16.- Feature Selection_25_01_listas_espera_v_01

June 10, 2023

#

CU25_Modelo de gestión de Lista de Espera Quirúrgica

Citizenlab Data Science Methodology > III - Feature Engineering Domain *** > # 16.- Feature Selection

Feature Selection is the process where you automatically or manually select the most relevant features which contribute most to the correct output of the model.

0.1 Tasks

Perform Selection of Categorical-Input/Categorical-Output
- Encoding-Categorical-Features - Chi-Squared-Feature-Selection - Mutual-Information-Feature-Selection - Evaluate-a-Logistic-Regression-model

Perform Selection of Numerical-Input/Categorical-Output
- ANOVA-F-test-Feature-Selection - Mutual-Information-Feature-Selection - Evaluating-a-Logistic-Regression-model - Tuning-the-Number-of-Selected-Features

Perform Selection of Numerical-Input/Numerical-Output
- Correlation-with-the-outcome-Feature-Selection - Mutual-Information-Feature-Selection - Evaluate-a-Lineal-Regression-model - Tuning-the-Number-of-Selected-Features

Perform Selection of Any-data
- RFE-(Recursive-Feature-Elimination) - Tuning-the-Number-of-Selected-Features - Automatically-Select-the-Number-of-Features

Explore the use of diferent algorithms wrapped by RFE

Explore the use od Hybrid feature selection algorithms

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_25_09.2_01_lista_espera_completo_clean_v_01.csv
- 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.

[26]:
```
Sys.setlocale(category = "LC_ALL", locale = "es_ES.UTF-8")
```

'LC_COLLATE=es_ES.UTF-8;LC_CTYPE=es_ES.UTF-8;LC_MONETARY=es_ES.UTF-8;LC_NUMERIC=C;LC_TIME=es_ES.UTF-8'

0.4 Settings

0.4.1 Libraries to use

[27]:
```
library(readr)
library(dplyr)
library(tidyr)
library(forcats)
library(lubridate)
```

0.4.2 Paths

[28]:
```
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

[29]:
```
# file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[30]:
```
iFile <- "CU_25_09.2_01_lista_espera_completo_clean_v_01.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_25_09.2_01_lista_espera_completo_clean_v_01.csv
```

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

[81]:
```
data <- read.csv(file_data)
```

Estructura de los datos:

