09.2.- Data Cleansing-Missing_04_19_vacunacion_completo_v_01

June 8, 2023

#

CU04_Optimización de vacunas

Citizenlab Data Science Methodology > II - Data Processing Domain *** > # 09.2.- Data Cleansing - Missing

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_04_09.1_20_vacunacion_gripe_train_and_test.csv
- Output File: CU_04_09.2_20_vacunacion_gripe_train_and_test_clean.csv

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.

[93]:
```
Sys.setlocale(category = "LC_ALL", locale = "es_ES.UTF-8")
```

```
Warning message in Sys.setlocale(category = "LC_ALL", locale = "es_ES.UTF-8"):
"OS reports request to set locale to "es_ES.UTF-8" cannot be honored"
```

”

0.4.2 Libraries to use

[114]:
```
library(readr)
library(dplyr)
library(tidyr)
library(stringr)
```

0.4.3 Paths

[95]:
```
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

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

OPCION B: Especificar el nombre de archivo

[96]:
```
iFile <- "CU_04_09.1_20_vacunacion_gripe_train_and_test.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_04_09.1_20_vacunacion_gripe_train_and_test.csv
```

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

[97]:
```
data <- read_csv(file_data)
```

```
Rows: 21736 Columns: 48
  Column specification



Delimiter: ","
chr  (3): GEOCODIGO, DESBDT, nombre_zona
dbl (44): ano, semana, n_vacunas, n_citas, tmed, prec, velmedia,
presMax, be…
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:

