, CPRO
<int>,
#   t1_1 <int>, t2_1 <int>, t2_2 <int>, t3_1 <int>, t4_1 <int>, t4_2
<int>,
#   t4_3 <int>, t5_1 <int>, t6_1 <int>, t7_1 <int>, t8_1 <int>, t9_1
<int>,
#   t10_1 <int>, t11_1 <int>, t12_1 <int>, NSEC <int>, area <int>,
#   elevation <int>, densidad_hab_km2 <int>, is_train <int>
```

[27]:
```
# Columns with a single unique value
#
# NOTE: WE MUST EXCLUDE TARGET COLUMN TO DELETE
```

0.12 Very Few Values

Select rate

[28]:
```
# Select rate
rate <- 0.1
```

Operation

[29]:
```
# Show features with over rate rows being the same value
features_with_same_values <- names(data)[apply(data, 2, function(x)␣
 ↪length(unique(x)) <= (1 - rate) * nrow(data))]
print(features_with_same_values)
```

```
 [1] "Fecha"            "Servicio"         "CMUN"             "CDIS"
 [5] "CSEC"             "Futbol"           "nservicios"       "capacidad"
 [9] "tmed"             "prec"             "velmedia"         "presMax"
[13] "ccaa"             "CPRO"             "t1_1"             "t2_1"
[17] "t2_2"             "t3_1"             "t4_1"             "t4_2"
[21] "t4_3"             "t5_1"             "t6_1"             "t7_1"
[25] "t8_1"             "t9_1"             "t10_1"            "t11_1"
[29] "t12_1"            "NSEC"             "area"             "elevation"
[33] "densidad_hab_km2" "is_train"
```

[30]:
```r
# Summarize the number of unique values in each column
# followed by the percentage of unique values for each
# variable as a percentage of the total number of rows
# in the dataset.

# Summarize the number of unique values in each column
unique_counts <- sapply(data, function(x) length(unique(x)))

# Calculate the percentage of unique values for each variable
percentage_unique <- unique_counts / nrow(data) * 100

# Create a data frame with the results
summary_data <- data.frame(Variable = names(unique_counts), Unique_Count =␣
 ↪unique_counts, Percentage_Unique = percentage_unique)

# Print the summary data
print(summary_data)
```

	Variable	Unique_Count	Percentage_Unique
Fecha	Fecha	32	1.164517e-02
Servicio	Servicio	3	1.091735e-03
CMUN	CMUN	179	6.514018e-02
CDIS	CDIS	21	7.642144e-03
CSEC	CSEC	221	8.042447e-02
Futbol	Futbol	3	1.091735e-03
nservicios	nservicios	63	2.292643e-02
capacidad	capacidad	25	9.097790e-03
tmed	tmed	137368	4.998981e+01
prec	prec	39881	1.451316e+01
velmedia	velmedia	137038	4.986972e+01
presMax	presMax	137393	4.999891e+01
ccaa	ccaa	2	7.278232e-04
CPRO	CPRO	2	7.278232e-04
t1_1	t1_1	1872	6.812425e-01
t2_1	t2_1	1022	3.719177e-01
t2_2	t2_2	1022	3.719177e-01
t3_1	t3_1	4346	1.581560e+00
t4_1	t4_1	1634	5.946316e-01
t4_2	t4_2	1963	7.143585e-01
t4_3	t4_3	2412	8.777548e-01
t5_1	t5_1	2232	8.122507e-01
t6_1	t6_1	2570	9.352528e-01
t7_1	t7_1	960	3.493551e-01
t8_1	t8_1	926	3.369822e-01
t9_1	t9_1	3137	1.141591e+00
t10_1	t10_1	1683	6.124632e-01
t11_1	t11_1	2216	8.064281e-01

```
t12_1                          t12_1         2199      8.002416e-01
NSEC                            NSEC         4433      1.613220e+00
area                            area         4433      1.613220e+00
elevation                  elevation          343      1.248217e-01
densidad_hab_km2 densidad_hab_km2         4410      1.604850e+00
is_train                    is_train            2      7.278232e-04
```

Select percent of the number of rows

[31]:
```
# Select percent of the number of rows
percentage_threshold <- 0.00
```

Operation

[32]:
```
# Summarize columns that have unique values that are less than
# "porcentage_threshold" percent of the number of rows.
columns_to_delete <- names(data)[sapply(data, function(x) length(unique(x)) <␣
 ↪percentage_threshold * nrow(data))]
columns_to_delete <- setdiff(columns_to_delete, "target_column")  # Excluir la␣
 ↪columna objetivo
columns_to_delete
```

Select percent of the number of rows

[33]:
```
# Select percent of the number of rows
```

Operation

[34]:
```
# Delete columns that have unique values that are less than
# "porcentage_threshold" percent of the number of rows.
#
# NOTE: WE MUST EXCLUDE TARGET COLUMN TO DELETE
```

[ ]:
```

```

0.13 Low Variance

A) Calculating variances

[35]:
```
# Calculate variance for all numeric variables
variance <- summarise(data, across(where(is.numeric), var))