[32]:
```
data |> glimpse()
```

```
Rows: 55,216
Columns: 46
$ Hospital          <chr> "HOSPITAL REY JUAN CARLOS",
"HOSPITAL CENTRAL DE LA …
$ Especialidad      <chr> "UROLOGÍA", "ODONTOESTOMATOLOGÍA",
"GINECOLOGÍA", "D…
$ total_pacientes   <dbl> 344, 0, 52, 37, 0, 4, 0, 718, 0,
271, 108, 0, 34, 86…
$ ano               <dbl> 2021, 2020, 2021, 2021, 2021, 2020,
2021, 2020, 2021…
$ semana            <dbl> 30, 36, 49, 23, 3, 5, 50, 7, 35, 1,
42, 10, 21, 33, …
$ CODCNH            <dbl> 281348, 280724, 281292, 281292,
281236, 280724, 2807…
$ id_area           <dbl> 8, 7, 11, 11, 11, 7, 3, 6, 1, 2, 2,
8, 11, 11, 1, 3,…
$ nombre_area       <chr> "SUR-OESTE I", "CENTRO-OESTE", "SUR
II", "SUR II", "…
$ cmunicipio        <dbl> 280920, 280796, 280133, 280133,
281610, 280796, 2800…
$ Municipio         <chr> "MÓSTOLES", "MADRID", "ARANJUEZ",
"ARANJUEZ", "VALDE…
$ CAMAS             <dbl> 382, 475, 98, 98, 182, 475, 507,
613, 269, 1143, 156…
$ Clase             <chr> "HOSPITALES GENERALES", "HOSPITALES
GENERALES", "HOS…
$ Dependencia       <chr> "SERVICIOS E INSTITUTOS DE SALUD DE
LAS COMUNIDADES …
$ TAC               <dbl> 2, 2, 1, 1, 1, 2, 3, 3, 0, 0, 1, 2,
6, 6, 1, 3, 4, 1…
$ RM                <dbl> 3, 2, 1, 1, 2, 2, 2, 3, 0, 0, 0, 2,
5, 5, 1, 2, 4, 1…
$ GAM               <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
2, 2, 0, 0, 2, 0…
```

```
$ HEM               <dbl> 1, 2, 0, 0, 1, 2, 1, 2, 0, 0, 0, 1,
3, 3, 0, 1, 1, 0…
$ ASD               <dbl> 2, 1, 1, 1, 1, 1, 1, 3, 0, 0, 0, 1,
2, 2, 0, 1, 2, 1…
$ ALI               <dbl> 1, 2, 0, 0, 0, 2, 0, 4, 0, 0, 0, 0,
3, 3, 0, 2, 2, 0…
$ SPECT             <dbl> 1, 1, 0, 0, 0, 1, 0, 4, 0, 0, 0, 0,
3, 3, 0, 0, 0, 0…
$ MAMOS             <dbl> 2, 1, 1, 1, 1, 1, 2, 2, 0, 0, 1, 2,
3, 3, 1, 1, 3, 1…
$ DO                <dbl> 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
2, 2, 0, 1, 2, 0…
$ DIAL              <dbl> 20, 24, 13, 13, 17, 24, 28, 31, 0,
0, 0, 28, 43, 43,…
$ X                 <dbl> -3.870412, -3.745529, -3.610795,
-3.610795, -3.69744…
$ Y                 <dbl> 40.33920, 40.38791, 40.05726,
40.05726, 40.19884, 40…
$ t3_1              <dbl> 42.34715, 45.37878, 42.06149,
42.06149, 42.06149, 45…
$ t1_1              <dbl> 532487, 511605, 899702, 899702,
899702, 511605, 3830…
$ t2_1              <dbl> 0.5122493, 0.5296804, 0.5240445,
0.5240445, 0.524044…
$ t2_2              <dbl> 0.4877507, 0.4703198, 0.4759555,
0.4759555, 0.475955…
$ t4_1              <dbl> 0.1659665, 0.1054260, 0.1540793,
0.1540793, 0.154079…
$ t4_2              <dbl> 0.6371549, 0.6742432, 0.6753787,
0.6753787, 0.675378…
$ t4_3              <dbl> 0.1968769, 0.2203341, 0.1705449,
0.1705449, 0.170544…
$ t5_1              <dbl> 0.1137647, 0.1744493, 0.1747059,
0.1747059, 0.174705…
$ t6_1              <dbl> 0.1604646, 0.2629599, 0.2641879,
0.2641879, 0.264187…
$ t7_1              <dbl> 0.05422176, 0.05481008, 0.04898547,
0.04898547, 0.04…
$ t8_1              <dbl> 0.04120012, 0.04653221, 0.03679912,
0.03679912, 0.03…
$ t9_1              <dbl> 0.3348780, 0.4914365, 0.3346063,
0.3346063, 0.334606…
$ t10_1             <dbl> 0.13692541, 0.12170996, 0.15173209,
0.15173209, 0.15…
$ t11_1             <dbl> 0.5072726, 0.4915713, 0.5024130,
0.5024130, 0.502413…
$ t12_1             <dbl> 0.5849309, 0.5597213, 0.5900028,
0.5900028, 0.590002…
```

```
$ capacidad          <dbl> 17, 0, 8, 5, 0, 5, 1, 24, 6, 6, 30,
4, 2, 15, 20, 6,…
$ pacientes          <dbl> 1447, 1211, 1293, 1501, 1240, 1504,
1502, 1533, 1463…
$ consultas          <dbl> 573, 45, 108, 103, 44, 42, 36,
1119, 34, 466, 220, 6…
$ hospitalizaciones <dbl> 12, 0, 2, 2, 0, 1, 0, 4, 0, 12, 3,
0, 2, 4, 1, 2, 15…
$ Target             <dbl> 54.45, 0.00, 37.96, 23.14, 0.00,
6.25, 0.00, 78.20, …
$ is_train           <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE, TRUE, TRUE…
```