[98]:
```
data |> glimpse()
```

```
Rows: 21,736
Columns: 48
$ GEOCODIGO         <chr> "097", "128", "155", "085", "049",
"254", "264", "27…
$ DESBDT            <chr> "GALAPAGAR", "LA RIBOTA",
"MAJADAHONDA", "ENSANCHE V…
```

```
$ ano               <dbl> 2022, 2021, 2022, 2021, 2022, 2022,
2022, 2023, 2022…
$ semana            <dbl> 33, 47, 39, 46, 24, 5, 38, 1, 26,
2, 47, 18, 23, 5, …
$ n_vacunas         <dbl> 0, 451, 0, 813, 0, 250, 0, 144, 0,
282, 166, 0, 0, 1…
$ n_citas           <dbl> 0, 437, 0, 789, 0, 235, 0, 137, 0,
271, 159, 0, 0, 1…
$ tmed              <dbl> 21.768536, 6.039860, 15.436997,
9.887983, 21.108264,…
$ prec              <dbl> 0.0550769418, 1.2404689012,
0.6913641020, 0.07183897…
$ velmedia          <dbl> 2.4482484, 2.7974515, 2.7535661,
2.5478336, 3.956291…
$ presMax           <dbl> 901.1438, 936.6692, 926.6612,
952.3018, 833.8937, 89…
$ benzene           <dbl> 0.1795784, 0.3697754, 0.2254214,
0.4194085, 0.195865…
$ co                <dbl> 0.4692918, 0.3468722, 0.4797698,
0.2673996, 0.331213…
$ no                <dbl> 2.005147, 9.513899, 6.130449,
10.993518, 2.451963, 7…
$ no2               <dbl> 10.213564, 24.689603, 22.593902,
36.187953, 10.93601…
$ nox               <dbl> 13.02255, 38.42422, 31.55546,
53.19129, 13.60685, 25…
$ o3                <dbl> 88.27507, 36.57543, 58.67398,
32.54918, 77.88477, 55…
$ pm10              <dbl> 13.887308, 9.361394, 10.401526,
12.783278, 44.451891…
$ pm2.5             <dbl> 8.707578, 6.051115, 5.266344,
6.459633, 17.136398, 1…
$ so2               <dbl> 2.086115, 1.552412, 2.758390,
2.444614, 2.854909, 3.…
$ campana           <dbl> 2021.533, 2021.000, 2022.000,
2021.000, 2021.533, 20…
$ scampana          <dbl> 11.62222, 12.00000, 4.00000,
11.00000, 11.62222, 22.…
$ capacidad_zona    <dbl> 11051, 8524, 12733, 15717, 3792,
6640, 10796, 3364, …
$ prop_riesgo       <dbl> 0.14603798, 0.16062611, 0.21143809,
0.06622598, 0.20…
$ tasa_riesgo       <dbl> 0.003617039, 0.009632178,
0.005353189, 0.012969731, …
$ tasa_mayores      <dbl> 0.018360890, 0.034418204,
0.018018046, 0.026783402, …
$ poblacion_mayores <dbl> 0.13306650, 0.14633197, 0.19219091,
0.06053132, 0.18…
```

```
$ nombre_zona       <chr> "GALAPAGAR", "LA RIBOTA",
"MAJADAHONDA", "ENSANCHE V…
$ nsec              <dbl> 17, 19, 34, 28, 6, 12, 22, 11, 20,
21, 10, 12, 15, 1…
$ t3_1              <dbl> 40.03807, 39.60720, 42.19556,
34.34724, 43.62860, 41…
$ t1_1              <dbl> 44067, 34068, 51144, 62530, 15146,
26552, 43267, 134…
$ t2_1              <dbl> 0.5121733, 0.5109523, 0.5298013,
0.5077573, 0.501588…
