16.- Feature Selection_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 *** > # 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_53_14_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=es_ES.UTF-8;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_14_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_14_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> 0.234430921, -0.247745795,
0.644506738, -0.070067671, -…
$ score_bhn      <dbl> 0.4097479, 0.1290857, 0.5753443,
0.4274030, 0.3293843, …
$ score_fow      <dbl> 0.22131225, -0.67087093, 0.55485637,
-0.04224433, -0.26…
$ score_opp      <dbl> 0.040287945, -0.176082184,
0.684177595, -0.557195503, -…
$ score_nbmc     <dbl> 0.4417846, -0.4611703, 0.4195220,
0.2610630, 0.5105377,…
$ score_ws       <dbl> 0.5398626, 0.3861578, 0.6209430,
0.1056095, 0.1274964, …
$ score_sh       <dbl> 0.6722671, 0.1921862, 0.7589734,
0.5545286, 0.6229812, …
$ score_ps       <dbl> -0.451618611, 0.297686264,
0.192315395, 0.822032832, -0…
$ score_abk      <dbl> 0.038575928, -1.291936532,
0.767026841, -0.404764773, -…
$ score_aic      <dbl> 0.65139291, -1.02544160, 0.37750377,
-0.38712186, 0.831…
$ score_hw       <dbl> -0.17460539, 0.46862381, 0.28376095,
0.31816759, -0.652…
$ score_eq       <dbl> 0.145913492, -0.124265238,
0.496988695, 0.680773448, -0…
$ score_pr       <dbl> 0.49893581, 0.02236525, 0.89103387,
-0.40632266, -0.458…
$ score_pfc      <dbl> -0.17705492, 0.12172033, 0.31379064,
-0.42914696, -0.28…
$ score_incl     <dbl> -0.412651603, 0.380048297,
0.997036708, 0.009655802, -0…
$ score_aae      <dbl> 0.13726735, -1.14969465, 0.14456184,
-1.20857192, -0.38…
$ 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>	s
80	0.23443092	0.4097479	0.22131225	0.04028795	0.4417846	0
97	-0.24774579	0.1290857	-0.67087093	-0.17608218	-0.4611703	0
46	0.64450674	0.5753443	0.55485637	0.68417760	0.4195220	0
84	-0.07006767	0.4274030	-0.04224433	-0.55719550	0.2610630	0
99	-0.16212549	0.3293843	-0.26860033	-0.50440793	0.5105377	0

0.6 Selecting Categorical Input / Categorical Output

[ ]:
```

```

0.6.1 Encoding Categorical Features

0.6.2 Chi-Squared Feature Selection

0.6.3 Mutual Information Feature Selection

0.6.4 Evaluating a Logistic Regression model

Select numer of Features to use

[9]:
```
# Select numer of Features to use
```

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

[10]:
```
# Select numer of Features to use
```

Operation

0.7.4 Tuning the Number of Selected Features

[ ]:
```

```

Know the best number of features to select

[ ]:
```

```

See the relationship between the number of selected features and accuracy

[ ]:

0.8 Selecting Numerical Input / Numerical Output

[8]:
```
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)
```

0.8.1 Correlation with the outcome Feature Selection

[10]:
```
# 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% "rank_score_spi")],␣
 ↪function(x) cor(x, train_set$rank_score_spi))

# 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
score_spi   score_spi  -0.9877809
score_fow   score_fow  -0.9629153
score_aae   score_aae  -0.9306451
score_opp   score_opp  -0.9262783
score_pfc   score_pfc  -0.9198194
score_hw     score_hw  -0.9048625
score_bhn   score_bhn  -0.9030321
score_abk   score_abk  -0.8851855
score_ws     score_ws  -0.8697752
score_ps     score_ps  -0.8652805
score_nbmc score_nbmc  -0.8575689
score_aic   score_aic  -0.8574032
score_sh     score_sh  -0.8374892
score_incl score_incl  -0.8161920
score_pr     score_pr  -0.7163100
score_eq     score_eq  -0.6775241
```