Muestra de los primeros datos:

[33]:
```
data |> slice_head(n = 5)
```

A spec_tbl_df: 5 × 46

Hospital <chr>	Especialidad <chr>
HOSPITAL REY JUAN CARLOS	UROLOGÍA
HOSPITAL CENTRAL DE LA DEFENSA GOMEZ ULLA	ODONTOESTOMATOLO
HOSPITAL UNIVERSITARIO DEL TAJO	GINECOLOGÍA
HOSPITAL UNIVERSITARIO DEL TAJO	DERMATOLOGÍA
HOSPITAL UNIVERSITARIO INFANTA ELENA	ODONTOESTOMATOLO

0.6 Selecting Categorical Input / Categorical Output

0.6.1 Encoding Categorical Features

[34]:
```
# Convert all character columns to factors
data <- mutate_if(data, is.character, as.factor)
data <- na.omit(data)

train_set <- subset(data[data$is_train == TRUE, ], select = -is_train)
train_set <- select_if(train_set, is.numeric)
test_set <- subset(data[data$is_train == FALSE, ], select = -is_train)
test_set <- select_if(test_set, is.numeric)
```

0.6.2 Chi-Squared Feature Selection

No aplica ya que el Target no es categórico.

0.6.3 Mutual Information Feature Selection

No aplica ya que el Target no es categórico.

0.6.4 Evaluating a Logistic Regression model

No aplica ya que el Target no es categórico.

Operation

0.7 Selecting Numerical Input / Categorical Output

0.7.1 ANOVA F-test Feature Selection

0.7.2 Mutual Information Feature Selection

0.7.3 Evaluating a Logistic Regression model

Selecting feature to use

[35]:
```
# Select numer of Features to use
```

Operation

0.8 Selecting Numerical Input / Numerical Output

0.8.1 Correlation with the outcome Feature Selection

[77]:
```
# Calculate the correlation between each feature and the outcome variable
correlations <- sapply(select_if(train_set, is.numeric)[,␣
 ↪-which(names(select_if(train_set, is.numeric)) %in% "Target")], function(x)␣
 ↪cor(x, train_set$Target))

# Create a dataframe from the correlations
correlation_df <- data.frame(Feature = names(correlations), Correlation =␣
 ↪correlations)

# Sort the dataframe by the absolute values of the correlations in descending␣
 ↪order
correlation_df <- correlation_df[order(-abs(correlation_df$Correlation)), ]

# Print the correlation dataframe
print(correlation_df)
```

```
                           Feature  Correlation
consultas                consultas  0.342597189
total_pacientes    total_pacientes  0.342584988
CAMAS                        CAMAS  0.313363306
HEM                            HEM  0.299355442
hospitalizaciones hospitalizaciones  0.296903803
TAC                            TAC  0.270636095
ALI                            ALI  0.243984162
ASD                            ASD  0.237658677
capacidad                capacidad  0.234853780
SPECT                        SPECT  0.224919516
DIAL                          DIAL  0.224345426
RM                              RM  0.172976711
X                                X  0.167823523
CODCNH                      CODCNH -0.167594808
```