$ t2_2              <dbl> 0.4878267, 0.4890477, 0.4701987,
0.4922427, 0.498411…
$ t4_1              <dbl> 0.17622140, 0.19623219, 0.16029496,
0.23756034, 0.14…
$ t4_2              <dbl> 0.6906908, 0.6574383, 0.6475255,
0.7018912, 0.676094…
$ t4_3              <dbl> 0.13306650, 0.14633197, 0.19219091,
0.06053132, 0.18…
$ t5_1              <dbl> 0.15387677, 0.07211496, 0.12445661,
0.12744893, 0.12…
$ t6_1              <dbl> 0.22398769, 0.11679614, 0.21183967,
0.19323644, 0.15…
$ t7_1              <dbl> 0.07342751, 0.05250060, 0.07595339,
0.04601377, 0.05…
$ t8_1              <dbl> 0.05728152, 0.03935768, 0.06703038,
0.03454148, 0.04…
$ t9_1              <dbl> 0.4408272, 0.4406703, 0.5570257,
0.4603761, 0.387025…
$ t10_1             <dbl> 0.12371972, 0.11272335, 0.08802468,
0.13945576, 0.11…
$ t11_1             <dbl> 0.5291455, 0.6094153, 0.5018791,
0.6560315, 0.515400…
$ t12_1             <dbl> 0.6040733, 0.6814646, 0.5505073,
0.7524379, 0.585228…
$ area              <dbl> 96647460.4, 1364369.5, 30837796.0,
48678625.6, 87516…
$ densidad_hab_km   <dbl> 455.95611, 24969.77491, 1658.48428,
1284.54736, 173.…
$ tuits_gripe       <dbl> 34, 280, 126, 206, 46, 144, 98, 24,
70, 508, 280, 12…
$ interes_gripe     <dbl> 11, 64, 42, 64, 21, 20, 32, 64, 20,
69, 64, 36, 26, …
$ is_train          <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE, TRUE, TRUE…
```

Muestra de los primeros datos:

[99]:
```
data |> slice_head(n = 5)
```

	GEOCODIGO <chr>	DESBDT <chr>	ano <dbl>	semana <dbl>	n_vacunas <dbl>	n_citas <dbl>	t
A spec_tbl_df: 5 × 48	097	GALAPAGAR	2022	33	0	0	2
	128	LA RIBOTA	2021	47	451	437	6
	155	MAJADAHONDA	2022	39	0	0	1
	085	ENSANCHE VALLECAS	2021	46	813	789	9
	049	CERCEDILLA	2022	24	0	0	2

0.6 Missing Values

0.6.1 Missing Values Identification

Missing Values Per Sample

[100]:
```
# Create a dataframe with ID and number of missing values per sample
missing_per_sample_df <- data.frame(ID = 1:nrow(data), NAs = rowSums(is.
  ↪na(data)))

# Filter rows with more than one missing value
filtered_missing_per_sample_df <-␣
  ↪missing_per_sample_df[missing_per_sample_df$NAs > 1, ]

# Print the filtered dataframe
print(filtered_missing_per_sample_df)
```

```
         ID NAs
85       85   2
105     105   2
175     175   2
177     177   2
190     190   2
191     191   2
193     193   2
239     239   2
274     274   2
314     314   2
321     321   2
336     336   2
411     411   2
421     421   2
449     449   2
488     488   2
609     609   2
616     616   2
660     660   2
696     696   2
754     754   2
759     759   2
766     766   2
```