# Print the variance
print(variance)
```

```
# A tibble: 1 × 27
  Futbol nservicios capacidad  tmed  prec velmedia presMax  ccaa  CPRO  t1_1
   <dbl>      <dbl>
<dbl> <dbl> <dbl>
```

```
<dbl>   <dbl> <dbl>
<dbl> <dbl>
1     NA          NA         NA
NA    NA        NA       NA
NA    NA    NA
#   17 more variables: t2_1 <dbl>, t2_2 <dbl>, t3_1 <dbl>, t4_1 <dbl>,
#    t4_2 <dbl>, t4_3 <dbl>, t5_1 <dbl>, t6_1 <dbl>, t7_1 <dbl>, t8_1

<dbl>,
#    t9_1 <dbl>, t10_1 <dbl>, t11_1 <dbl>, t12_1 <dbl>, area <dbl>,
#    elevation <dbl>, densidad_hab_km2 <dbl>
```

B) Automatic calculation and representation of variances

Define thresholds to check

[36]:
```
# define thresholds to check
threshold <- 0.05 # Example thresholds, adjust as needed
```

Operation

[37]:
```
data_without_date <- data[, !(names(data) %in% "fecha")]
variances <- apply(data_without_date, 2, function(x) var(x, na.rm = TRUE))
```

```
Warning message in var(x, na.rm = TRUE):
"NAs introduced by coercion"
Warning message in var(x, na.rm = TRUE):
"NAs introduced by coercion"
Warning message in var(x, na.rm = TRUE):
"NAs introduced by coercion"
Warning message in var(x, na.rm = TRUE):
"NAs introduced by coercion"
```

[38]:
```
variances
```

Fecha <NA> **Servicio** <NA> **CMUN** 5527.68615831896 **CDIS** 34.4733384036426 **CSEC** 2380.64829693307 **Futbol** 0.153023202490354 **nservicios** 416.202956570919 **capacidad** 65.9877989856807 **tmed** 5.64636605015807 **prec** 3.46402639272103 **velmedia** 1.09544886237419 **presMax** 297.127447284222 **ccaa** 0 **CPRO** 0 **t1_1** 389141.446431067 **t2_1** 0.000605442445052498 **t2_2** 0.000605441383215074 **t3_1** 21.2475029665694 **t4_1** 0.0026326531517449 **t4_2** 0.00365292597935491 **t4_3** 0.0069921106611222 **t5_1** 0.00651523579409711 **t6_1** 0.0117748388058043 **t7_1** 0.000576004620424424 **t8_1** 0.000569082131620167 **t9_1** 0.0328849947400712 **t10_1** 0.00260051253358624 **t11_1** 0.00698462518143089 **t12_1** 0.00664636278643087 **NSEC** <NA> **area** 66855909771843.4 **elevation** 8251.85591673514 **densidad_hab_km2** 402182029.5966 **is_train** <NA>

C) Delete variables with low variance

Select column

[39]:
```
# Select column
low_variance_columns <- names(variances)[variances < threshold]
```

[40]:
```
low_variance_columns
```

1. NA 2. NA 3. 'ccaa' 4. 'CPRO' 5. 't2_1' 6. 't2_2' 7. 't4_1' 8. 't4_2' 9. 't4_3' 10. 't5_1' 11. 't6_1' 12. 't7_1' 13. 't8_1' 14. 't9_1' 15. 't10_1' 16. 't11_1' 17. 't12_1' 18. NA 19. NA

Operation

[41]:
```
# drop columns with low variance
data <- data[, !(names(data) %in% low_variance_columns)]
```

[42]:
```
# drop columns
```

0.14 Duplicates

Entendido como ERROR –> Eliminar duplicados

0.14.1 Identificación de Duplicates

[43]:
```
# Buscamos filas duplicadas
duplicated_rows <- data[duplicated(data),]
```

0.14.2 Eliminación de primera fila duplicada

[44]:
```
# Eliminamos filas duplicadas
# Deja la primera y elimina el resto duplicadas
data_no_duplicates <- data[!duplicated(data),]
```

0.14.3 Eliminación de todas las filas duplicadas

[45]:
```
# Eliminamos filas duplicadas
# Elimina todas las filas que están duplicadas (no deja una)

# data_no_duplicates_all <- data %>% distinct()
```

0.14.4 Eliminación de filas duplicadas en columnas concretas

Select columns

[46]:
```
# Select columns
```

Operation

[47]:
```
# Eliminamos filas duplicadas
# Elimina las filas que están duplicadas en columnas seleccionadas
```

0.15 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

[48]:
```
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

0.15.1 Proceso 09.1

[49]:
```
caso <- "CU_34"
proceso <- '_091'
tarea <- "_05"
archivo <- ""
proper <- "_servicios_completo"
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

[50]:
```
# 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/.

[51]:
```
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_34_091_05_servicios_completo.csv'

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

[52]:
```
path_in <- paste0(iPath, file_save)
file.copy(path_out, path_in, overwrite = TRUE)
```

TRUE

0.16 REPORT

A continuación se realizará un informe de las acciones realizadas

0.17 Main Actions Carried Out

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

0.18 Main Conclusions

- Los datos están limpios para el despliegue

0.19 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

[ ]:
```

```