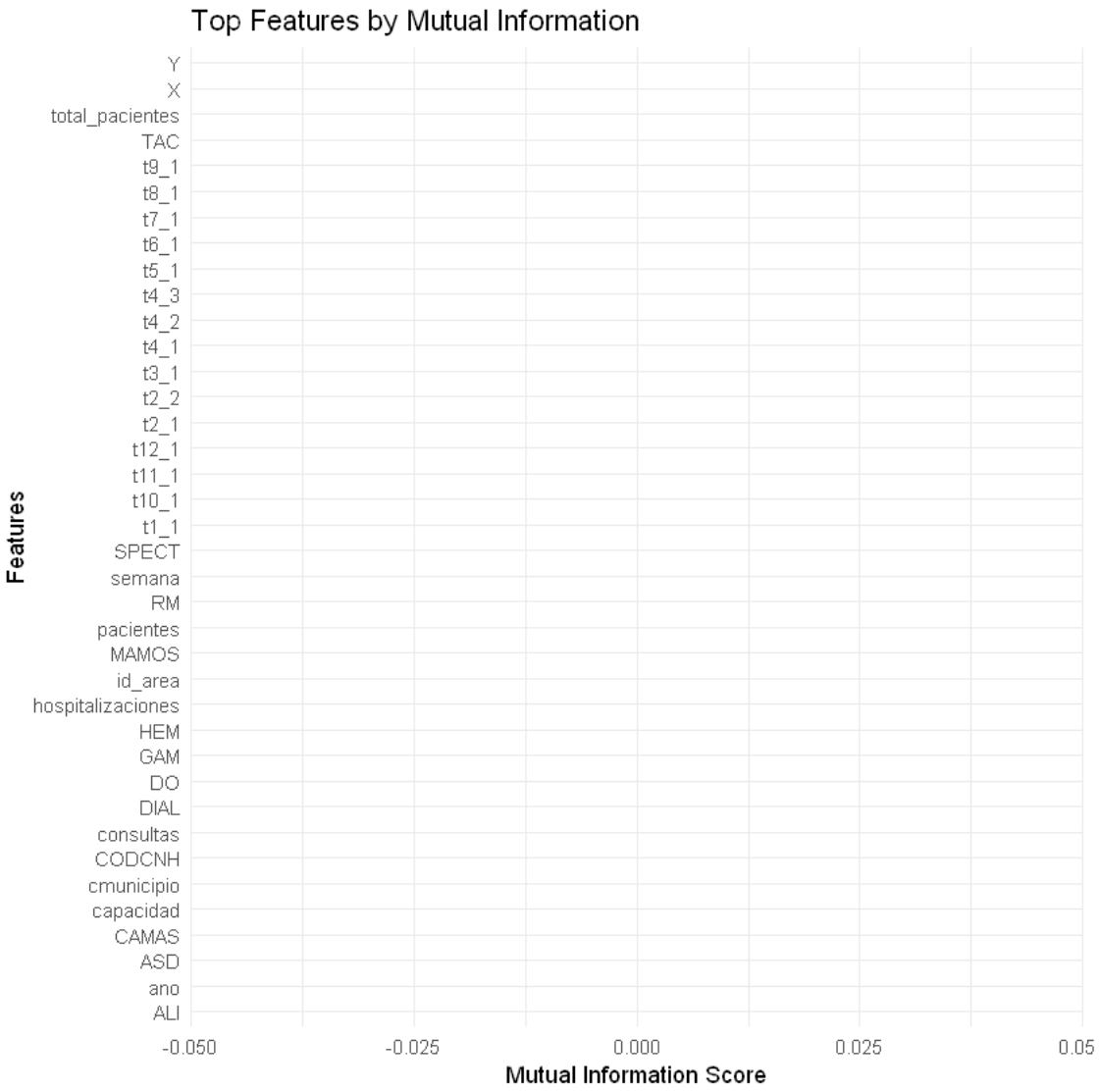
```
DO                              DO  0.164167441
GAM                            GAM  0.162103092
MAMOS                        MAMOS  0.159968110
Y                                Y  0.122519855
t5_1                          t5_1  0.111325391
t6_1                          t6_1  0.082282323
t4_2                          t4_2  0.071829666
t4_1                          t4_1 -0.069954598
t1_1                          t1_1 -0.060904427
t9_1                          t9_1  0.055168040
id_area                    id_area -0.048659120
t8_1                          t8_1  0.048501179
t3_1                          t3_1  0.045284963
ano                            ano  0.042236631
t10_1                        t10_1 -0.040038361
t2_2                          t2_2 -0.034344806
t2_1                          t2_1  0.032881739
t4_3                          t4_3  0.027963286
t7_1                          t7_1  0.027407609
semana                      semana  0.026980146
t11_1                        t11_1  0.019186010
t12_1                        t12_1 -0.009926472
cmunicipio                cmunicipio  0.003314892
pacientes                  pacientes  0.001849233
```