778	778	2
787	787	2
799	799	2
843	843	2
858	858	2
878	878	2
888	888	2
893	893	2
935	935	2
1029	1029	2
1079	1079	2
1094	1094	2
1155	1155	2
1207	1207	2
1259	1259	2
1311	1311	2
1323	1323	2
1379	1379	2
1413	1413	2
1462	1462	2
1492	1492	2
1564	1564	2
1583	1583	2
1586	1586	2
1625	1625	2
1670	1670	2
1684	1684	2
1698	1698	2
1710	1710	2
1728	1728	2
1764	1764	2
1798	1798	2
1823	1823	2
1860	1860	2
1917	1917	2
1925	1925	2
2002	2002	2
2019	2019	2
2048	2048	2
2084	2084	2
2124	2124	2
2133	2133	2
2158	2158	2
2198	2198	2
2262	2262	2
2271	2271	2
2329	2329	2
2334	2334	2

2379	2379	2
2387	2387	2
2398	2398	2
2425	2425	2
2483	2483	2
2487	2487	2
2501	2501	2
2518	2518	2
2583	2583	2
2594	2594	2
2595	2595	2
2604	2604	2
2607	2607	2
2649	2649	2
2691	2691	2
2707	2707	2
2764	2764	2
2769	2769	2
2795	2795	2
2857	2857	2
2873	2873	2
3004	3004	2
3063	3063	2
3153	3153	2
3176	3176	2
3189	3189	2
3190	3190	2
3200	3200	2
3264	3264	2
3375	3375	2
3397	3397	2
3510	3510	2
3546	3546	2
3601	3601	2
3608	3608	2
3618	3618	2
3716	3716	2
3788	3788	2
3988	3988	2
4111	4111	2
4117	4117	2
4154	4154	2
4163	4163	2
4171	4171	2
4273	4273	2
4285	4285	2
4311	4311	2
4386	4386	2

4390	4390	2
4392	4392	2
4424	4424	2
4465	4465	2
4514	4514	2
4526	4526	2
4528	4528	2
4530	4530	2
4537	4537	2
4610	4610	2
4657	4657	2
4685	4685	2
4708	4708	2
4827	4827	2
4943	4943	2
4962	4962	2
4987	4987	2
5012	5012	2
5039	5039	2
5046	5046	2
5061	5061	2
5069	5069	2
5085	5085	2
5097	5097	2
5178	5178	2
5374	5374	2
5383	5383	2
5405	5405	2
5457	5457	2
5511	5511	2
5690	5690	2
5748	5748	2
5897	5897	2
5952	5952	2
5989	5989	2
6063	6063	2
6155	6155	2
6218	6218	2
6266	6266	2
6300	6300	2
6367	6367	2
6414	6414	2
6450	6450	2
6480	6480	2
6491	6491	2
6493	6493	2
6512	6512	2
6577	6577	2

6578	6578	2
6580	6580	2
6611	6611	2
6672	6672	2
6673	6673	2
6682	6682	2
6714	6714	2
6739	6739	2
6760	6760	2
6805	6805	2
6822	6822	2
6918	6918	2
6969	6969	2
6973	6973	2
6999	6999	2
7077	7077	2
7086	7086	2
7136	7136	2
7174	7174	2
7185	7185	2
7193	7193	2
7202	7202	2
7227	7227	2
7272	7272	2
7275	7275	2
7331	7331	2
7333	7333	2
7342	7342	2
7383	7383	2
7411	7411	2
7435	7435	2
7483	7483	2
7537	7537	2
7574	7574	2
7605	7605	2
7624	7624	2
7628	7628	2
7632	7632	2
7699	7699	2
7708	7708	2
7750	7750	2
7789	7789	2
7811	7811	2
7819	7819	2
7862	7862	2
7951	7951	2
8061	8061	2
8106	8106	2