0.8.2 Mutual Information Feature Selection

[22]:
```
# Instala el paquete necesario si aún no está instalado
# Verifica si el paquete FSelectorRcpp ya está instalado.
# Si no está instalado, entonces lo instala.
#if (!require(FSelectorRcpp)) {
#  install.packages('FSelectorRcpp')
#}

# Carga la biblioteca necesaria
# library(FSelectorRcpp)

# Calcula la información mutua entre cada variable y el objetivo
# mi_scores <- information_gain(train_data[, setdiff(names(train_data),␣
 ↪"rank_score_spi")], train_data$Target)

# Convierte el objeto top_features en un marco de datos
# mi_scores_df <- as.data.frame(mi_scores)

# Renombra las columnas
# names(mi_scores_df) <- c("Feature", "Score")

# Ordena el marco de datos por Score en orden descendente
# mi_scores_df <- mi_scores_df[order(-mi_scores_df$Score),]

# Crea un gráfico de barras
# 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()
```

0.8.3 Evaluating a Lineal Regression model

[23]:
```
# Select numer of Features to use
k <- 5
```

Operation

[19]:
```
# Instala el paquete 'caret' si no está ya instalado
if (!require(caret)) {
  install.packages('caret')
}

# Carga la biblioteca 'caret'
```

```r
library(caret)

# Luego puedes usar la función postResample
result <- postResample(pred = predictions, obs = test_set$rank_score_spi)

# Fit a linear regression model
model_all_features <- lm(rank_score_spi ~ ., 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$rank_score_spi)
```

RMSE 6.46463812889704 **Rsquared** 0.982435210421272 **MAE** 4.84989747978453

[ ]:

Selecting feature to use

[11]:
```r
# Select numer of Features to use
```

Operation

[ ]:

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 Classification

Selecting feature to use

[12]:
```r
# Select numer of Features to use
```

Operation

[ ]:

0.9.2 RFE for Regression

Selecting feature to use

[13]:
```r
# Select numer of Features to use
```

Operation

[25]:
```
# 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% "rank_score_spi"],␣
 ↪train_set$rank_score_spi, 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 7.600   0.9760 5.935 0.5291   0.003103 0.3390
         2 7.549   0.9763 5.842 0.6117   0.003604 0.3482
         3 7.286   0.9778 5.594 0.7704   0.004589 0.5326
         4 6.964   0.9799 5.312 0.6802   0.003950 0.5471
         5 6.888   0.9804 5.250 0.6565   0.003818 0.5285
         6 6.822   0.9807 5.163 0.6721   0.003889 0.5432
         7 6.707   0.9814 5.050 0.6659   0.003781 0.5021
         8 6.630   0.9818 4.992 0.6656   0.003814 0.4988
         9 6.629   0.9818 4.982 0.6595   0.003792 0.5042
        10 6.349   0.9834 4.768 0.4688   0.002212 0.2785
        11 6.315   0.9835 4.732 0.4787   0.002228 0.2812
        12 6.297   0.9836 4.718 0.5062   0.002350 0.3259
        13 6.263   0.9838 4.685 0.4843   0.002252 0.3122
        14 6.242   0.9839 4.668 0.4874   0.002244 0.3153        *
        15 6.250   0.9839 4.675 0.5013   0.002287 0.3294
        16 6.251   0.9839 4.677 0.5035   0.002294 0.3323

The top 5 variables (out of 14):
   score_spi, score_fow, score_opp, score_aic, score_bhn
```

0.9.3 Tuning the Number of Selected Features

RFE for Classification Use dataset as example: diabetes

Selecting feature to use

[14]:
```
# Select range [f..l] of Features to evaluate
f=2
l=10
```

Operation

[ ]:
```

```

RFE for Regression Selecting feature to use

[15]:
```
# Select range [f..l] of Features to evaluate
f=2
l=10
```

Operation

[ ]:
```

```

0.9.4 Automatically Select the Number of Features

RFE for Classification

[ ]:
```

```

RFE for Regression

[ ]:
```

```

0.10 Exploring the use of diferent algorithms wrapped by RFE

RFE for Classification

[ ]:
```

```

RFE for Regression Use dataset as example: Boston houses

[ ]:
```

```

0.11 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

[16]:
```
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

0.11.1 Proceso 16

[26]:
```
caso <- "CU_53"
proceso <- '_16'
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

[27]:
```
# 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/.

[28]:
```
# 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

[29]:
```
# path_in <- paste0(iPath, file_save)
# file.copy(path_out, path_in, overwrite = TRUE)
```

0.12 REPORT

A continuación se realizará un informe de las acciones realizadas

0.13 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

0.14 Main Conclusions

- Los datos están listos para la modelización y despliegue

0.15 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

[ ]:
```
```