0.8.2 Mutual Information Feature Selection

[40]:
```r
# install the necessary packages if not already installed
if (!require(FSelectorRcpp)) {
  install.packages('FSelectorRcpp')
}

# Load necessary library
library(FSelectorRcpp)
library(ggplot2)
train_data <- train_set

# Calculate mutual information between each variable and the target
mi_scores <- information_gain(train_data[, setdiff(names(train_data),␣
 ↪"Target")], train_data$Target)

# Convert the top_features object into a dataframe
mi_scores_df <- as.data.frame(mi_scores)

# Rename the columns
names(mi_scores_df) <- c("Feature", "Score")
```

```r
# Order the dataframe by Score in descending order
mi_scores_df <- mi_scores_df[order(-mi_scores_df$Score),]

# Create a bar plot
ggplot(mi_scores_df, aes(x = reorder(Feature, Score), y = Score)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  xlab("Features") +
  ylab("Mutual Information Score") +
  ggtitle("Top Features by Mutual Information") +
  theme_minimal()
```

```
Warning message in .information_gain.data.frame(formula, data, type = type,
equal = equal, :
"Dependent variable is a numeric! It will be converted to factor with simple
factor(y). We do not discretize dependent variable in FSelectorRcpp by default!
You can choose equal frequency binning discretization by setting equal argument
to TRUE."
```



0.8.3 Evaluating a Lineal Regression model

[50]:
```
# Select numer of Features to use
k <- 2
```

Selecting feature to use

[51]:
```
# Select numer of Features to use
# Fit a linear regression model
library(tidyverse)
library(caret)

model_all_features <- lm(Target ~ ., data = train_set)
```

```
# Predict on the test set
predictions <- predict(model_all_features, newdata = test_set)

# Evaluate the model
postResample(pred = predictions, obs = test_set$Target)
```

RMSE 35.9881884319083 **Rsquared** 0.282703312782414 **MAE** 27.696087321463

Operation

[78]:
```
# Select the top k features
     k <- 5

# Get the vector of the first k features
top_features <- correlation_df$Feature[1:k]
top_features
# Fit a linear regression model with only the top k features
model_top_features <- lm(Target ~ ., data = train_set[, c(top_features,␣
 ↪"Target")])

# Predict on the test set
predictions_top_features <- predict(model_top_features, newdata = test_set[,␣
 ↪top_features])

# Evaluate the model
postResample(pred = predictions_top_features, obs = test_set$Target)
```

1. 'consultas' 2. 'total_pacientes' 3. 'CAMAS' 4. 'HEM' 5. 'hospitalizaciones'

RMSE 37.9884517440069 **Rsquared** 0.200806958439625 **MAE** 30.2444257238044

0.8.4 Tuning the Number of Selected Features

Know the best number of features to select

See the relationship between the number of selected features and MAE

0.9 Any data: RFE (Recursive Feature Elimination)

0.9.1 RFE for Regression

Selecting feature to use

[ ]:
```
# Select numer of Features to use
k <- 5
```

Operation

[79]:
```
# Define control parameters for rfe function
ctrl <- rfeControl(functions=lmFuncs, method="cv", number=10)

# Determine number of predictors
predictors_number <- ncol(train_set) - 1 # Assuming the last column is the␣
 ↪target variable

# Apply the RFE algorithm with cross validation.
result <- rfe(train_set[, !names(train_set) %in% "Target"], train_set$Target,␣
 ↪sizes=c(1:predictors_number), rfeControl=ctrl)

# Print the result
print(result)

# Top ranking variables in the optimal subset size
top_features <- predictors(result, result$optsize)
```

```
Recursive feature selection

Outer resampling method: Cross-Validated (10 fold)

Resampling performance over subset size:

 Variables  RMSE Rsquared   MAE RMSESD RsquaredSD  MAESD Selected
         1 41.99 0.003049 34.18 0.5618   0.002440 0.2945
         2 41.88 0.008042 34.09 0.5529   0.003176 0.2685
         3 41.76 0.013921 33.99 0.5501   0.005456 0.2896
         4 41.67 0.017936 33.98 0.5043   0.003247 0.2481
         5 41.61 0.020733 34.00 0.4806   0.003814 0.2104
         6 41.50 0.025895 33.87 0.4907   0.005185 0.2280
         7 41.34 0.033713 33.65 0.4796   0.005620 0.1934
         8 41.12 0.043599 33.41 0.5071   0.010784 0.2590
         9 40.66 0.065085 33.06 0.4140   0.010609 0.1964
        10 40.17 0.086948 32.41 0.9213   0.040546 1.0934
        11 39.91 0.098612 32.23 1.0181   0.045953 1.1659
        12 38.29 0.170397 30.25 0.5215   0.036227 0.8328
        13 37.80 0.191790 29.76 0.4026   0.012642 0.3036
        14 37.74 0.194651 29.69 0.4041   0.012843 0.2000
        15 37.70 0.196237 29.70 0.3819   0.012921 0.1868
        16 37.69 0.196876 29.67 0.3910   0.012984 0.1951
        17 37.66 0.198162 29.62 0.3806   0.013048 0.1888
        18 37.64 0.198809 29.62 0.3889   0.013252 0.1976
        19 37.63 0.199463 29.60 0.4014   0.012969 0.1960
        20 37.61 0.200005 29.59 0.4089   0.012964 0.1880
        21 37.57 0.201759 29.57 0.4094   0.013239 0.2180
        22 37.53 0.203417 29.53 0.4161   0.013240 0.2093
        23 37.28 0.214053 29.21 0.6934   0.028548 0.6820
```

```
24 37.20 0.217035 29.15 0.7616   0.031431 0.7156
25 36.20 0.258799 27.94 0.5744   0.017950 0.2615
26 35.96 0.269084 27.83 0.5508   0.013822 0.2316
27 35.87 0.272710 27.78 0.5645   0.014811 0.2337
28 35.76 0.277002 27.74 0.5363   0.012104 0.2118
29 35.75 0.277559 27.73 0.5282   0.011797 0.2104
30 35.75 0.277653 27.73 0.5299   0.011998 0.2132
31 35.74 0.277723 27.73 0.5280   0.011998 0.2116        *
32 35.74 0.277715 27.73 0.5281   0.011965 0.2114
33 35.74 0.277715 27.73 0.5281   0.011965 0.2114
34 35.74 0.277715 27.73 0.5281   0.011965 0.2114
35 35.74 0.277715 27.73 0.5281   0.011965 0.2114
36 35.74 0.277715 27.73 0.5281   0.011965 0.2133
37 35.74 0.277715 27.73 0.5281   0.011965 0.2133
38 35.74 0.277716 27.73 0.5281   0.011964 0.2109

The top 5 variables (out of 31):
   t4_3, t2_2, t4_2, t2_1, t4_1
```

0.10 Data Save

- 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

0.10.1 Proceso 16

[ ]:
```
# caso <- "CU_XX"
# proceso <- '_09.2'
# tarea <- "_XX"
# archivo <- ""
# proper <- "_xxxxx"
# 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/.

[ ]:
```
# 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

[ ]:
```
# 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

```
[ ]:
```