8117	8117	2
8156	8156	2
8164	8164	2
8199	8199	2
8211	8211	2
8259	8259	2
8312	8312	2
8348	8348	2
8423	8423	2
8486	8486	2
8619	8619	2
8663	8663	2
8718	8718	2
8748	8748	2
8815	8815	2
8969	8969	2
8996	8996	2
9041	9041	2
9049	9049	2
9080	9080	2
9098	9098	2
9120	9120	2
9213	9213	2
9229	9229	2
9287	9287	2
9302	9302	2
9371	9371	2
9376	9376	2
9379	9379	2
9460	9460	2
9543	9543	2
9565	9565	2
9570	9570	2
9635	9635	2
9650	9650	2
9710	9710	2
9756	9756	2
9834	9834	2
9850	9850	2
9872	9872	2
9888	9888	2
9910	9910	2
9993	9993	2
10000	10000	2
10025	10025	2
10049	10049	2
10141	10141	2
10209	10209	2

```
10254 10254   2
10276 10276   2
10288 10288   2
10381 10381   2
10403 10403   2
10422 10422   2
10476 10476   2
10513 10513   2
10579 10579   2
10627 10627   2
10676 10676   2
10740 10740   2
10749 10749   2
10758 10758   2
10800 10800   2
10803 10803   2
10819 10819   2
10975 10975   2
10995 10995   2
11087 11087   2
11104 11104   2
11167 11167   2
11228 11228   2
11237 11237   2
11332 11332   2
11354 11354   2
11372 11372   2
11429 11429   2
11493 11493   2
11529 11529   2
11544 11544   2
11568 11568   2
11582 11582   2
11591 11591   2
11711 11711   2
11750 11750   2
11776 11776   2
11806 11806   2
11851 11851   2
11881 11881   2
11923 11923   2
11955 11955   2
12025 12025   2
12107 12107   2
12124 12124   2
12140 12140   2
12146 12146   2
12270 12270   2
```

```
12271 12271   2
12303 12303   2
12326 12326   2
12409 12409   2
12447 12447   2
12478 12478   2
12537 12537   2
12553 12553   2
12564 12564   2
12582 12582   2
12600 12600   2
12711 12711   2
12787 12787   2
12823 12823   2
12863 12863   2
12962 12962   2
12964 12964   2
13024 13024   2
13042 13042   2
13045 13045   2
13069 13069   2
13119 13119   2
13153 13153   2
13164 13164   2
13191 13191   2
13210 13210   2
13266 13266   2
13381 13381   2
13420 13420   2
13434 13434   2
13498 13498   2
13520 13520   2
13576 13576   2
13607 13607   2
13627 13627   2
13658 13658   2
13694 13694   2
13787 13787   2
13814 13814   2
13826 13826   2
13895 13895   2
13932 13932   2
13940 13940   2
13987 13987   2
14015 14015   2
14047 14047   2
14065 14065   2
14105 14105   2
```

```
14136 14136   2
14186 14186   2
14268 14268   2
14332 14332   2
14401 14401   2
14443 14443   2
14495 14495   2
14500 14500   2
14515 14515   2
14527 14527   2
14580 14580   2
14592 14592   2
14620 14620   2
14641 14641   2
14700 14700   2
14759 14759   2
14819 14819   2
14910 14910   2
14942 14942   2
14953 14953   2
14977 14977   2
14992 14992   2
14993 14993   2
15056 15056   2
15063 15063   2
15067 15067   2
15101 15101   2
15110 15110   2
15171 15171   2
15223 15223   2
15339 15339   2
15353 15353   2
15379 15379   2
15381 15381   2
15408 15408   2
15458 15458   2
15472 15472   2
15501 15501   2
15679 15679   2
15717 15717   2
15771 15771   2
15774 15774   2
15778 15778   2
15828 15828   2
15832 15832   2
15840 15840   2
15841 15841   2
15858 15858   2
```

```
15938 15938   2
15958 15958   2
16011 16011   2
16041 16041   2
16062 16062   2
16105 16105   2
16172 16172   2
16187 16187   2
16200 16200   2
16282 16282   2
16293 16293   2
16321 16321   2
16351 16351   2
16411 16411   2
16453 16453   2
16465 16465   2
16476 16476   2
16524 16524   2
16598 16598   2
16603 16603   2
16623 16623   2
16662 16662   2
16667 16667   2
16671 16671   2
16811 16811   2
16970 16970   2
17041 17041   2
17052 17052   2
17059 17059   2
17141 17141   2
17143 17143   2
17214 17214   2
17226 17226   2
17236 17236   2
17241 17241   2
17286 17286   2
17297 17297   2
17330 17330   2
17385 17385   2
21553 21553   2
21554 21554   2
21555 21555   2
21556 21556   2
21557 21557   2
21558 21558   2
21559 21559   2
21560 21560   2
21561 21561   2
```

```
21562 21562   2
21563 21563   2
21564 21564   2
21565 21565   2
21566 21566   2
21567 21567   2
21568 21568   2
21569 21569   2
21570 21570   2
21571 21571   2
21572 21572   2
21573 21573   2
21574 21574   2
21575 21575   2
21576 21576   2
21577 21577   2
21578 21578   2
21579 21579   2
21580 21580   2
21581 21581   2
21582 21582   2
21583 21583   2
21584 21584   2
21585 21585   2
21586 21586   2
21587 21587   2
21588 21588   2
21589 21589   2
21590 21590   2
21592 21592   2
21593 21593   2
21594 21594   2
21595 21595   2
21596 21596   2
21597 21597   2
21598 21598   2
21599 21599   2
21600 21600   2
21601 21601   2
21602 21602   2
21603 21603   2
21604 21604   2
21605 21605   2
21606 21606   2
21607 21607   2
21608 21608   2
21609 21609   2
21610 21610   2
```

```
21611 21611   2
21612 21612   2
21613 21613   2
21614 21614   2
21615 21615   2
21616 21616   2
21617 21617   2
21618 21618   2
21619 21619   2
21685 21685   2
21686 21686   2
21687 21687   2
21688 21688   2
21689 21689   2
21690 21690   2
21691 21691   2
21692 21692   2
21693 21693   2
21694 21694   2
21695 21695   2
21696 21696   2
21697 21697   2
21698 21698   2
21699 21699   2
21700 21700   2
21701 21701   2
21702 21702   2
21703 21703   2
21704 21704   2
21705 21705   2
21706 21706   2
21707 21707   2
21708 21708   2
21709 21709   2
21710 21710   2
21711 21711   2
21712 21712   2
21713 21713   2
21714 21714   2
21715 21715   2
21716 21716   2
21717 21717   2
21718 21718   2
21719 21719   2
21720 21720   2
21721 21721   2
21722 21722   2
21723 21723   2
```

```
21724 21724   2
21725 21725   2
21726 21726   2
21727 21727   2
21728 21728   2
21729 21729   2
21730 21730   2
21731 21731   2
21732 21732   2
21733 21733   2
21734 21734   2
21735 21735   2
21736 21736   2
```

Missing Values Per Feature

[101]:
```r
# Calculate the number of missing values per feature
missing_per_feature <- colSums(is.na(data))

# Print the number of missing values per feature
print(missing_per_feature)
```

```
       GEOCODIGO            DESBDT               ano            semana 
               0               868                 0                 0 
       n_vacunas          n_citas              tmed              prec 
               0                 0                 0                 0 
        velmedia          presMax           benzene                co 
               0                 0                 0                 0 
              no              no2               nox                o3 
               0                 0                 0                 0 
            pm10             pm2.5               so2          campana 
               0                 0                 0                 0 
        scampana    capacidad_zona      prop_riesgo      tasa_riesgo 
               0                 0                 0                 0 
    tasa_mayores poblacion_mayores       nombre_zona              nsec 
               0                 0               564                 0 
            t3_1              t1_1              t2_1              t2_2 
               0                 0                 0                 0 
            t4_1              t4_2              t4_3              t5_1 
               0                 0                 0                 0 
            t6_1              t7_1              t8_1              t9_1 
               0                 0                 0                 0 
           t10_1             t11_1             t12_1              area 
               0                 0                 0                 0 
 densidad_hab_km       tuits_gripe     interes_gripe          is_train 
               0                 0                 0                 0 
```

Zero Missing Values

[104]:
```r
# Detecting columns with minimum value of zero (0).

# Calculate the frequency of zeros in each column
zero_counts <- sapply(data, function(x) sum(x == 0, na.rm = TRUE))

# Calculate the proportion of zeros in each column
zero_proportions <- zero_counts / nrow(data)

# Set a threshold for the proportion of zeros
zero_threshold <- 0.9  # Adjust as needed

# Identify columns with a high proportion of zeros
columns_with_high_zeros <- names(zero_proportions[zero_proportions >=␣
 ↪zero_threshold])

# Print the columns with a high proportion of zeros
print(columns_with_high_zeros)
```

```
character(0)
```

Select column to replace

[ ]:
```r
# Select column to replace
```

Operation

[105]:
```r
# Replace zero missing values by nan
# Replace columns with a high proportion of zeros with NaN
data[, columns_with_high_zeros] <- NA
```

[ ]:
```r

```

Other Missing Values Select column to replace

[ ]:
```r
# Select column to replace and missing value
```

Operation

[ ]:
```r
# Replace other missing values by nan
```

Null/NaN Missing Values

[ ]:
```r
# Intuitivamente: miramos nº datos en todas las columnas
# los null no los cuenta --> debe hacer el mismo nº por columna
```

[ ]:
```r
# Podemos mirar directamente info donde viene
```

[ ]:
```r
# Contamos los nulos de forma explícita
```

[ ]:
```
# summarize the number of rows with missing values for each column
```

[ ]:
```

```

0.6.2 Delete Missing Values

Deleting Rows with Missing Values in Target Column

[107]:
```
# data <- data[!is.na(data$Target), ]
```

Deleting Rows with Missing Values Only in case of high data size

[108]:
```
# Eliminamos las filas con valores nulos
# data <- data[complete.cases(data), ]
```

Deleting Features with some Missing Values Only with many features and for non-relevant features

[109]:
```
# Selecciono las columnas con algún valor missing:
# Identify columns with any missing values
cols_with_na <- sapply(data, function(x) any(is.na(x)))

# Keep only those columns that do not have missing values
data <- data[, !cols_with_na]
```

0.6.3 Basic Imputation

Imputation by Previous Row Value

[115]:
```
# Sustituimos valores null por otro valor: VALOR FILA ANTERIOR
data <- data %>% fill(everything(), .direction = "down")
```

Imputation by Next Row Value

[116]:
```
# Sustituimos valores null por otro valor: VALOR FILA SIGUIENTE
data <- data %>% fill(everything(), .direction = "up")
```

0.6.4 Statistical Imputation

A popular approach for data imputation is to calculate a statistical value for each column (such as a mean) and replace all missing values for that column with the statistic.

Selection of Imputation Strategy

[ ]:
```
# The mean accuracy of each approach can then be compared.
#
# Specific results may vary given the stochastic nature of
# the learning algorithm, the evaluation procedure, or
# differences in numerical precision. Consider running the
```

```
# example a few times and compare the average performance.
#
```

[ ]:
```
# Plot model performance for comparison
# box and whisker plot is created for each set of results,
# allowing the distribution of results to be compared.
```

Constant Imputation Select constant value

[118]:
```
# Select constant value
constant = 0
```

Operation

[ ]:
```
# Constant imputation
data <- replace_na(data, list(everything() <- constant))
```

Mean Imputation

[ ]:
```
# Identify numeric columns
numeric_cols <- sapply(data, is.numeric)

# Apply mean imputation only on numeric columns
data_imputed <- data
for(column in names(data)[numeric_cols]) {
    data_imputed[is.na(data_imputed[,column]), column] <- mean(data[,column],␣
 ↪na.rm = TRUE)
}
```

Median Imputation

[ ]:
```
# Identify numeric columns
numeric_cols <- sapply(data, is.numeric)

# Apply median imputation only on numeric columns
data_imputed <- data
for(column in names(data)[numeric_cols]) {
    data_imputed[is.na(data_imputed[,column]), column] <- median(data[,column],␣
 ↪na.rm = TRUE)
}
```

Most Frequent Imputation

[ ]:
```
# Create a function to calculate the mode
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
```

```r
# Apply most frequent imputation to each column
data_imputed <- data
for(column in names(data)) {
    data_imputed[is.na(data_imputed[,column]), column] <- getmode(data[,column])
}
```

Interpolation Imputation

[ ]:
```r
# If not already installed, install the zoo package
if(!require(zoo)) install.packages('zoo')

# Load the zoo package
library(zoo)

# Choose an interpolation method
interpolation_method <- "linear"  # "linear" or "spline"

# Apply interpolation imputation
data_imputed <- data
if(interpolation_method == "linear") {
    data_imputed <- na.approx(data_imputed, na.rm = FALSE)
} else if(interpolation_method == "spline") {
    data_imputed <- na.spline(data_imputed, na.rm = FALSE)
} else {
    stop("Invalid interpolation method")
}
```

0.6.5 Prediction Imputation (KNN Imputation)

An approach to missing data imputation is to use a model to predict the missing values.

Evaluating k-hyperparmeter in KNN Imputation Select numbers of neighbors to evaluate

[ ]:
```r
# Numbers of neighbors to evaluate
k_values <- c(3, 5, 7, 9, 11)
```

Operation

[ ]:
```r
# If not already installed, install the VIM package
if(!require(VIM)) install.packages('VIM')

# Load the VIM package
library(VIM)

# Cross-validation
cv_errors <- sapply(k_values, function(k) {
    # Apply KNN imputation
```

```
    data_imputed <- kNN(data, k = k)

    # Calculate and return the mean squared error
    mean((data - data_imputed)^2, na.rm = TRUE)
})

# Print the cross-validation errors
print(cv_errors)
```

Applying KNN Imputation

[ ]:

Select numbers of neighbors to evaluate

[ ]:
```
# Number of neighbors
optimal_k <- k_values[which.min(cv_errors)]
```

Operation

[ ]:
```
data_imputed <- kNN(data, k = optimal_k)
```

0.6.6 Iterative Imputation

Evaluating Different Imputation Order We can experiment with different imputation order strategies, such as descending, right-to-left (Arabic), left-to-right (Roman), and random.

[ ]:
```
# If not already installed, install the mice package
if(!require(mice)) install.packages('mice')

# Load the mice package
library(mice)

# Determine the percentage of missing values in each column
missing_values <- sapply(data, function(x) sum(is.na(x))/length(x))

# Order the variables based on the percentage of missing values
ascending_order <- order(missing_values)
descending_order <- order(missing_values, decreasing = TRUE)
random_order <- sample(length(missing_values))

# Print the imputation orders
print(ascending_order)
print(descending_order)
print(random_order)
```

Applying Iterative Imputation Select strategie

[ ]:
```
# Choose an imputation order strategy
imputation_order <- ascending_order  # or descending_order, or random_order
```

Operation

[ ]:
```
# Perform iterative imputation
mice_output <- mice(data[, imputation_order], m = 5, maxit = 50, method =␣
↪'pmm', seed = 500)

# Get the imputed data
data_imputed <- complete(mice_output, 1)
```

[ ]:
```

```

[ ]:
```
# Generating the new Data dataframe
```

0.7 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

[ ]:
```
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

0.7.1 Proceso 09.2

[123]:
```
caso <- "CU_04"
proceso <- '_09.2'
tarea <- "_20"
archivo <- ""
proper <- "_vacunacion_gripe_train_and_test_clean"
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

[ ]:
```
# 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/.

[125]:
```
file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out <- paste0(oPath, file_save)
write_csv(data, path_out)

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

```
File saved as:
```

'Data/Output/CU_04_09.2_20_vacunacion_gripe_train_and_test_clean.csv'

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

[126]:
```
path_in <- paste0(iPath, file_save)
file.copy(path_out, path_in, overwrite = TRUE)
```

TRUE

0.8 REPORT

A continuación se realizará un informe de las acciones realizadas

0.9 Main Actions Carried Out

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

0.10 Main Conclusions

- Los datos están limpios para el despliegue

0.11 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

[ ]: