Artem Urlapov Sedova - Interpretable Machine Learning - Sistema Experto para Predicción y Mejora del Rendimiento Escolar

October 26, 2024

[1]:
```python
#Librerías
import sys
import os
import pathlib
import math
import tensorflow as tf
import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow.python.keras as tfk
import numpy as np
import sklearn as sk
import seaborn as sns
import missingno as msno
import patsy
import statsmodels.api as sm
import random
import shap
import lime

#Funciones
from pathlib import Path
from math import ceil
from numpy import abs, logical_and, nan
from pandas import read_csv, DataFrame, get_dummies
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.inspection import PartialDependenceDisplay
from sklearn.neighbors import LocalOutlierFactor
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.svm import OneClassSVM, SVR
from sklearn.preprocessing import MinMaxScaler, StandardScaler, OneHotEncoder,␣
↪LabelEncoder
```

```python
from scipy.stats.mstats import winsorize
from scipy.stats import expon, reciprocal
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Input
from lime.lime_tabular import LimeTabularExplainer

#Texto centrado
from IPython.core.display import HTML

HTML("""
<style>
    p {
        text-align: justify;
    }
</style>
""")


def printVersion(obj, name):
  if hasattr(obj, '__version__'):
    return f'{name} version: {obj.__version__} \n\n'
  else:
    return f'{name} no tiene un atributo __version__.\n\n'


print(
    printVersion(tf, 'TensorFlow'),
    printVersion(mpl, 'Matplotlib'),
    printVersion(pd, 'Pandas'),
    printVersion(tfk, 'Keras'),
    printVersion(np, 'Numpy'),
    printVersion(sk, 'Sklearn'),
    printVersion(sns, 'Seaborn'),
    printVersion(msno, 'Missingno')
)

print()
```

```
TensorFlow version: 2.16.1

 Matplotlib version: 3.7.2

 Pandas version: 2.0.3

 Keras no tiene un atributo __version__.

 Numpy version: 1.24.3
```

```
Sklearn version: 1.3.0

Seaborn version: 0.12.2

Missingno version: 0.5.2
```

[2]:
```python
new_directory = 'C:/Users/artem/Desktop/IML'

os.chdir(new_directory)

dataset = read_csv('student-mat.csv', sep=';', decimal='.')

print(dataset ,'\n\n\n\n\n')

print(dataset.info())
```

```
    school sex  age address famsize Pstatus  Medu  Fedu      Mjob      Fjob  \
0       GP   F   18       U     GT3       A     4     4   at_home   teacher
1       GP   F   17       U     GT3       T     1     1   at_home     other
2       GP   F   15       U     LE3       T     1     1   at_home     other
3       GP   F   15       U     GT3       T     4     2    health  services
4       GP   F   16       U     GT3       T     3     3     other     other
..     …  ..  …     …     …     …   …   …       …       …
390     MS   M   20       U     LE3       A     2     2  services  services
391     MS   M   17       U     LE3       T     3     1  services  services
392     MS   M   21       R     GT3       T     1     1     other     other
393     MS   M   18       R     LE3       T     3     2  services     other
394     MS   M   19       U     LE3       T     1     1     other   at_home

    … famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0   …      4        3      4     1     1      3        6   5   6   6
1   …      5        3      3     1     1      3        4   5   5   6
2   …      4        3      2     2     3      3       10   7   8  10
3   …      3        2      2     1     1      5        2  15  14  15
4   …      4        3      2     1     2      5        4   6  10  10
..  …    …       …     …   …   …     …       …  ..  ..  ..
390 …      5        5      4     4     5      4       11   9   9   9
391 …      2        4      5     3     4      2        3  14  16  16
392 …      5        5      3     3     3      3        3  10   8   7
393 …      4        4      1     3     4      5        0  11  12  10
394 …      3        2      3     3     3      5        5   8   9   9

[395 rows x 33 columns]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
None
```

[3]:
```python
#Separación del Dataset en variables numéricas y categóricas

for column in dataset:
  if dataset[column].dtype == 'object':
    dataset[column] = dataset[column].astype('category')

print(dataset.info());
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      395 non-null    category
 1   sex         395 non-null    category
 2   age         395 non-null    int64
 3   address     395 non-null    category
 4   famsize     395 non-null    category
 5   Pstatus     395 non-null    category
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    category
 9   Fjob        395 non-null    category
 10  reason      395 non-null    category
 11  guardian    395 non-null    category
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
 15  schoolsup   395 non-null    category
 16  famsup      395 non-null    category
 17  paid        395 non-null    category
 18  activities  395 non-null    category
 19  nursery     395 non-null    category
 20  higher      395 non-null    category
 21  internet    395 non-null    category
 22  romantic    395 non-null    category
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: category(17), int64(16)
memory usage: 58.4 KB
```

None

Exploratory Data Analysis

Parte 1 - Variables Categóricas

[4]:
```
categorical_variables = dataset.select_dtypes(include='category')
print(categorical_variables)
print(categorical_variables.shape)

for catColumn in categorical_variables:
  categorias = dataset[catColumn].cat.categories
  categoriasP = []
  for categoria in categorias:
    categoriasP.append(categoria)
  categoriasP = ' | '.join(map(str, categoriasP))
  print(f'Los posibles valores para la categoria {catColumn.upper()} son:
 ↪\t\t{categoriasP}')


  print('')
```

```
    school sex address famsize Pstatus      Mjob      Fjob  reason guardian  \
0       GP   F       U     GT3       A   at_home   teacher  course   mother
1       GP   F       U     GT3       T   at_home     other  course   father
2       GP   F       U     LE3       T   at_home     other   other   mother
3       GP   F       U     GT3       T    health  services    home   mother
4       GP   F       U     GT3       T     other     other    home   father
..     ...  ..     ...     ...     ...       ...       ...     ...      ...
390     MS   M       U     LE3       A  services  services  course    other
391     MS   M       U     LE3       T  services  services  course   mother
392     MS   M       R     GT3       T     other     other  course    other
393     MS   M       R     LE3       T  services     other  course   mother
394     MS   M       U     LE3       T     other   at_home  course   father

    schoolsup famsup paid activities nursery higher internet romantic
0         yes     no   no         no     yes    yes       no       no
1          no    yes   no         no      no    yes      yes       no
2         yes     no  yes         no     yes    yes      yes       no
3          no    yes  yes        yes     yes    yes      yes      yes
4          no    yes  yes         no     yes    yes       no       no
..        ...    ...  ...        ...     ...    ...      ...      ...
390        no    yes  yes         no     yes    yes       no       no
391        no     no   no         no      no    yes      yes       no
392        no     no   no         no      no    yes       no       no
393        no     no   no         no      no    yes      yes       no
394        no     no   no         no     yes    yes      yes       no

[395 rows x 17 columns]
```

```
(395, 17)
Los posibles valores para la categoria SCHOOL son:             GP | MS

Los posibles valores para la categoria SEX son:                F | M

Los posibles valores para la categoria ADDRESS son:            R | U

Los posibles valores para la categoria FAMSIZE son:            GT3 | LE3

Los posibles valores para la categoria PSTATUS son:            A | T

Los posibles valores para la categoria MJOB son:               at_home | health
| other | services | teacher

Los posibles valores para la categoria FJOB son:               at_home | health
| other | services | teacher

Los posibles valores para la categoria REASON son:             course | home |
other | reputation

Los posibles valores para la categoria GUARDIAN son:           father | mother
| other

Los posibles valores para la categoria SCHOOLSUP son:          no | yes

Los posibles valores para la categoria FAMSUP son:             no | yes

Los posibles valores para la categoria PAID son:               no | yes

Los posibles valores para la categoria ACTIVITIES son:         no | yes

Los posibles valores para la categoria NURSERY son:            no | yes

Los posibles valores para la categoria HIGHER son:             no | yes

Los posibles valores para la categoria INTERNET son:           no | yes

Los posibles valores para la categoria ROMANTIC son:           no | yes
```

[5]:
```
#Descripción de las variables categóricas
print(categorical_variables.describe().T)

print()
```

```
           count unique     top freq
school       395      2      GP  349
sex          395      2       F  208
```

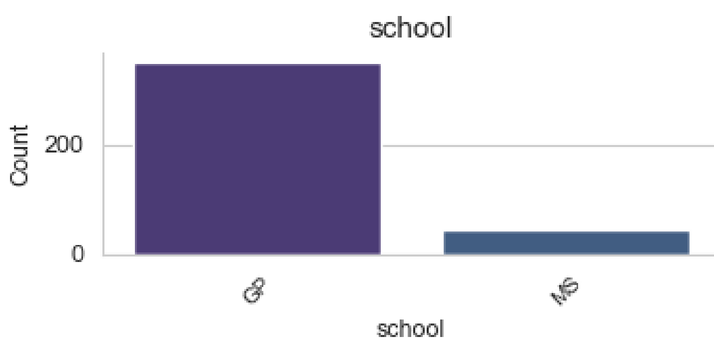
```
address      395   2       U  307
famsize      395   2     GT3  281
Pstatus      395   2       T  354
Mjob         395   5   other  141
Fjob         395   5   other  217
reason       395   4  course  145
guardian     395   3  mother  273
schoolsup    395   2      no  344
famsup       395   2     yes  242
paid         395   2      no  214
activities   395   2     yes  201
nursery      395   2     yes  314
higher       395   2     yes  375
internet     395   2     yes  329
romantic     395   2      no  263
```

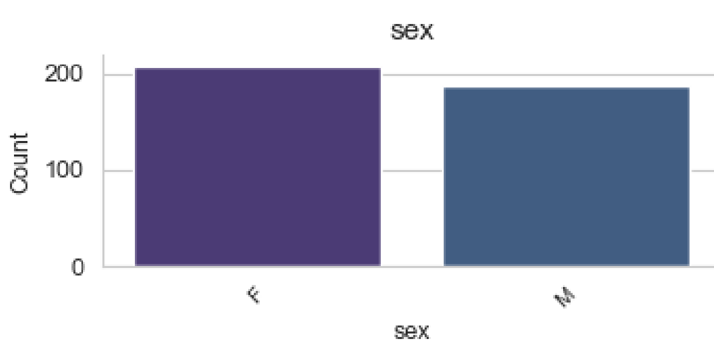
[6]:

```
num_columns = 4
num_rows = int(np.ceil(len(categorical_variables.columns) / num_columns))

fig, axes = plt.subplots(num_rows, num_columns, figsize=(20, num_rows * 5))

for i, cat in enumerate(categorical_variables.columns):
    row = i // num_columns
    col = i % num_columns
    ax = axes[row, col]
    categorical_variables[cat].value_counts().plot.bar(ax=ax)
    ax.set_title(cat)

for j in range(i + 1, num_rows * num_columns):
    fig.delaxes(axes.flatten()[j])

plt.tight_layout()
plt.show()
```

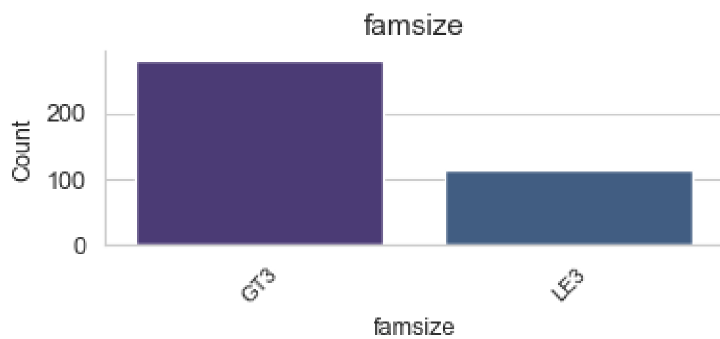
[7]:
```
#Alternativamente - Lo Mismo - En Seaborn

sns.set_style("whitegrid")
sns.set_palette("viridis")
```

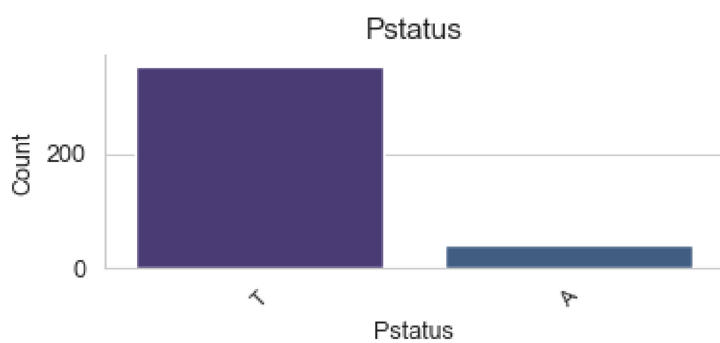
```
for cat in categorical_variables.columns:
    plt.figure(figsize=(4, 2))

    sns.countplot(data=categorical_variables, x=cat,␣
 ↪order=categorical_variables[cat].value_counts().index)

    plt.title(cat, fontsize=11)
    plt.xlabel(cat, fontsize=9)
    plt.ylabel('Count', fontsize=9)
    plt.xticks(rotation=45, fontsize=8)
    plt.yticks(fontsize=9)
    sns.despine()

    plt.tight_layout()
    plt.show()
    print()
```

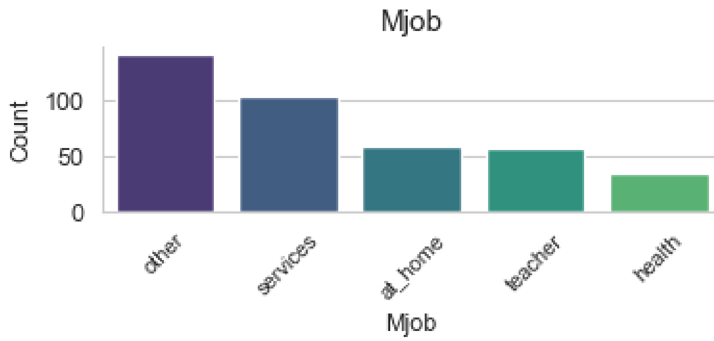


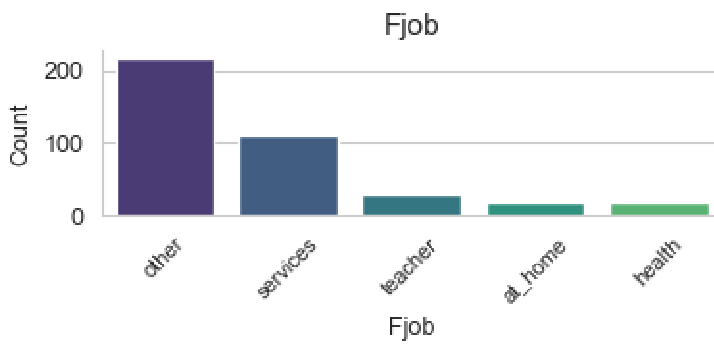


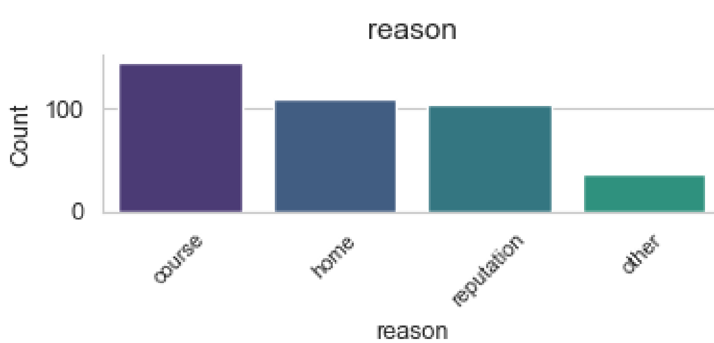

address
Count
200
0
U
R
address

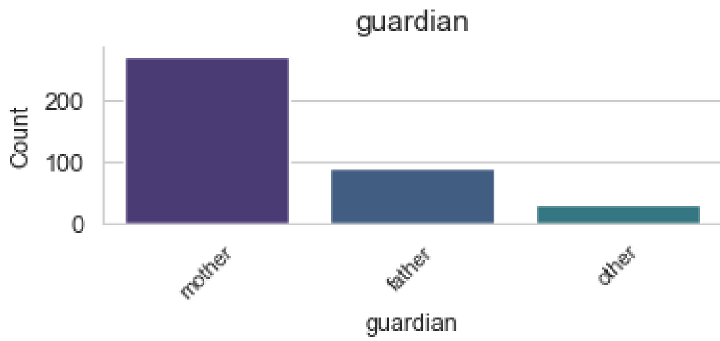

famsize
Count
200
100
0
GT3
LE3
famsize

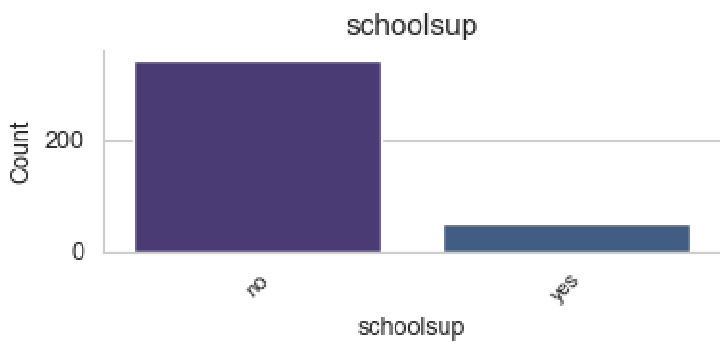

Pstatus
Count
200
0
T
A
Pstatus

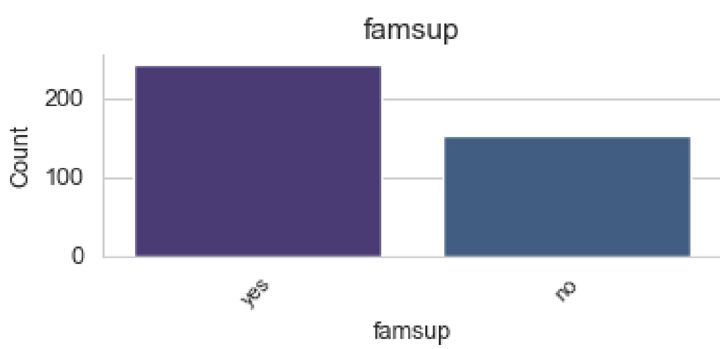

Mjob
Count
100
50
0
other
services
at_home
teacher
health
Mjob


Fjob
Count
200
100
0
other
services
teacher
at_home
health
Fjob

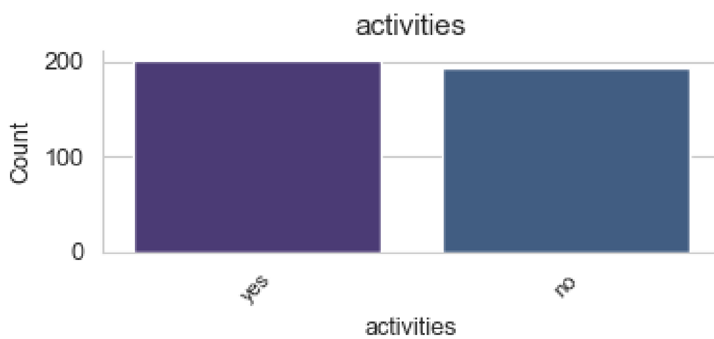

reason
Count
100
0
course
home
reputation
other
reason

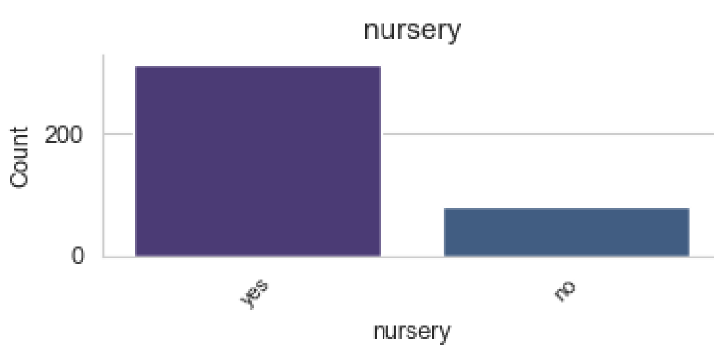

guardian
Count
200
100
0
mother
father
other
guardian

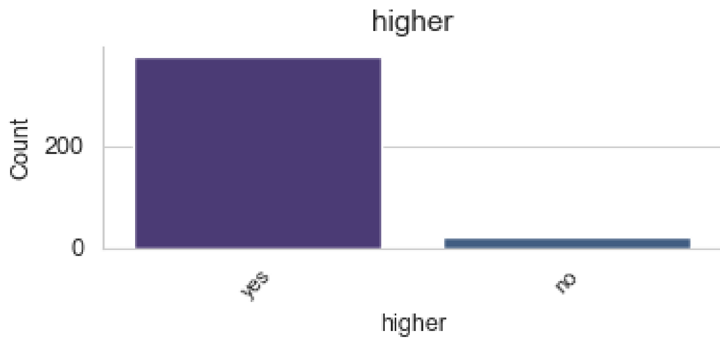

schoolsup
Count
200
0
no
yes
schoolsup

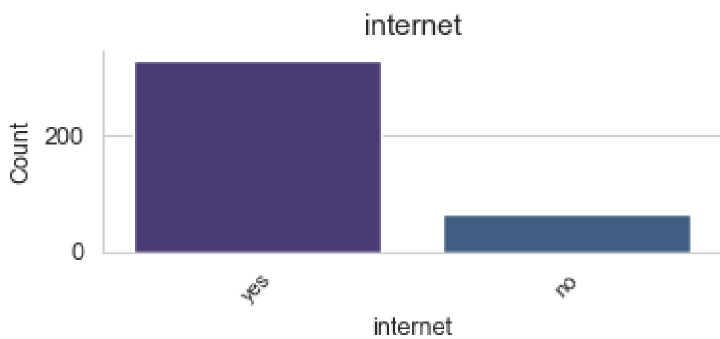

famsup
Count
200
100
0
yes
no
famsup

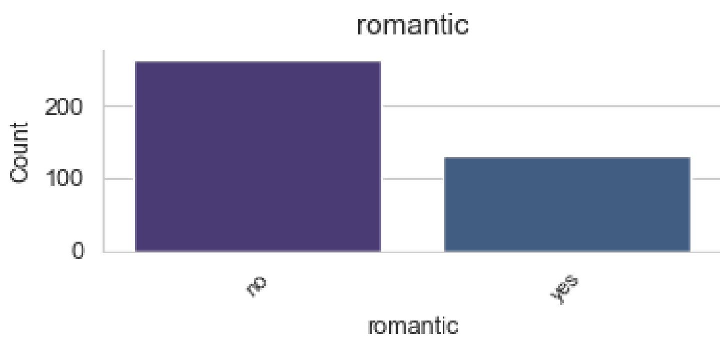

paid
200
100
0
Count
no
yes
paid


activities
200
100
0
Count
yes
no
activities


nursery
200
0
Count
yes
no
nursery

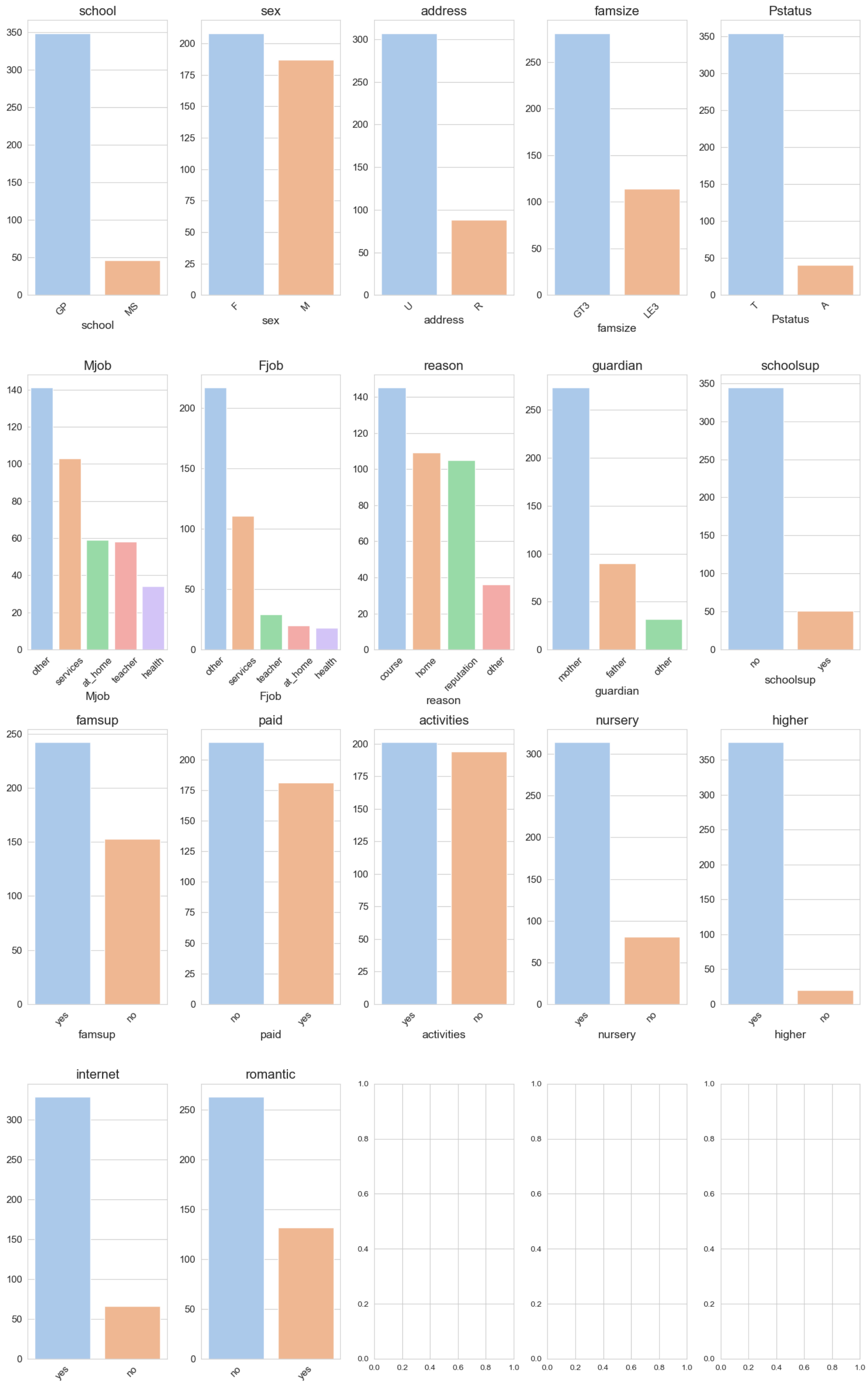






[8]:
```
sns.set_style("whitegrid")
sns.set_palette("pastel")
```

```
n_cols = 5
n_rows = ceil(categorical_variables.columns.size / n_cols)

fig, axs = plt.subplots(n_rows, n_cols, figsize=(15, 6 * n_rows))

for i, cat in enumerate(categorical_variables.columns):
    row = i // n_cols
    col = i % n_cols

    sns.countplot(data=categorical_variables, x=cat, ax=axs[row][col], 
 ↪order=categorical_variables[cat].value_counts().index)

    axs[row][col].set_title(cat, fontsize=16)
    axs[row][col].set_xlabel(cat, fontsize=14)
    axs[row][col].set_ylabel('', fontsize=14)
    axs[row][col].tick_params(axis='x', rotation=45, labelsize=12)
    axs[row][col].tick_params(axis='y', labelsize=12)

plt.tight_layout()
plt.show()
```


school
GP
MS
school
sex
F
M
sex
address
U
R
address
famsize
GT3
LE3
famsize
Pstatus
T
A
Pstatus
Mjob
other
services
at_home
teacher
health
Mjob
Fjob
other
services
teacher
at_home
health
Fjob
reason
course
home
reputation
other
reason
guardian
mother
father
other
guardian
schoolsup
no
yes
schoolsup
famsup
yes
no
famsup
paid
no
yes
paid
activities
yes
no
activities
nursery
yes
no
nursery
higher
yes
no
higher
internet
yes
no
internet
romantic
no
yes
romantic

Parte 2 - Variables Numéricas

[9]:
```
def showInfo(catName, data):
  print(f'\n\n {catName}')
␣
↪print("--------------------------------------------------------------------------------------------------------------------
↪\n\n")
  print(data)

nums_values = dataset.select_dtypes(include = 'number')

showInfo('INFO GENERAL VARIABLES NUMERICAS',nums_values)
showInfo('DESCRIPCION DE LAS VARIABLES NUMERICAS',nums_values.describe().T)
showInfo('CORRELACIONES DE LAS VARIABLES NUMERICAS',nums_values.corr())

#El método default en Python es Pearson

#Añadimos el método Spearman para captar tanto las relaciones lineales como no␣
↪lineales

showInfo('CORRELACIONES SPEARMAN', nums_values.corr(method='spearman'))
```

```
 INFO GENERAL VARIABLES NUMERICAS
--------------------------------------------------------------------------------
--------------------------


     age  Medu  Fedu  traveltime  studytime  failures  famrel  freetime  \
0     18     4     4           2          2         0       4         3
1     17     1     1           1          2         0       5         3
2     15     1     1           1          2         3       4         3
3     15     4     2           1          3         0       3         2
4     16     3     3           1          2         0       4         3
..   …    …    …         …        …       …      …        …
390   20     2     2           1          2         2       5         5
391   17     3     1           2          1         0       2         4
392   21     1     1           1          1         3       5         5
393   18     3     2           3          1         0       4         4
394   19     1     1           1          1         0       3         2

     goout  Dalc  Walc  health  absences  G1  G2  G3
0        4     1     1       3         6   5   6   6
1        3     1     1       3         4   5   5   6
```

```
2        2     2     3       3         10   7   8  10
3        2     1     1       5          2  15  14  15
4        2     1     2       5          4   6  10  10
..     …   …     …       …      …   ..   ..   ..
390      4     4     5       4         11   9   9   9
391      5     3     4       2          3  14  16  16
392      3     3     3       3          3  10   8   7
393      1     3     4       5          0  11  12  10
394      3     3     3       5          5   8   9   9

[395 rows x 16 columns]
```

```
 DESCRIPCION DE LAS VARIABLES NUMERICAS
--------------------------------------------------------------------------------
--------------------------


            count       mean       std   min   25%   50%   75%   max
age         395.0  16.696203  1.276043  15.0  16.0  17.0  18.0  22.0
Medu        395.0   2.749367  1.094735   0.0   2.0   3.0   4.0   4.0
Fedu        395.0   2.521519  1.088201   0.0   2.0   2.0   3.0   4.0
traveltime  395.0   1.448101  0.697505   1.0   1.0   1.0   2.0   4.0
studytime   395.0   2.035443  0.839240   1.0   1.0   2.0   2.0   4.0
failures    395.0   0.334177  0.743651   0.0   0.0   0.0   0.0   3.0
famrel      395.0   3.944304  0.896659   1.0   4.0   4.0   5.0   5.0
freetime    395.0   3.235443  0.998862   1.0   3.0   3.0   4.0   5.0
goout       395.0   3.108861  1.113278   1.0   2.0   3.0   4.0   5.0
Dalc        395.0   1.481013  0.890741   1.0   1.0   1.0   2.0   5.0
Walc        395.0   2.291139  1.287897   1.0   1.0   2.0   3.0   5.0
health      395.0   3.554430  1.390303   1.0   3.0   4.0   5.0   5.0
absences    395.0   5.708861  8.003096   0.0   0.0   4.0   8.0  75.0
G1          395.0  10.908861  3.319195   3.0   8.0  11.0  13.0  19.0
G2          395.0  10.713924  3.761505   0.0   9.0  11.0  13.0  19.0
G3          395.0  10.415190  4.581443   0.0   8.0  11.0  14.0  20.0
```

```
 CORRELACIONES DE LAS VARIABLES NUMERICAS
--------------------------------------------------------------------------------
--------------------------


                 age      Medu      Fedu  traveltime  studytime  failures  \
age         1.000000 -0.163658 -0.163438    0.070641  -0.004140  0.243665
Medu       -0.163658  1.000000  0.623455   -0.171639   0.064944 -0.236680
Fedu       -0.163438  0.623455  1.000000   -0.158194  -0.009175 -0.250408
traveltime  0.070641 -0.171639 -0.158194    1.000000  -0.100909  0.092239
studytime  -0.004140  0.064944 -0.009175   -0.100909   1.000000 -0.173563
```

```
failures    0.243665 -0.236680 -0.250408    0.092239  -0.173563  1.000000
famrel      0.053940 -0.003914 -0.001370   -0.016808   0.039731 -0.044337
freetime    0.016434  0.030891 -0.012846   -0.017025  -0.143198  0.091987
goout       0.126964  0.064094  0.043105    0.028540  -0.063904  0.124561
Dalc        0.131125  0.019834  0.002386    0.138325  -0.196019  0.136047
Walc        0.117276 -0.047123 -0.012631    0.134116  -0.253785  0.141962
health     -0.062187 -0.046878  0.014742    0.007501  -0.075616  0.065827
absences    0.175230  0.100285  0.024473   -0.012944  -0.062700  0.063726
G1         -0.064081  0.205341  0.190270   -0.093040   0.160612 -0.354718
G2         -0.143474  0.215527  0.164893   -0.153198   0.135880 -0.355896
G3         -0.161579  0.217147  0.152457   -0.117142   0.097820 -0.360415

              famrel  freetime     goout      Dalc      Walc    health  \
age         0.053940  0.016434  0.126964  0.131125  0.117276 -0.062187
Medu       -0.003914  0.030891  0.064094  0.019834 -0.047123 -0.046878
Fedu       -0.001370 -0.012846  0.043105  0.002386 -0.012631  0.014742
traveltime -0.016808 -0.017025  0.028540  0.138325  0.134116  0.007501
studytime   0.039731 -0.143198 -0.063904 -0.196019 -0.253785 -0.075616
failures   -0.044337  0.091987  0.124561  0.136047  0.141962  0.065827
famrel      1.000000  0.150701  0.064568 -0.077594 -0.113397  0.094056
freetime    0.150701  1.000000  0.285019  0.209001  0.147822  0.075733
goout       0.064568  0.285019  1.000000  0.266994  0.420386 -0.009577
Dalc       -0.077594  0.209001  0.266994  1.000000  0.647544  0.077180
Walc       -0.113397  0.147822  0.420386  0.647544  1.000000  0.092476
health      0.094056  0.075733 -0.009577  0.077180  0.092476  1.000000
absences   -0.044354 -0.058078  0.044302  0.111908  0.136291 -0.029937
G1          0.022168  0.012613 -0.149104 -0.094159 -0.126179 -0.073172
G2         -0.018281 -0.013777 -0.162250 -0.064120 -0.084927 -0.097720
G3          0.051363  0.011307 -0.132791 -0.054660 -0.051939 -0.061335

            absences        G1        G2        G3
age         0.175230 -0.064081 -0.143474 -0.161579
Medu        0.100285  0.205341  0.215527  0.217147
Fedu        0.024473  0.190270  0.164893  0.152457
traveltime -0.012944 -0.093040 -0.153198 -0.117142
studytime  -0.062700  0.160612  0.135880  0.097820
failures    0.063726 -0.354718 -0.355896 -0.360415
famrel     -0.044354  0.022168 -0.018281  0.051363
freetime   -0.058078  0.012613 -0.013777  0.011307
goout       0.044302 -0.149104 -0.162250 -0.132791
Dalc        0.111908 -0.094159 -0.064120 -0.054660
Walc        0.136291 -0.126179 -0.084927 -0.051939
health     -0.029937 -0.073172 -0.097720 -0.061335
absences    1.000000 -0.031003 -0.031777  0.034247
G1         -0.031003  1.000000  0.852118  0.801468
G2         -0.031777  0.852118  1.000000  0.904868
G3          0.034247  0.801468  0.904868  1.000000
```

CORRELACIONES SPEARMAN
----

	age	Medu	Fedu	traveltime	studytime	failures
age	1.000000	-0.161294	-0.149596	0.109804	0.031557	0.236464
Medu	-0.161294	1.000000	0.631577	-0.147849	0.063498	-0.242373
Fedu	-0.149596	0.631577	1.000000	-0.154454	0.018429	-0.236616
traveltime	0.109804	-0.147849	-0.154454	1.000000	-0.105969	0.079917
studytime	0.031557	0.063498	0.018429	-0.105969	1.000000	-0.157633
failures	0.236464	-0.242373	-0.236616	0.079917	-0.157633	1.000000
famrel	0.031380	0.012361	0.011400	-0.038656	0.058141	-0.051389
freetime	0.000302	0.028493	-0.017132	-0.022279	-0.131321	0.088058
goout	0.140131	0.064954	0.047961	-0.001430	-0.065979	0.105419
Dalc	0.097073	0.022729	0.003994	0.066477	-0.217904	0.187492
Walc	0.132799	-0.044332	-0.014486	0.063654	-0.264021	0.127912
health	-0.075150	-0.035686	0.018113	-0.015452	-0.091497	0.079688
absences	0.149276	0.097562	0.003568	-0.025061	-0.046180	0.096028
G1	-0.057630	0.209662	0.194737	-0.085501	0.162286	-0.346052
G2	-0.167622	0.236354	0.194844	-0.123795	0.129160	-0.362357
G3	-0.173438	0.225036	0.170049	-0.120530	0.105170	-0.361224

	famrel	freetime	goout	Dalc	Walc	health
age	0.031380	0.000302	0.140131	0.097073	0.132799	-0.075150
Medu	0.012361	0.028493	0.064954	0.022729	-0.044332	-0.035686
Fedu	0.011400	-0.017132	0.047961	0.003994	-0.014486	0.018113
traveltime	-0.038656	-0.022279	-0.001430	0.066477	0.063654	-0.015452
studytime	0.058141	-0.131321	-0.065979	-0.217904	-0.264021	-0.091497
failures	-0.051389	0.088058	0.105419	0.187492	0.127912	0.079688
famrel	1.000000	0.143142	0.063549	-0.106338	-0.116060	0.085341
freetime	0.143142	1.000000	0.285182	0.194223	0.130246	0.088975
goout	0.063549	0.285182	1.000000	0.255146	0.393333	-0.018541
Dalc	-0.106338	0.194223	0.255146	1.000000	0.639906	0.095139
Walc	-0.116060	0.130246	0.393333	0.639906	1.000000	0.093625
health	0.085341	0.088975	-0.018541	0.095139	0.093625	1.000000
absences	-0.086577	0.013397	0.133280	0.129651	0.208508	-0.070132
G1	0.026433	0.006973	-0.151636	-0.111438	-0.108368	-0.052224
G2	0.008165	-0.016765	-0.160985	-0.110086	-0.109144	-0.050900
G3	0.054977	-0.004994	-0.166119	-0.120944	-0.104459	-0.047790

	absences	G1	G2	G3
age	0.149276	-0.057630	-0.167622	-0.173438
Medu	0.097562	0.209662	0.236354	0.225036
Fedu	0.003568	0.194737	0.194844	0.170049
traveltime	-0.025061	-0.085501	-0.123795	-0.120530
studytime	-0.046180	0.162286	0.129160	0.105170

```
failures    0.096028 -0.346052 -0.362357 -0.361224
famrel     -0.086577  0.026433  0.008165  0.054977
freetime    0.013397  0.006973 -0.016765 -0.004994
goout       0.133280 -0.151636 -0.160985 -0.166119
Dalc        0.129651 -0.111438 -0.110086 -0.120944
Walc        0.208508 -0.108368 -0.109144 -0.104459
health     -0.070132 -0.052224 -0.050900 -0.047790
absences    1.000000  0.004479 -0.033600  0.017731
G1          0.004479  1.000000  0.894792  0.878001
G2         -0.033600  0.894792  1.000000  0.957125
G3          0.017731  0.878001  0.957125  1.000000
```

[10]:
```python
#Correlaciones Pearson

def showRelevantCorrsPearson(higher, lower):
  corrMatrixPearson = nums_values.corr()
  correlacionesAltasPearson = corrMatrixPearson[(corrMatrixPearson > higher) &␣
 ↪(corrMatrixPearson <= 1)]
  correlacionesBajasPearson = corrMatrixPearson[(corrMatrixPearson < lower) &␣
 ↪(corrMatrixPearson >= -1)]
  return [correlacionesAltasPearson, correlacionesBajasPearson]

def cleaningCorrsPearson(corr):

  return corr.unstack().sort_values().drop_duplicates().dropna();


relevant_pearson = showRelevantCorrsPearson(0.5, -0.2)
print('CORRELACIONES DIRECTAS MÉTODO DE PEARSON␣
 ↪\n\n',cleaningCorrsPearson(relevant_pearson[0]),
      '\n\n\nCORRELACIONES INVERSAS MÉTODO DE PEARSON \n\n',␣
 ↪cleaningCorrsPearson(relevant_pearson[1]))
```

```
CORRELACIONES DIRECTAS MÉTODO DE PEARSON

 Medu  Fedu    0.623455
Dalc  Walc    0.647544
G3    G1      0.801468
G2    G1      0.852118
      G3      0.904868
      G2      1.000000
dtype: float64


CORRELACIONES INVERSAS MÉTODO DE PEARSON

 failures   G3         -0.360415
           G2         -0.355896
```

```
           G1          -0.354718
studytime  Walc        -0.253785
Fedu       failures    -0.250408
Medu       failures    -0.236680
dtype: float64
```

[11]:
```python
#Correlaciones Spearman

def showRelevantCorrsSpearman(higher, lower):
  corrMatrixSpearman = nums_values.corr(method='spearman')
  correlacionesAltasSpearman = corrMatrixSpearman[(corrMatrixSpearman > higher)␣
↪& (corrMatrixSpearman <= 1)]
  correlacionesBajasSpearman = corrMatrixSpearman[(corrMatrixSpearman < lower)␣
↪& (corrMatrixSpearman >= -1)]
  return [correlacionesAltasSpearman, correlacionesBajasSpearman]

def cleaningCorrsSpearman(corr):

  return corr.unstack().sort_values().drop_duplicates().dropna();


relevant_spearman = showRelevantCorrsSpearman(0.5, -0.2)
print('CORRELACIONES DIRECTAS MÉTODO DE SPEARMAN␣
↪\n\n',cleaningCorrsSpearman(relevant_spearman[0]),
      '\n\n\nCORRELACIONES INVERSAS MÉTODO DE SPEARMAN \n\n',␣
↪cleaningCorrsSpearman(relevant_spearman[1]))
```

```
CORRELACIONES DIRECTAS MÉTODO DE SPEARMAN

 Medu  Fedu    0.631577
Dalc  Walc    0.639906
G3    G1      0.878001
G2    G1      0.894792
      G3      0.957125
      G2      1.000000
dtype: float64


CORRELACIONES INVERSAS MÉTODO DE SPEARMAN

 failures   G2          -0.362357
           G3          -0.361224
           G1          -0.346052
studytime  Walc        -0.264021
Medu       failures    -0.242373
Fedu       failures    -0.236616
studytime  Dalc        -0.217904
dtype: float64
```

Si bien es bueno y útil comprobar tanto las posibles relaciones lineales como no lineales, podemos observar que no hay una diferencia significativa en los coeficientes obtenidos de forma general (aunque hay algunas excepciones).

De esta forma, podemos observar:

1 - Notas.

Codificadas como G1 (Primer Período), G2 (Segundo Período) y G3 (Nota Final).

Correlación directa muy alta entre G1 (igual a 0.85) y G2, y correlación todavía más alta entre G2 (0.90 según el método de Pearson y 0.95 según el método de Spearman) y G3.

Mientras que la correlación directa entre G1 y G3 es muy alta (igual a 0.80), es cierto también que, como hemos visto antes, influye más G2 (Segundo Período) a efectos de la Nota Final.

2 - Notas y Fracasos.

Resulta interesante observar que, en cuanto a la correlación inversa, el número de fracasos afecta negativamente de una forma muy similar (en -0.35 para G1, G2 y G3, con apenas pequeñas variaciones decimales de diferencia).

3 - Consumo de alcohol.

Observamos que el consumo de alcohol entre semana (variable DALC) y los fines de semana (variable WALC) está directamente correlacionado en apróximadamente dos tercios (el coeficiente es igual a 0.65).

4 - Consumo de alcohol y tiempo de estudio.

Se puede ver una ligera correlación negativa lineal entre el consumo de alcohol los fines de semana y el tiempo de estudio (de -0.25), así como una correlación negativa de Spearman de -0.21 en cuanto al consumo de alcohol entre semana y el tiempo de estudio.

5 - Educación de la madre y la del padre.

Variables (MEDU y FEDU) codificadas en 5 niveles (0 = sin educación, 1 = más bajo, 4 = más alto).

Hay una correlación directa positiva (igual a 0.62).

6 - Educación padres y fracasos.

En ambos casos hay una correlación negativa de -0.25 con la variable 'failures'.

Teniendo en cuenta la presencia de niveles, sería conveniente explorar esta dependencia más en detalle.

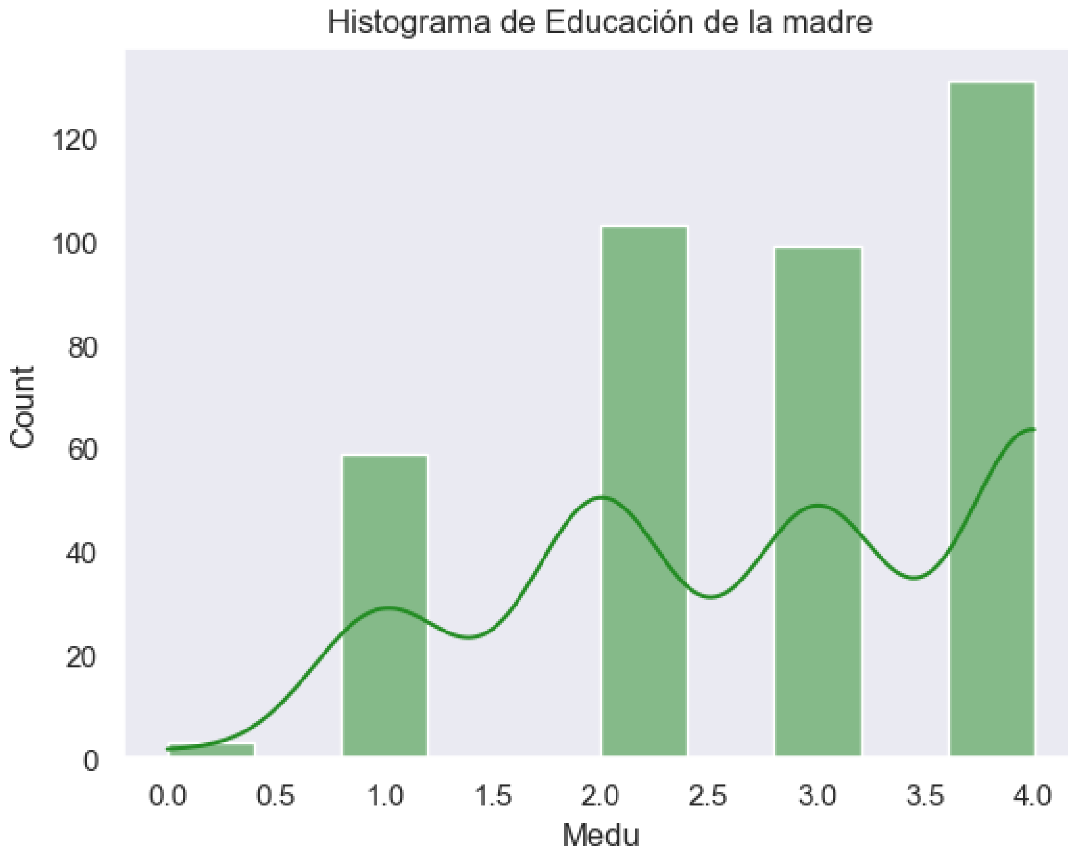
A continuación pues, vamos a explorar algunas de las correlaciones más significativas, que han sido descritas previamente:

[12]:
```
titles = {
    "age": ["Edad del estudiante", "numérica: de 15 a 22"],
    "Medu": ["Educación de la madre", "numérica: 0 - ninguna, 1 - educación␣
 ↪primaria (4to grado), 2 - 5to a 9no grado, 3 - educación secundaria, 4 -␣
 ↪educación superior"],
```

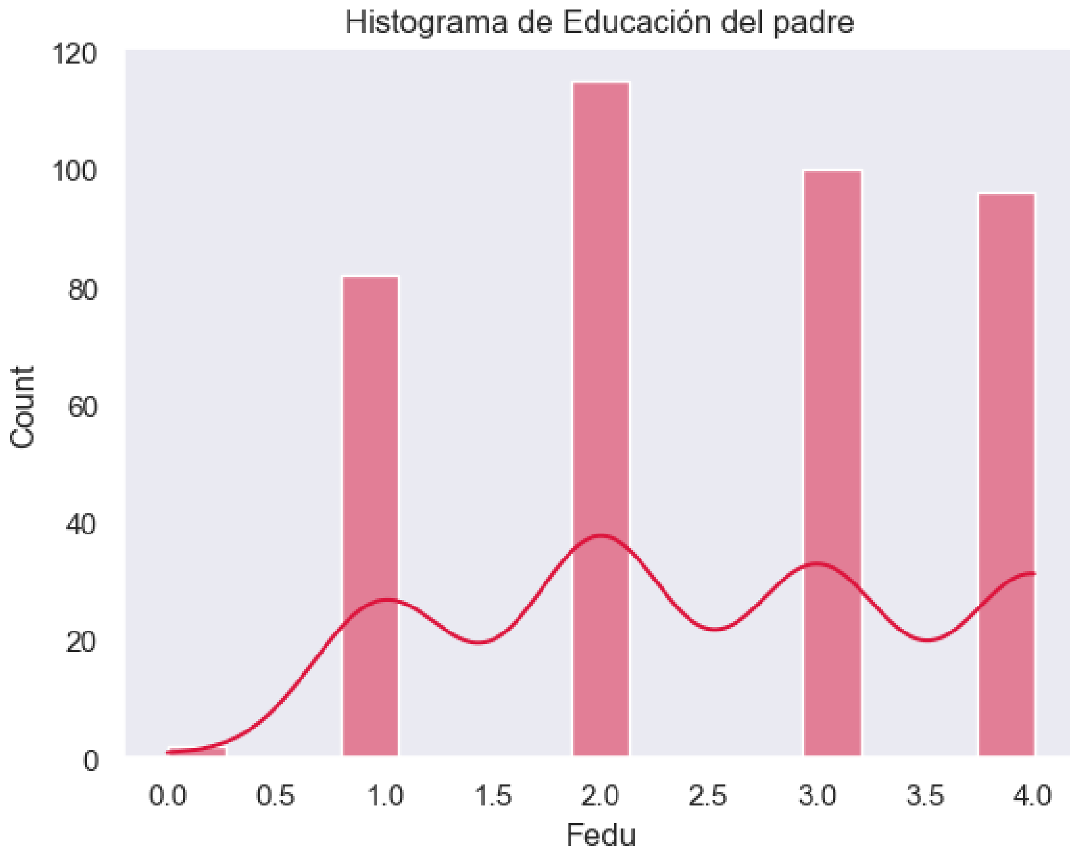
```
    "Fedu": ["Educación del padre", "numérica: 0 - ninguna, 1 - educación␣
↪primaria (4to grado), 2 - 5to a 9no grado, 3 - educación secundaria, 4 -␣
↪educación superior"],
    "studytime": ["Tiempo de estudio semanal", "numérica: 1 - <2 horas, 2 - 2 a␣
↪5 horas, 3 - 5 a 10 horas, 4 - >10 horas"],
    "failures": ["Número de fracasos en clases anteriores", "numérica: n si␣
↪1<=n<3, de lo contrario 4"],
    "Dalc": ["Consumo de alcohol entre semana", "numérica: de 1 - muy bajo a 5␣
↪- muy alto"],
    "Walc": ["Consumo de alcohol los fines de semana", "numérica: de 1 - muy␣
↪bajo a 5 - muy alto"],
    "G1": ["Nota del primer período", "numérica: de 0 a 20"],
    "G2": ["Nota del segundo período", "numérica: de 0 a 20"],
    "G3": ["Nota final", "objetivo de salida, numérica: de 0 a 20"],
    "outlier": ["Indica si es outlier", "1 para valores que no lo son, -1 para␣
↪valores outliers"]
}

colors = ['midnightblue', 'forestgreen', 'crimson', 'indigo']

#Omitimos algunas variables a fin de mostrar sólo las más relevantes
skip_keys = ["traveltime", "famrel", "freetime", "goout", "health", "absences"]

for i, n in enumerate(nums_values):
    if n in skip_keys:
        continue

    sns.set_theme(style='dark')
    print(titles[n][0])
    print(titles[n][1])

    sns.histplot(nums_values[n], kde=True, color=colors[i%4])
    plt.title(f'Histograma de {titles[n][0]}')
    plt.show()
```

```
Edad del estudiante
numérica: de 15 a 22
```

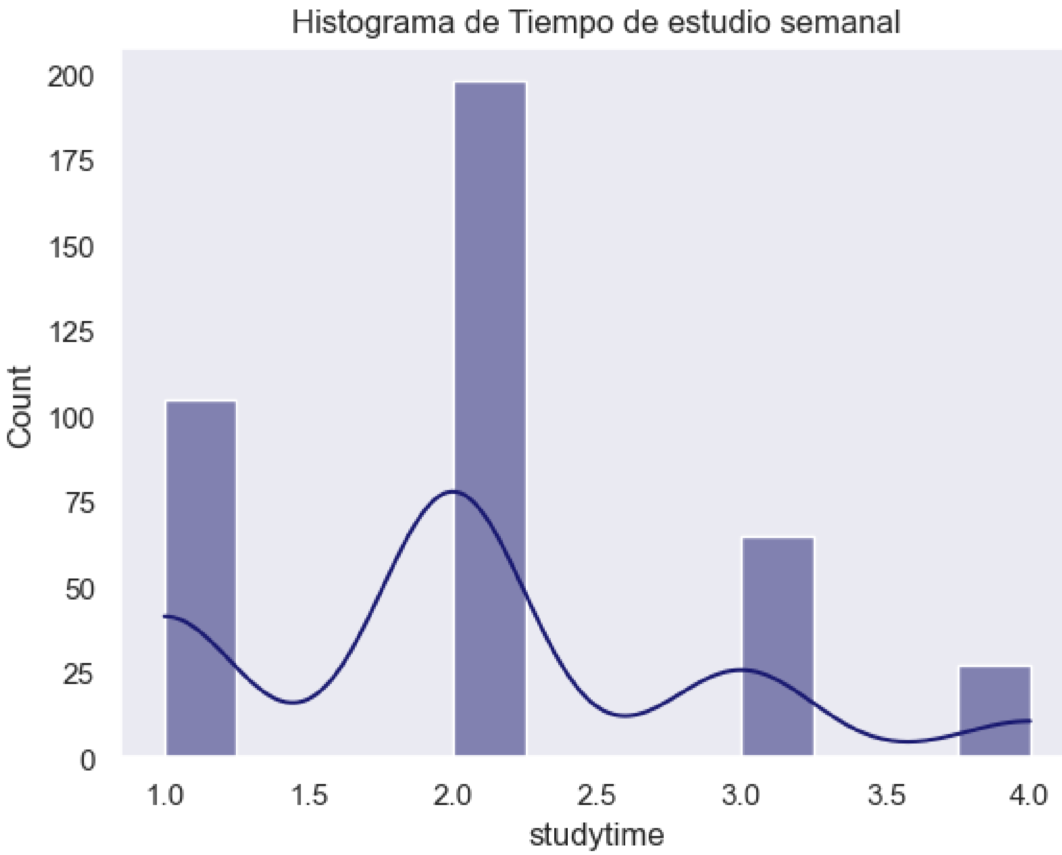
Educación de la madre
numérica: 0 - ninguna, 1 - educación primaria (4to grado), 2 - 5to a 9no grado,
3 - educación secundaria, 4 - educación superior



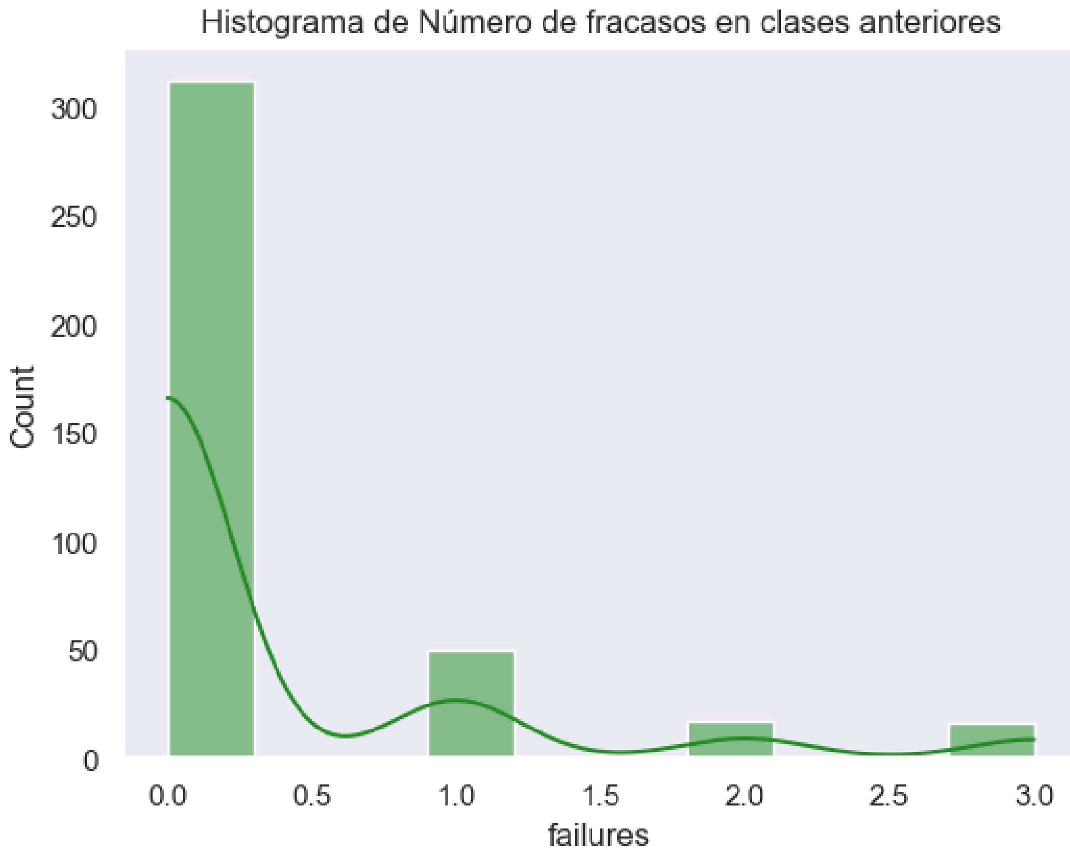
Educación del padre
numérica: 0 - ninguna, 1 - educación primaria (4to grado), 2 - 5to a 9no grado, 3 - educación secundaria, 4 - educación superior



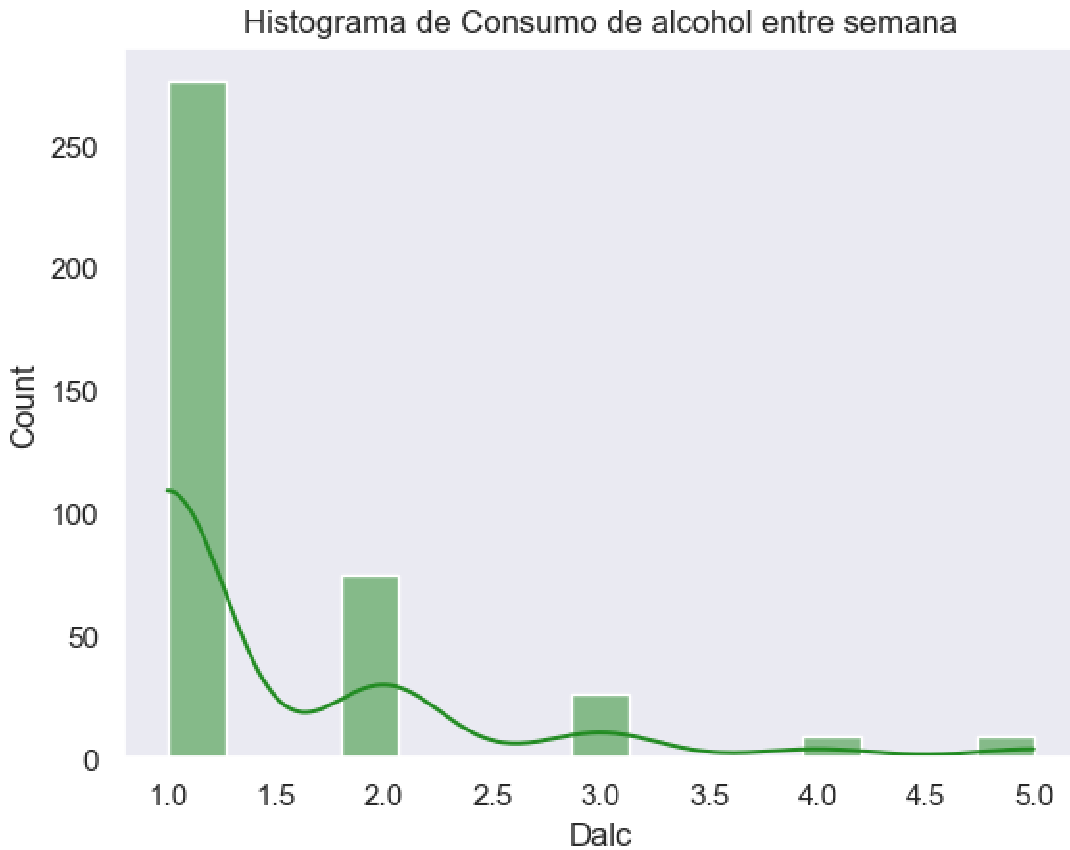
`Tiempo de estudio semanal`
`numérica: 1 - <2 horas, 2 - 2 a 5 horas, 3 - 5 a 10 horas, 4 - >10 horas`



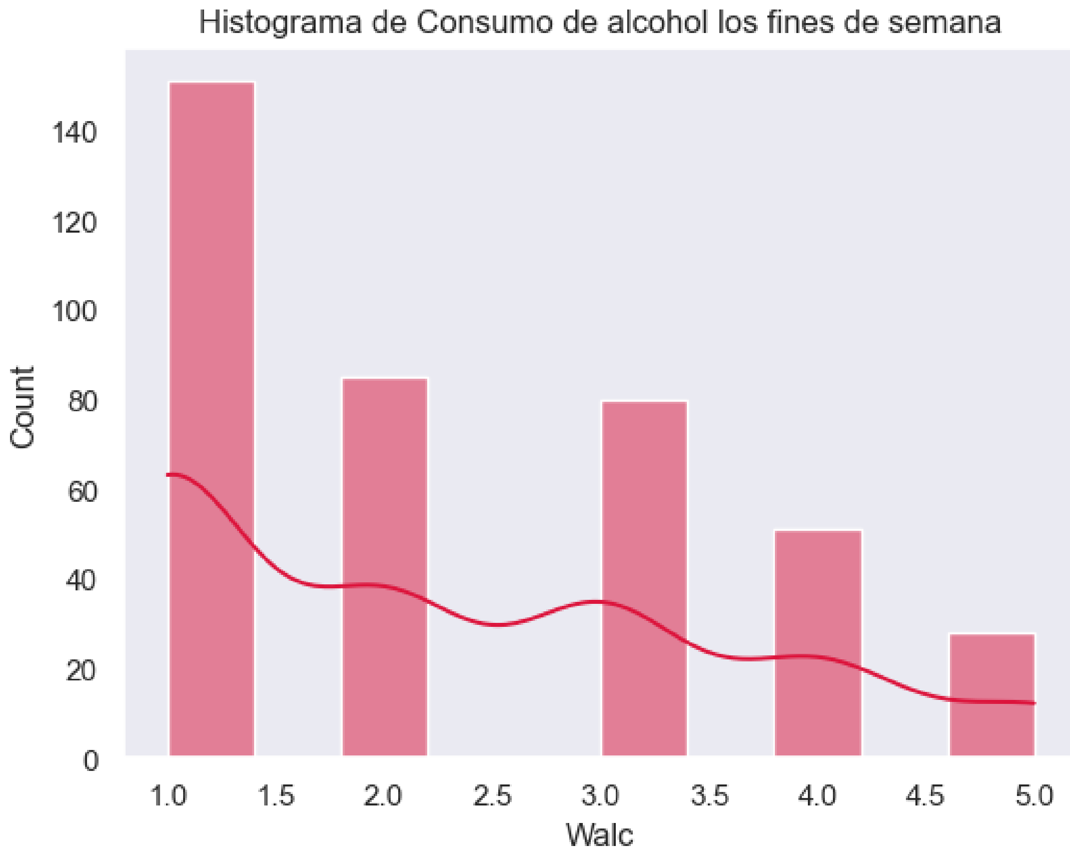
Número de fracasos en clases anteriores
numérica: n si 1<=n<3, de lo contrario 4



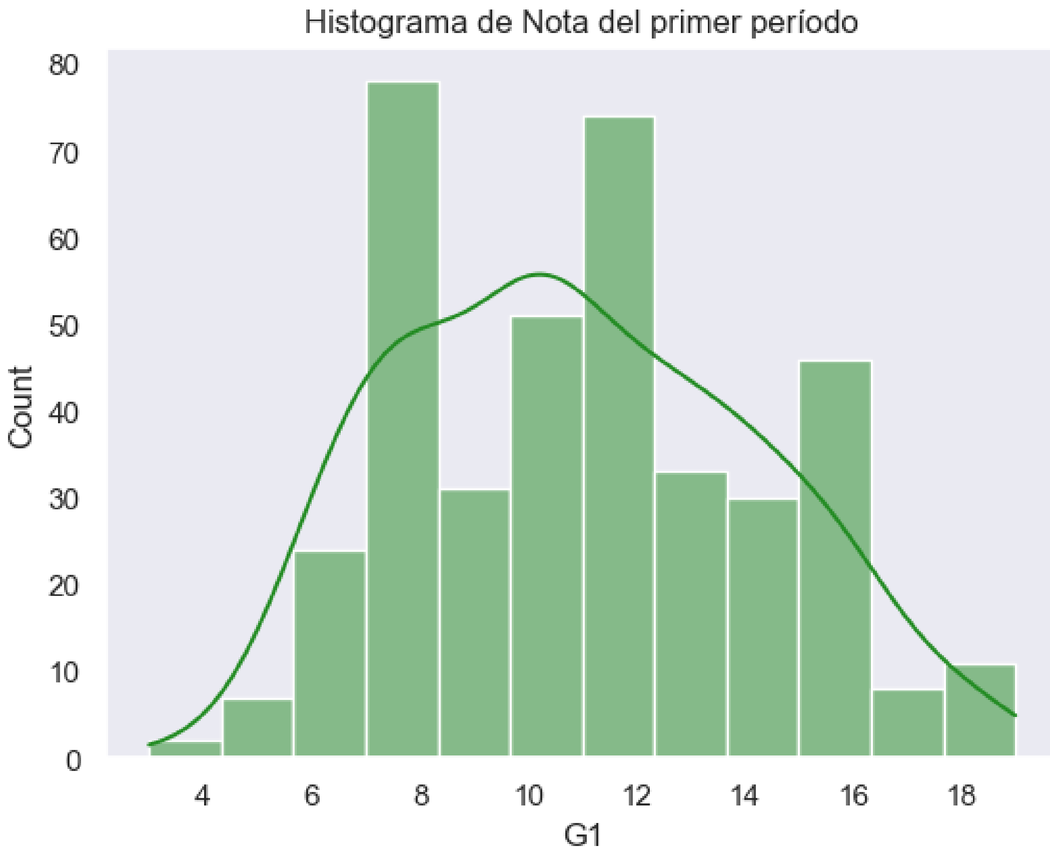

```
Consumo de alcohol entre semana
numérica: de 1 - muy bajo a 5 - muy alto
```



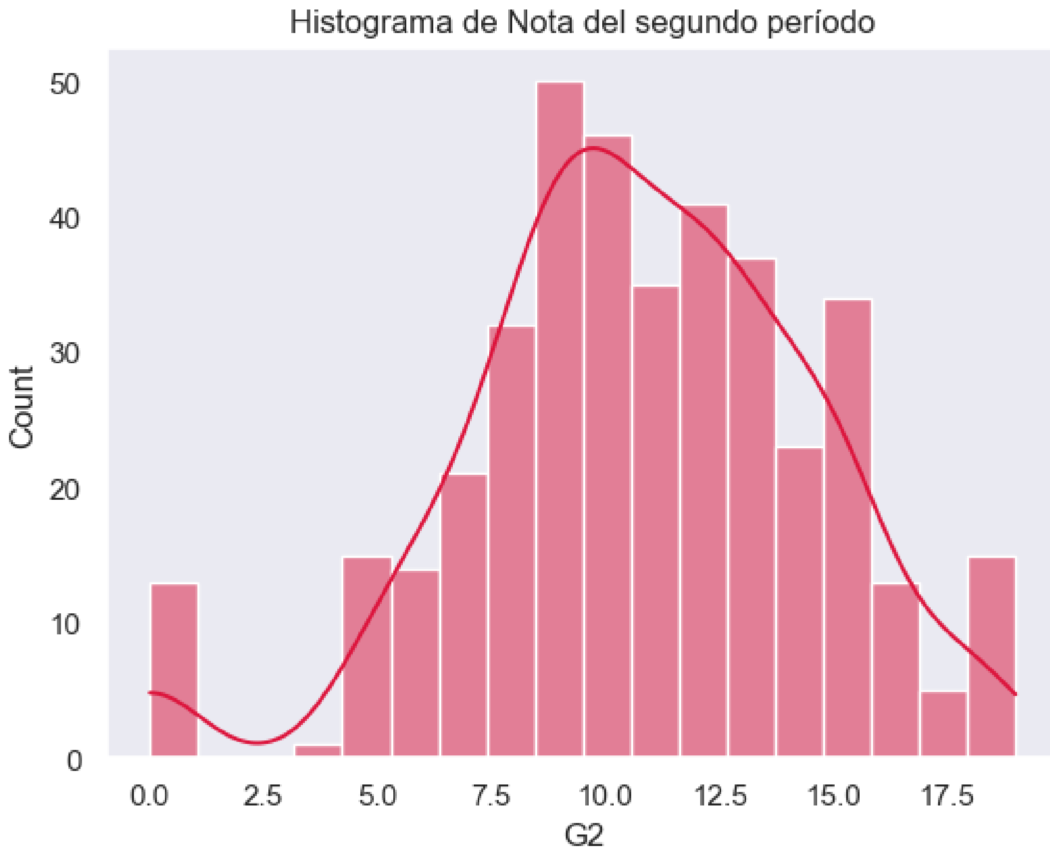

```
Consumo de alcohol los fines de semana
numérica: de 1 - muy bajo a 5 - muy alto
```



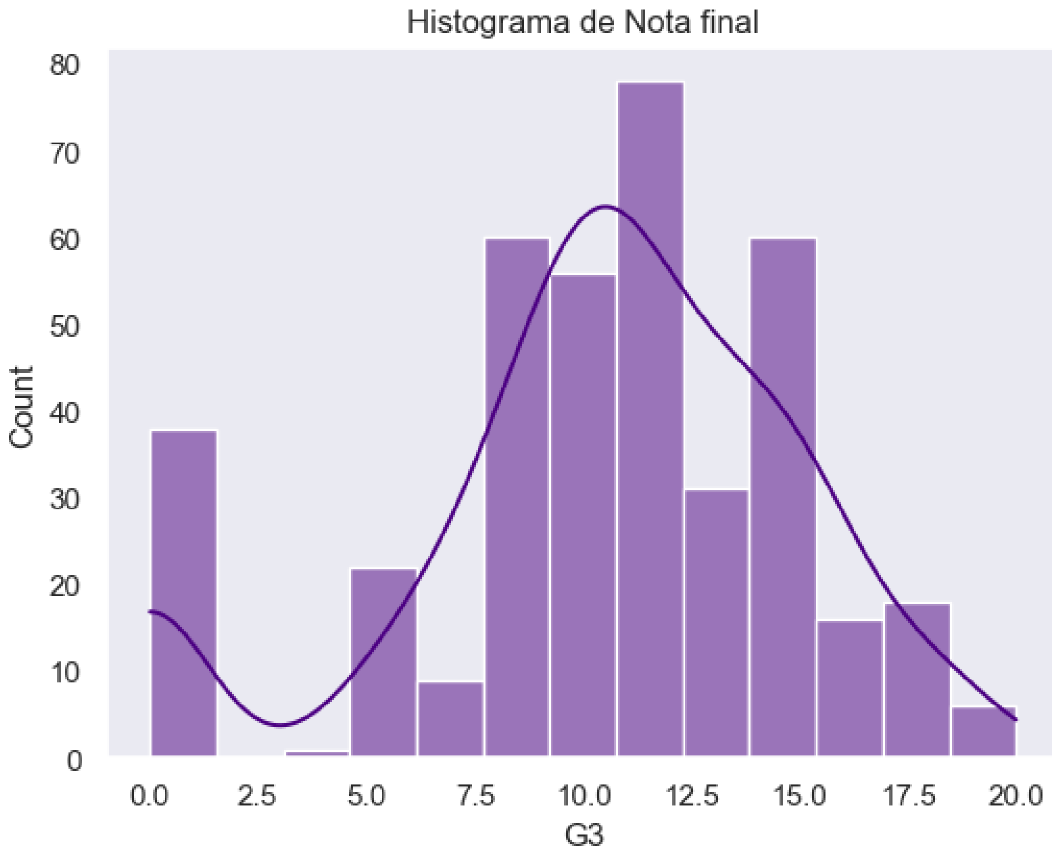

```
Nota del primer período
numérica: de 0 a 20
```



Nota del segundo período
numérica: de 0 a 20




```
Nota final
objetivo de salida, numérica: de 0 a 20
```



En base a los histogramas anteriores, podemos sacar las siguientes conclusiones:

1 - Edad del estudiante.

Los estudiantes están concentrados en la franja de edad de 15 a 18 años, ambos inclusive. Hay una presencia relativamente baja de estudiantes de 19 años, y apenas no hay de los de 20 - 22 años.

2 - Educación padres.

Se puede ver que las madres tienen en términos generales más nivel educativo que los padres (hombres), con mayor prevalencia de estudios universitarios.

En cuanto a la naturaleza del propio dataset, se puede decir que éste otorga una representividad un tanto desproporcionada a los estudios primarios.

Esto quiere decir que, en realidad, los padres (tanto mujeres como hombres) que poseen estudios universitarios son en realidad muy pocos.

3 - Tiempo de estudio semanal.

Es generalmente muy bajo, siendo la mayor frecuencia la de 2 horas de estudio semanal y tendiendo a menos (1 hora) en bastante casos. El número de alumnos que dedica 3 horas es infrecuente, y apenas hay alumnos que dediquen 4 horas.

4 - Número de fracasos.

A pesar de las estadísticas anteriores, podemos ver que el número de fracasos es casi inexistente.

5 - Consumo de alcohol.

Como hemos visto anteriormente en el análisis de correlaciones, el consumo de alcohol es bastante bajo. Aunque es más habitual que los estudiantes lo consuman los fines de semana.

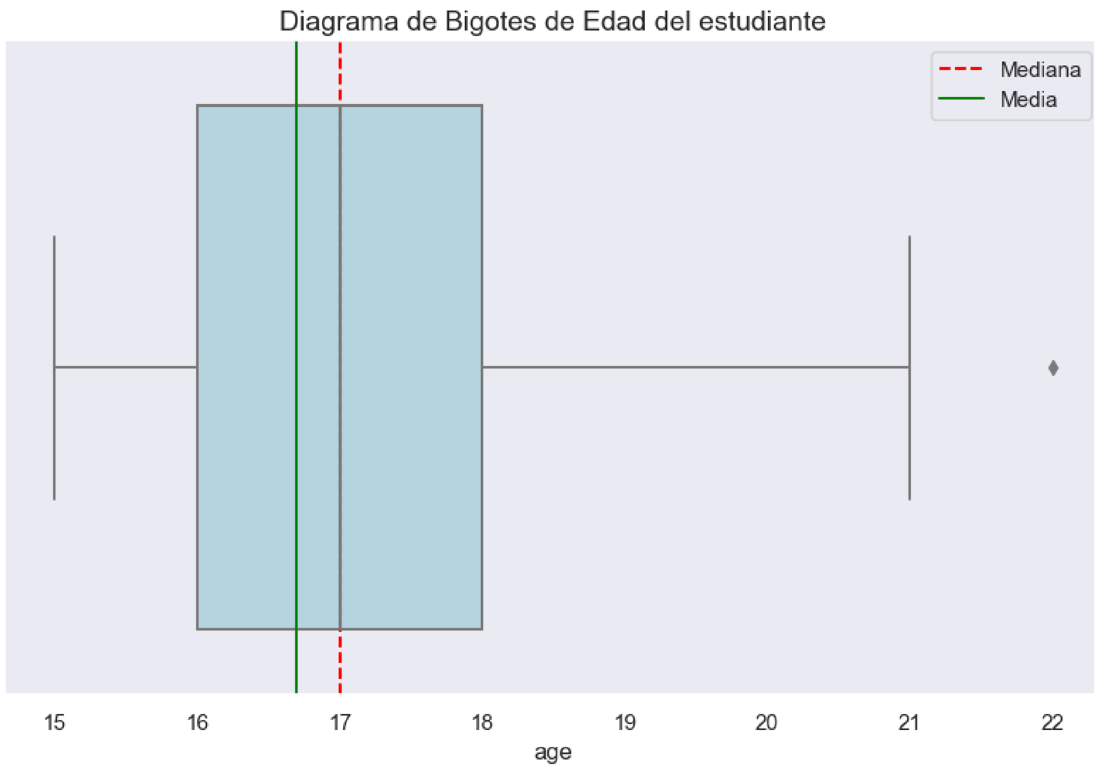
6 - Notas.

Por lo general, se observa una distribución que tiende a la normal en los 3 casos. Si bien hay una frecuencia relativamente importante de ceros en G2 (Segundo Período). Podemos inferir que este factor incide sobre la Nota Final, puesto que existe una cantidad relativamente importante de ceros también en la frecuencia de esta distribución.

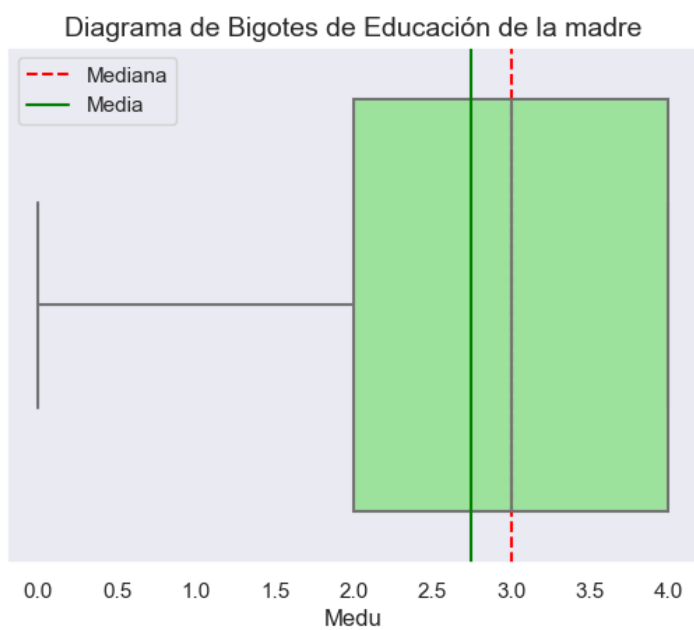
Podremos ver detalles como los de las Notas con mayor precisión si utilizamos el Diagrama de Bigotes. Aparte de mostrar la media y la mediana, esto nos servirá también para identificar outliers de cara al proceso de limpiado de datos:

[13]:
```python
light_colors = ['lightblue', 'lightgreen', 'salmon', 'lavender']

plt.figure(figsize=(10, 6))

skip_keys = ["traveltime", "famrel", "freetime", "goout", "health", "absences"]

for i, n in enumerate(nums_values):
    if n in skip_keys:
        continue

    sns.set_theme(style='dark')
    boxplot = sns.boxplot(x=nums_values[n], color=light_colors[i%4])

    plt.title(f'Diagrama de Bigotes de {titles[n][0]}', fontsize=14)
    plt.suptitle(titles[n][1], fontsize=10)

    median_val = nums_values[n].median()
    mean_val = nums_values[n].mean()
    plt.axvline(median_val, color='red', linestyle='--', label='Mediana')
    plt.axvline(mean_val, color='green', linestyle='-', label='Media')

    plt.legend()
    plt.show()
```

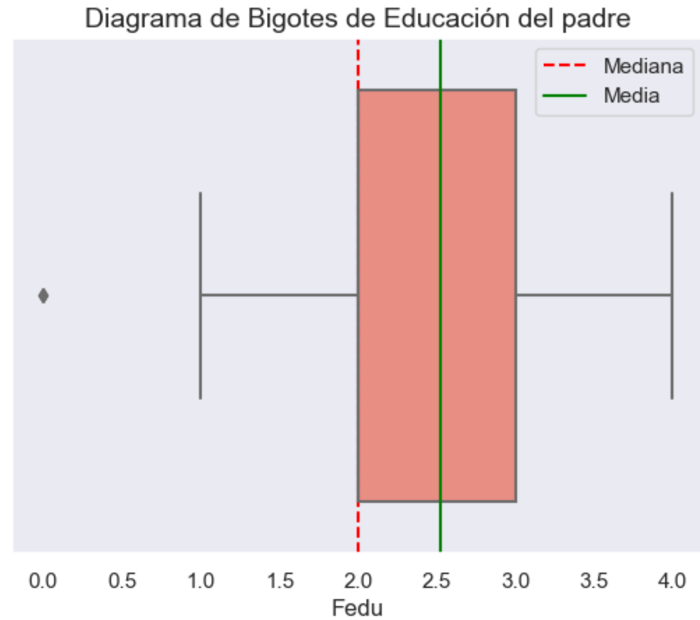
numérica: de 15 a 22



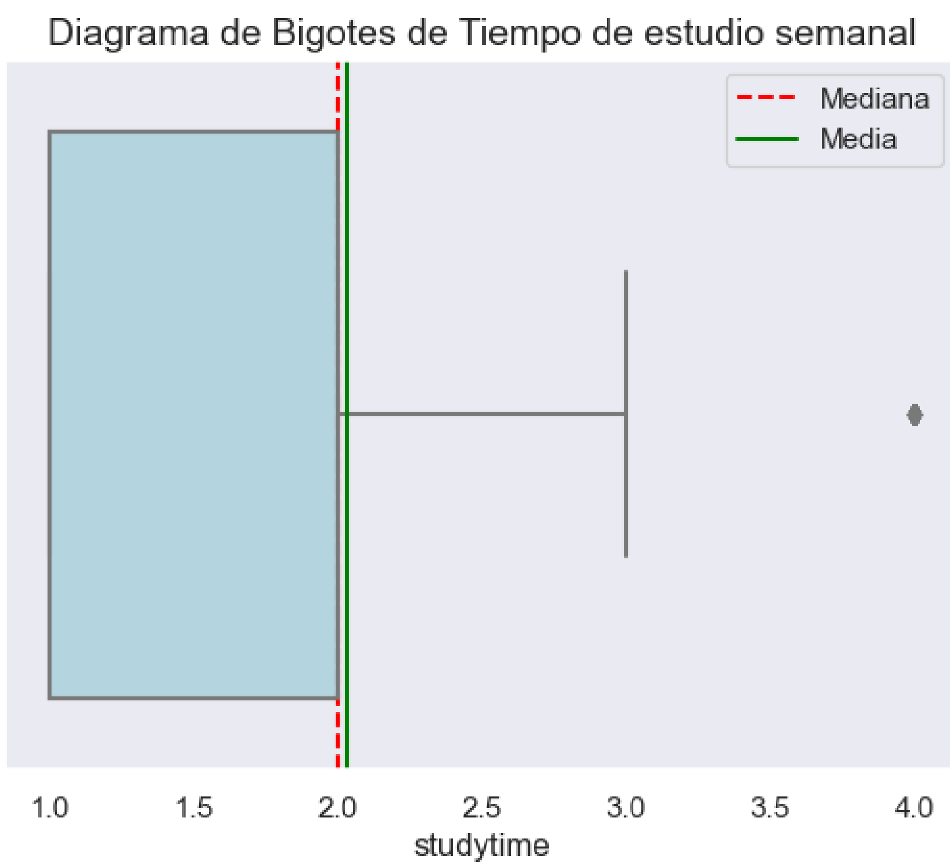
numérica: 0 - ninguna, 1 - educación primaria (4to grado), 2 - 5to a 9no grado, 3 - educación secundaria, 4 - educación superior



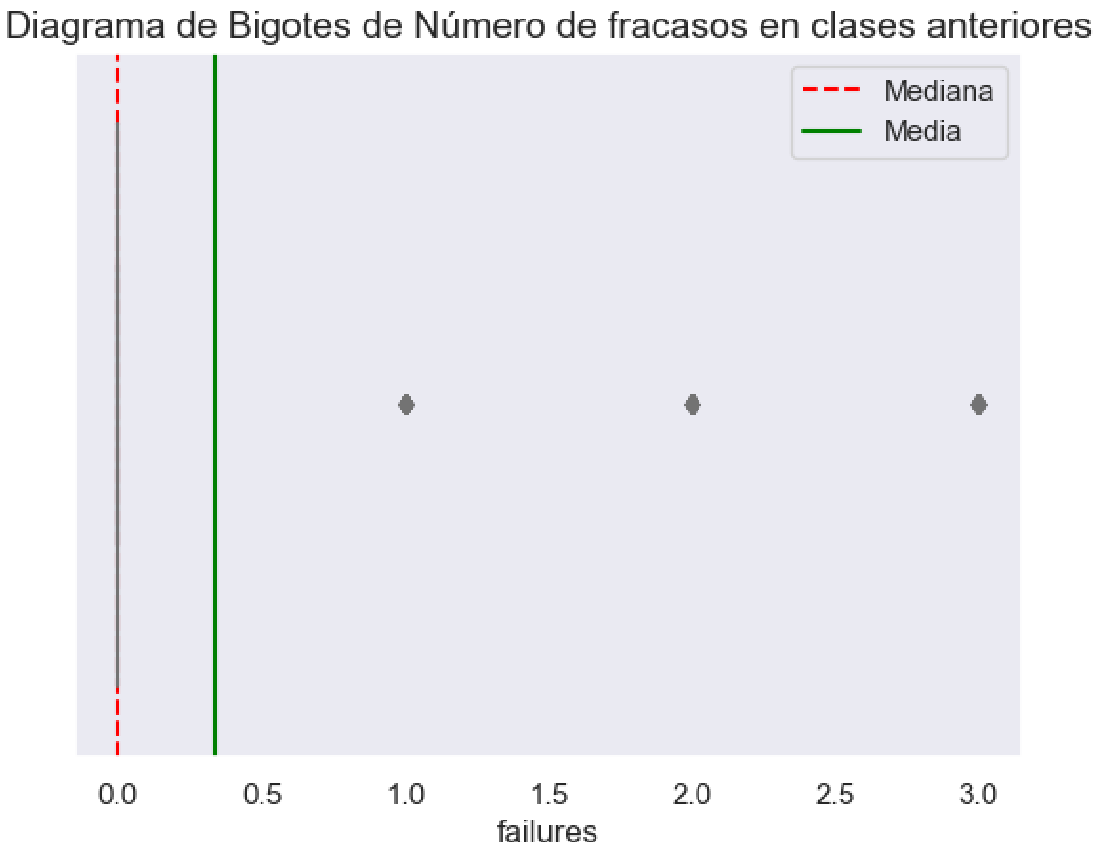
numérica: 0 - ninguna, 1 - educación primaria (4to grado), 2 - 5to a 9no grado, 3 - educación secundaria, 4 - educación superior



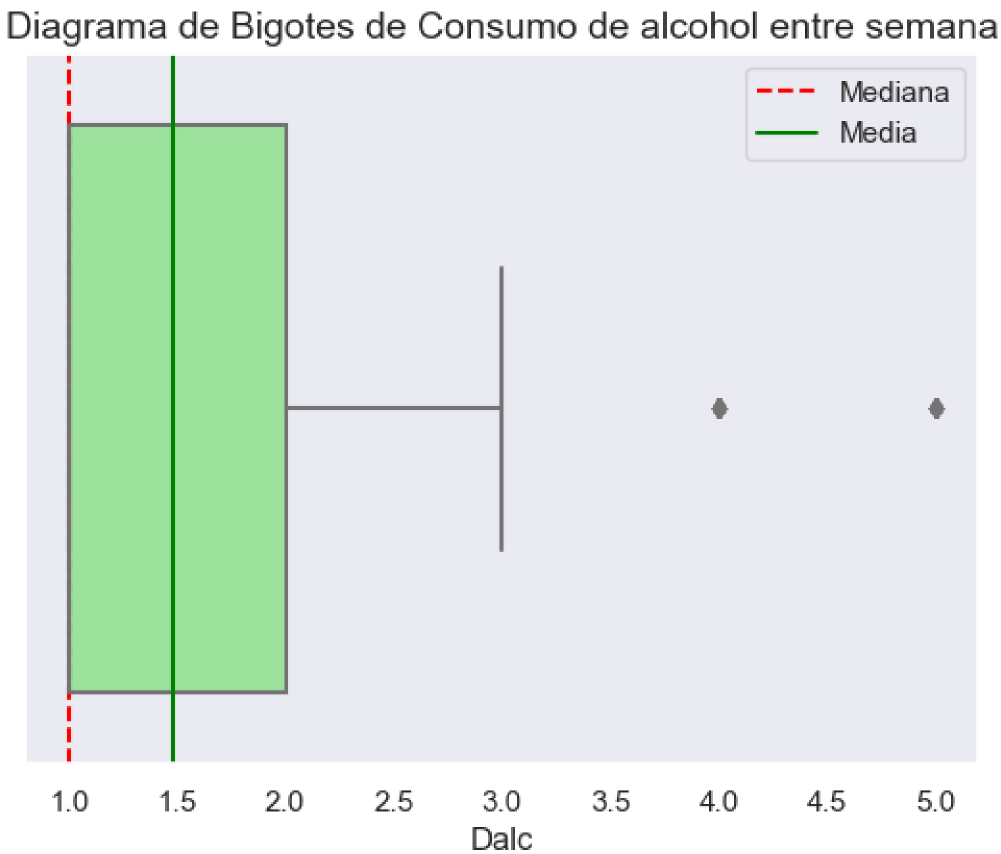
numérica: 1 - <2 horas, 2 - 2 a 5 horas, 3 - 5 a 10 horas, 4 - >10 horas

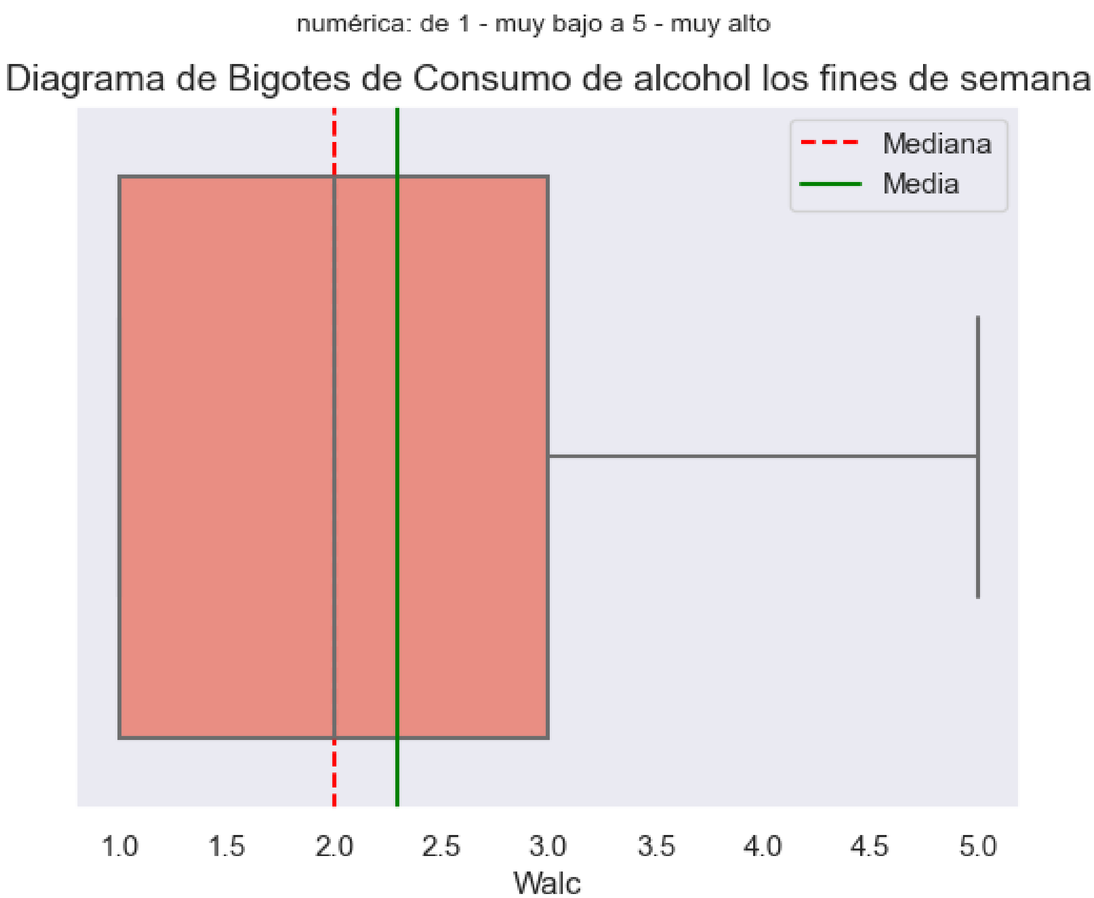


numérica: n si 1<=n<3, de lo contrario 4

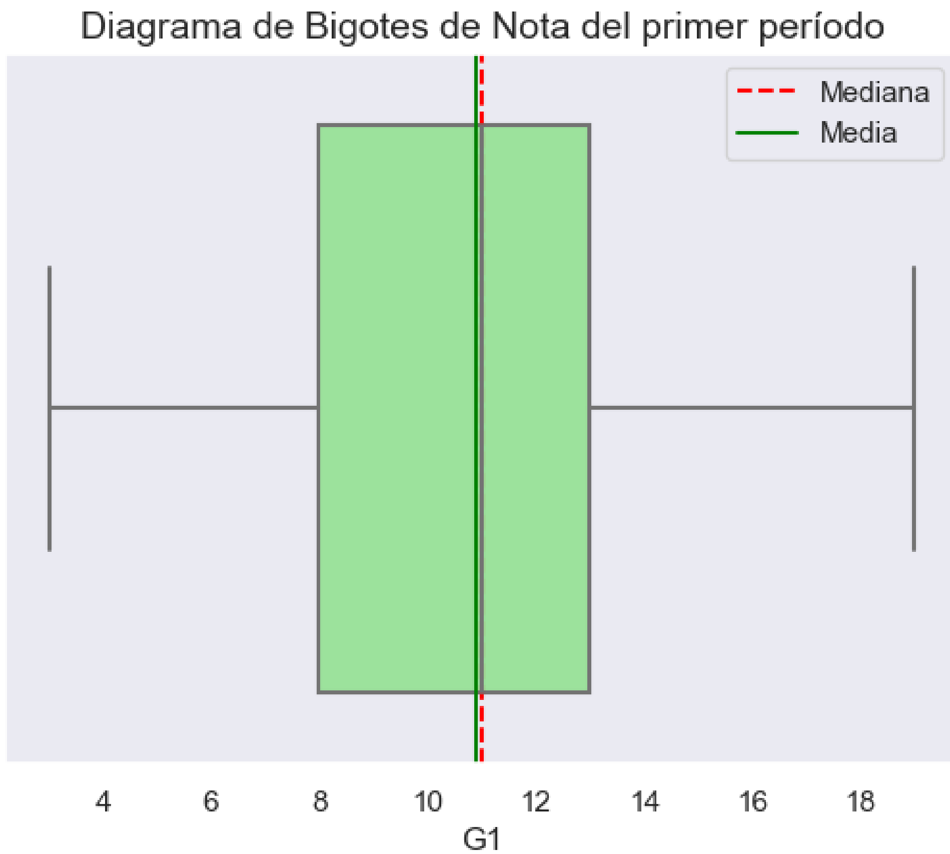


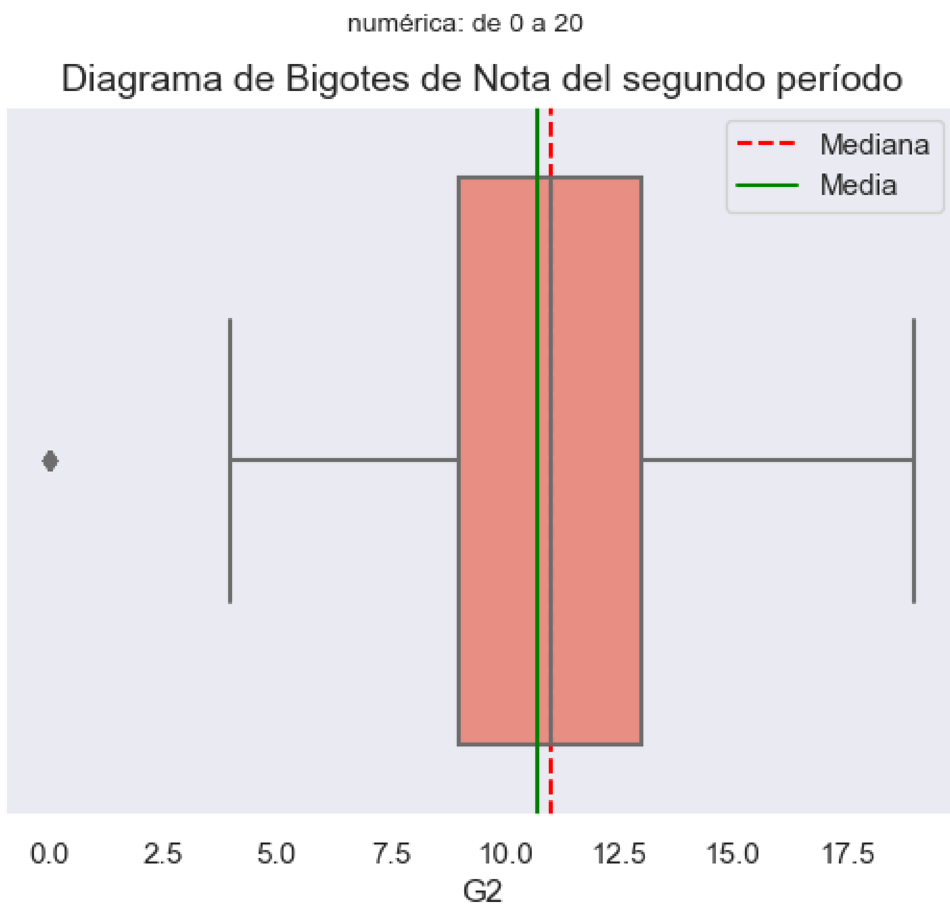
numérica: de 1 - muy bajo a 5 - muy alto



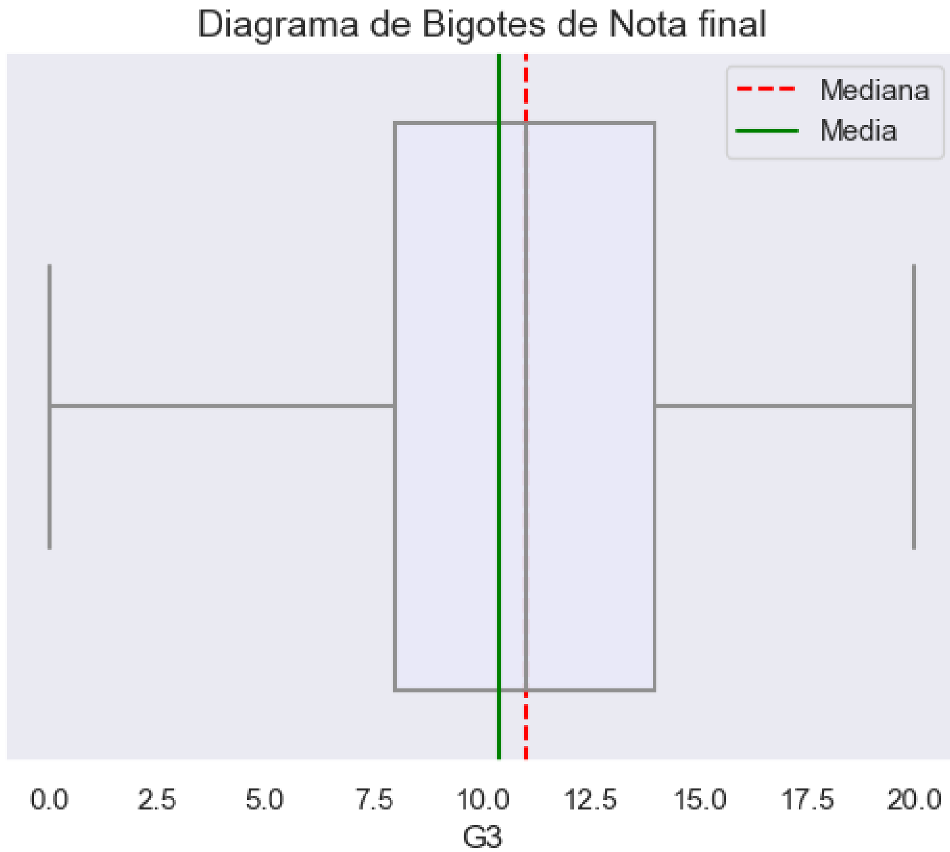

numérica: de 1 - muy bajo a 5 - muy alto
Diagrama de Bigotes de Consumo de alcohol los fines de semana
Mediana
Media
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Walc

numérica: de 0 a 20




numérica: de 0 a 20
Diagrama de Bigotes de Nota del segundo período
Mediana
Media
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
G2

objetivo de salida, numérica: de 0 a 20



Estos diagramas nos confirman, con aun mayor precisión, lo descrito previamente.

Data Pre-Processing

En primer lugar, comprobamos si es necesario imputar valores:

[14]:
```
missing_values_count = dataset.isnull().sum()
print("Missing values per column:")
print(missing_values_count)

total_missing_values = missing_values_count.sum()
print(f"Total missing values in the dataset: {total_missing_values}")

missing_percentage = (missing_values_count / len(dataset)) * 100
print("Percentage of missing values per column:")
print(missing_percentage)
```

```
Missing values per column:
school        0
sex           0
```

```
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
Fjob          0
reason        0
guardian      0
traveltime    0
studytime     0
failures      0
schoolsup     0
famsup        0
paid          0
activities    0
nursery       0
higher        0
internet      0
romantic      0
famrel        0
freetime      0
goout         0
Dalc          0
Walc          0
health        0
absences      0
G1            0
G2            0
G3            0
dtype: int64
Total missing values in the dataset: 0
Percentage of missing values per column:
school        0.0
sex           0.0
age           0.0
address       0.0
famsize       0.0
Pstatus       0.0
Medu          0.0
Fedu          0.0
Mjob          0.0
Fjob          0.0
reason        0.0
guardian      0.0
traveltime    0.0
studytime     0.0
```

```
failures      0.0
schoolsup     0.0
famsup        0.0
paid          0.0
activities    0.0
nursery       0.0
higher        0.0
internet      0.0
romantic      0.0
famrel        0.0
freetime      0.0
goout         0.0
Dalc          0.0
Walc          0.0
health        0.0
absences      0.0
G1            0.0
G2            0.0
G3            0.0
dtype: float64
```

Vemos que no es necesario imputar valores.

Para facilitar nuestro trabajo, podemos hacer uso de una stepwise regression, que consiste en combinar el enfoque forward con el enfoque backward a efectos de seleccionar las variables más relevantes para explicar nuestra variable objetivo (G3, es decir, Nota Final).

[15]:
```python
formula = 'G3 ~ ' + ' + '.join(dataset.drop(columns='G3').columns)
y, X = patsy.dmatrices(formula, data=dataset, return_type='dataframe')

model = sm.OLS(y, X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                     G3   R-squared:                       0.846
Model:                            OLS   Adj. R-squared:                  0.828
Method:                 Least Squares   F-statistic:                     47.21
Date:                Sat, 26 Oct 2024   Prob (F-statistic):          7.20e-119
Time:                        14:42:19   Log-Likelihood:                -791.99
No. Observations:                 395   AIC:                             1668.
Df Residuals:                     353   BIC:                             1835.
Df Model:                          41
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
```

```
--------
Intercept             -1.1155      2.117     -0.527      0.599      -5.279
3.048
school[T.MS]           0.4807      0.367      1.312      0.190      -0.240
1.202
sex[T.M]               0.1744      0.234      0.747      0.456      -0.285
0.634
address[T.U]           0.1045      0.271      0.386      0.700      -0.428
0.637
famsize[T.LE3]         0.0365      0.227      0.161      0.872      -0.409
0.482
Pstatus[T.T]          -0.1277      0.336     -0.380      0.704      -0.788
0.532
Mjob[T.health]        -0.1464      0.518     -0.282      0.778      -1.166
0.873
Mjob[T.other]          0.0741      0.332      0.223      0.824      -0.579
0.727
Mjob[T.services]       0.0470      0.370      0.127      0.899      -0.680
0.774
Mjob[T.teacher]       -0.0263      0.482     -0.055      0.957      -0.974
0.921
Fjob[T.health]         0.3309      0.667      0.496      0.620      -0.980
1.642
Fjob[T.other]         -0.0836      0.477     -0.175      0.861      -1.021
0.854
Fjob[T.services]      -0.3221      0.493     -0.653      0.514      -1.292
0.648
Fjob[T.teacher]       -0.1124      0.601     -0.187      0.852      -1.295
1.071
reason[T.home]        -0.2092      0.256     -0.816      0.415      -0.713
0.295
reason[T.other]        0.3076      0.380      0.809      0.419      -0.440
1.055
reason[T.reputation]   0.1291      0.267      0.483      0.629      -0.397
0.655
guardian[T.mother]     0.1957      0.253      0.775      0.439      -0.301
0.693
guardian[T.other]      0.0066      0.464      0.014      0.989      -0.905
0.918
schoolsup[T.yes]       0.4564      0.320      1.428      0.154      -0.172
1.085
famsup[T.yes]          0.1769      0.224      0.789      0.431      -0.264
0.618
paid[T.yes]            0.0758      0.222      0.341      0.733      -0.361
0.513
activities[T.yes]     -0.3460      0.206     -1.680      0.094      -0.751
0.059
nursery[T.yes]        -0.2227      0.254     -0.876      0.382      -0.723
```

```
0.277
higher[T.yes]          0.2259      0.500      0.451      0.652      -0.758
1.210
internet[T.yes]       -0.1445      0.288     -0.502      0.616      -0.710
0.421
romantic[T.yes]       -0.2720      0.220     -1.238      0.217      -0.704
0.160
age                   -0.1733      0.101     -1.720      0.086      -0.372
0.025
Medu                   0.1297      0.150      0.865      0.388      -0.165
0.425
Fedu                  -0.1339      0.129     -1.040      0.299      -0.387
0.119
traveltime             0.0970      0.158      0.615      0.539      -0.213
0.407
studytime             -0.1048      0.135     -0.777      0.438      -0.370
0.160
failures              -0.1605      0.161     -0.997      0.319      -0.477
0.156
famrel                 0.3569      0.114      3.127      0.002       0.132
0.581
freetime               0.0470      0.110      0.426      0.670      -0.170
0.264
goout                  0.0120      0.105      0.114      0.909      -0.195
0.219
Dalc                  -0.1850      0.153     -1.208      0.228      -0.486
0.116
Walc                   0.1768      0.115      1.538      0.125      -0.049
0.403
health                 0.0630      0.075      0.842      0.400      -0.084
0.210
absences               0.0459      0.013      3.421      0.001       0.020
0.072
G1                     0.1888      0.062      3.028      0.003       0.066
0.312
G2                     0.9573      0.053     17.907      0.000       0.852
1.062
==============================================================================
Omnibus:                      172.835   Durbin-Watson:                   1.925
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              735.778
Skew:                          -1.923   Prob(JB):                    1.69e-160
Kurtosis:                       8.470   Cond. No.                         569.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

De esta forma, podemos observar que, atendiendo a los P-Valores, las variables que inciden claramente en G3 (Nota Final y nuestra Variable Objetivo), son:

G2 (Nota Segundo Período) G1 (Nota Primer Período) absences (Ausencias) famrel (Calidad de Relaciones Familiares)

Adicionalmente, dado que los P-Valores son bastante limítrofes, podemos incluir: Walc (Consumo de Alcohol los Fines de Semana) age (Edad) activities (Actividades)

Debido a las posibles relaciones no lineales, puede ser conveniente aplicar también un Principal Component Analysis.

Si bien Principal Component Analysis es una de las principales técnicas en Unsupervised Machine Learning, también puede ser una muy buena herramienta a efectos de data pre-processing.

Así, podemos aplicar una serie de transformaciones (ajuste de escala para variables numéricas o encoding para variables categóricas), comprobar que los resultados son consistentes con respecto a los obtenidos mediante la stepwise regression, y finalmente guardar el output en un archivo nuevo, quedándonos sólo con las variables más relevantes.

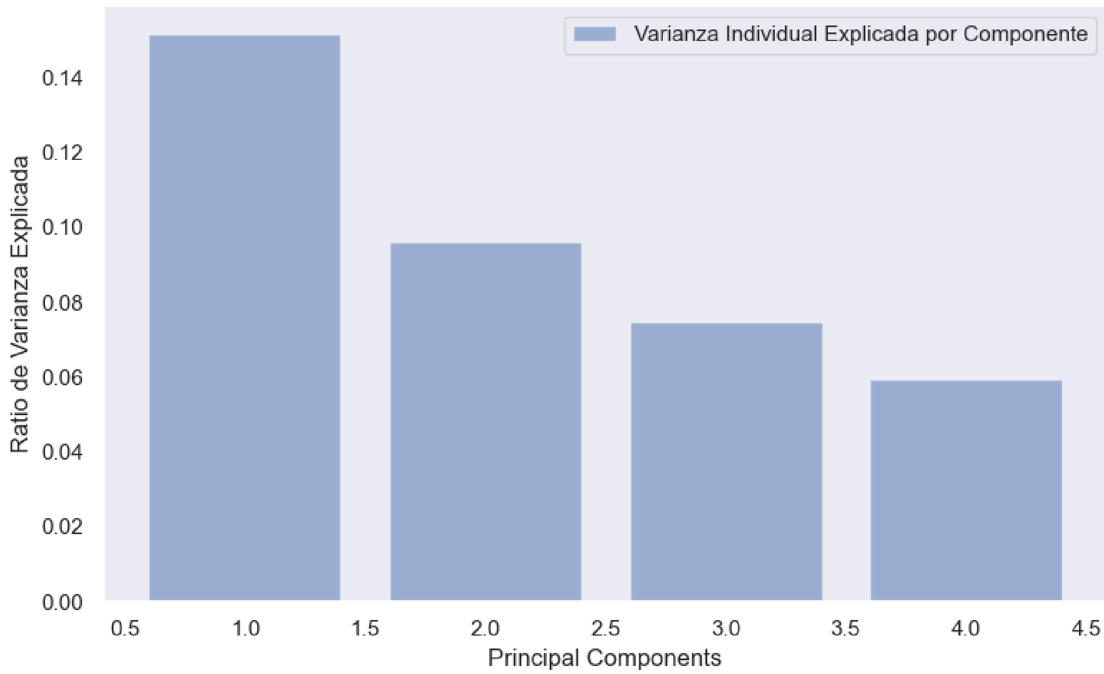
[16]:
```
random.seed(42)

#Identificamos las respectivas variables numéricas y categóricas
numeric_features = dataset.select_dtypes(include=['int64', 'float64']).columns
categorical_features = dataset.select_dtypes(include=['object', 'category']).
 ↪columns

#Ajustamos escala (variables numéricas) y hacemos encoding (variables␣
 ↪categóricas)
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

#Ajuste y transformación
data_processed = preprocessor.fit_transform(dataset)

#Aplicamos Principal Component Analysis a 4 componentes
pca = PCA(n_components=4)
principal_components = pca.fit_transform(data_processed)

#Creamos DataFrame para los componentes
principal_df = pd.DataFrame(data=principal_components, columns=['PC 1', 'PC 2',␣
 ↪'PC3', 'PC4'])

#Nombres de los componentes
features_after_encoding = (preprocessor.named_transformers_['num'].
 ↪get_feature_names_out(numeric_features).tolist() +
```

```python
                              preprocessor.named_transformers_['cat'].
 ↪get_feature_names_out().tolist())

#Features procesados
loadings = pd.DataFrame(data=pca.components_.T, columns=['PC 1', 'PC 2', 'PC3',␣
 ↪'PC4'], index=features_after_encoding)

#Visualización de los componentes
print(principal_df)

#Visualización features procesados
print("PCA Component Loadings:")
print(loadings)

#Ratio de Varianza Explicada
print("Ratio de Varianza Explicada:")
print(pca.explained_variance_ratio_)

#Visualización
plt.figure(figsize=(8, 5))
plt.bar(range(1, len(pca.explained_variance_ratio_) + 1), pca.
 ↪explained_variance_ratio_, alpha=0.5, align='center', label='Varianza␣
 ↪Individual Explicada por Componente')
plt.ylabel('Ratio de Varianza Explicada')
plt.xlabel('Principal Components')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

```
         PC 1      PC 2       PC3       PC4
0    1.253363 -0.924006 -2.072979  0.464239
1    2.162052 -2.434154  0.224991 -0.835760
2    2.383158 -0.846343  1.115137  0.226220
3   -2.792521 -1.292548 -0.449818  0.338404
4    0.057999 -1.204821 -1.147271 -0.522616
..        …         …         …         …
390  3.051158  3.341803  1.172530  0.442626
391 -0.192146  3.269674  2.422541  0.629132
392  4.122480  1.242689  3.022775 -0.361219
393  1.074481  1.785756  2.064965 -0.811824
394  2.194200  0.581688  1.948832  0.538610

[395 rows x 4 columns]
PCA Component Loadings:
                          PC 1      PC 2       PC3       PC4
age                   0.172971  0.061271  0.231519  0.348060
Medu                 -0.245785  0.197877 -0.496146  0.112037
```

Fedu	-0.219519	0.177696	-0.500512	0.049008
traveltime	0.152532	0.042419	0.217797	0.013056
studytime	-0.156254	-0.239001	-0.042124	0.160902
failures	0.315242	0.015072	0.089879	0.044102
famrel	-0.018982	-0.013428	0.001725	-0.377349
freetime	0.060120	0.293395	-0.001906	-0.403790
goout	0.152728	0.347715	-0.131995	-0.029590
Dalc	0.160369	0.486831	0.052351	0.101398
Walc	0.183907	0.502815	0.063586	0.108350
health	0.069323	0.075187	-0.048056	-0.381833
absences	0.033762	0.130338	-0.043048	0.507893
G1	-0.435852	0.142716	0.286868	0.012442
G2	-0.454184	0.160686	0.287820	0.019703
G3	-0.439103	0.183560	0.283017	-0.003253
school_GP	-0.028886	-0.012792	-0.064347	-0.029788
school_MS	0.028886	0.012792	0.064347	0.029788
sex_F	-0.007001	-0.153594	-0.038817	0.131233
sex_M	0.007001	0.153594	0.038817	-0.131233
address_R	0.042943	-0.001933	0.054766	0.027032
address_U	-0.042943	0.001933	-0.054766	-0.027032
famsize_GT3	0.003099	-0.045102	-0.055215	-0.017586
famsize_LE3	-0.003099	0.045102	0.055215	0.017586
Pstatus_A	-0.009247	0.011374	-0.015947	0.030876
Pstatus_T	0.009247	-0.011374	0.015947	-0.030876
Mjob_at_home	0.032815	-0.042487	0.064230	0.013607
Mjob_health	-0.025737	0.015312	-0.029731	0.000811
Mjob_other	0.044067	-0.037422	0.055451	-0.000780
Mjob_services	-0.014638	0.021709	-0.002525	-0.014390
Mjob_teacher	-0.036506	0.042888	-0.087426	0.000754
Fjob_at_home	-0.001137	-0.010598	0.007568	0.001563
Fjob_health	-0.012920	-0.005205	-0.023217	0.000902
Fjob_other	0.034874	-0.022283	0.050281	-0.027167
Fjob_services	0.005921	0.021908	0.005033	0.024942
Fjob_teacher	-0.026738	0.016178	-0.039665	-0.000240
reason_course	0.034639	0.000172	0.015311	-0.107044
reason_home	0.003241	0.002618	-0.006243	0.044292
reason_other	0.002072	0.024440	0.011153	0.005224
reason_reputation	-0.039952	-0.027230	-0.020221	0.057528
guardian_father	-0.012823	0.003174	0.002249	-0.039277
guardian_mother	-0.012250	0.001424	-0.034530	-0.004816
guardian_other	0.025073	-0.004599	0.032281	0.044093
schoolsup_no	-0.009361	0.033456	0.047110	0.014775
schoolsup_yes	0.009361	-0.033456	-0.047110	-0.014775
famsup_no	0.019613	0.022890	0.136627	-0.020142
famsup_yes	-0.019613	-0.022890	-0.136627	0.020142
paid_no	0.046951	-0.017895	0.065352	-0.079871
paid_yes	-0.046951	0.017895	-0.065352	0.079871
activities_no	0.029160	-0.011412	0.055116	0.054747

```
activities_yes    -0.029160  0.011412 -0.055116 -0.054747
nursery_no         0.036267  0.004175  0.059954 -0.013081
nursery_yes       -0.036267 -0.004175 -0.059954  0.013081
higher_no          0.036094  0.005903  0.022902  0.004283
higher_yes        -0.036094 -0.005903 -0.022902 -0.004283
internet_no        0.032493 -0.032680  0.052475 -0.020553
internet_yes      -0.032493  0.032680 -0.052475  0.020553
romantic_no       -0.026580  0.005928  0.025423 -0.099497
romantic_yes       0.026580 -0.005928 -0.025423  0.099497
Ratio de Varianza Explicada:
[0.15172055 0.09598337 0.07453343 0.05929673]
```



La selección del número de 4 Componentes Principales parece razonable en tanto que explicamos una parte importante del dataset y, como podemos ver, la contribución de los componentes siguientes comienza a ser marginal.

Vemos que los resultados obtenidos concuerdan bastante con el análisis previo. Así, las variables más importantes son, entre otras:

G3, G2, G1 (Notas) age (Edad) Medu y Fedu (Nivel Educativo Padres) failures (Fracasos) Dalc y Walc (Consumo de Alcohol)

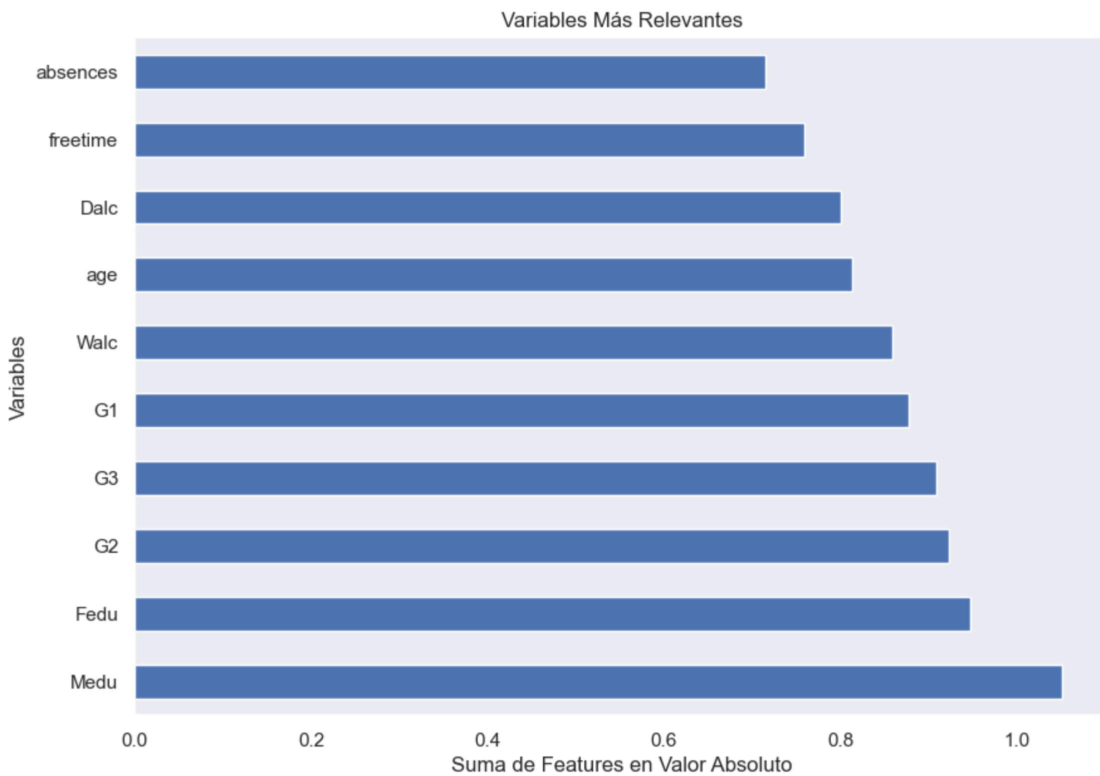
A continuación, vamos a realizar unas pocas permutaciones adicionales y guardar las variables más relevantes en un documento CSV con el que trabajaremos posteriormente.

[17]:
```
random.seed(42)
```

```
numeric_features = dataset.select_dtypes(include=['int64', 'float64']).columns
categorical_features = dataset.select_dtypes(include=['object', 'category']).
 ↪columns

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

data_processed = preprocessor.fit_transform(dataset)
features_after_encoding = preprocessor.transformers_[0][1].
 ↪get_feature_names_out(numeric_features).tolist() + \
                          preprocessor.transformers_[1][1].
 ↪get_feature_names_out().tolist()

pca = PCA(n_components=4)
pca.fit(data_processed)

loadings = pd.DataFrame(data=pca.components_.T, columns=[f'PC{i+1}' for i in␣
 ↪range(4)], index=features_after_encoding)

loadings['Suma de Features en Valor Absoluto'] = loadings.abs().sum(axis=1)

top_variables = loadings.nlargest(10, 'Suma de Features en Valor Absoluto').
 ↪index.tolist()

#Decodificación de variables
original_feature_names = [name.split('_')[0] for name in top_variables if '_'␣
 ↪in name] + \
                         [name for name in top_variables if '_' not in name]

top_features_data = dataset[original_feature_names]


top_features_data.to_csv('dataset_cleaned.csv', index=False)

print(top_features_data.head())

plt.figure(figsize=(10, 7))
loadings['Suma de Features en Valor Absoluto'].nlargest(10).plot(kind='barh')
plt.title('Variables Más Relevantes')
plt.xlabel('Suma de Features en Valor Absoluto')
plt.ylabel('Variables')
plt.show()
```

```
Medu  Fedu  G2  G3  G1  Walc  age  Dalc  freetime  absences
```

0	4	4	6	6	5	1	18	1	3	6
1	1	1	5	6	5	1	17	1	3	4
2	1	1	8	10	7	3	15	2	3	10
3	4	2	14	15	15	1	15	1	2	2
4	3	3	10	10	6	2	16	1	3	4



Podemos plantear el siguiente Sistema Experto:

Ante todo, tenemos que tener en cuenta que hay una serie de variables estructurales en las que no podemos influir.

Estas son:

Medu (Nivel educativo de la madre) Fedu (Nivel educativo del padre) age (Edad)

Tampoco podemos influir de forma directa ni en G1 (Primer Período) ni en G2 (Segundo Período).

Por otra parte, nuestra variable objetivo es G3 (Nota Final).

Los factores que inciden en ella y en los que sí podemos influir son:

absences (ausencias: se trataría de reducir su número) Dalc (consumo de alcohol entre semana)

Finalmente, los efectos de las variables freetime y Walc pueden ser un tanto ambiguos.

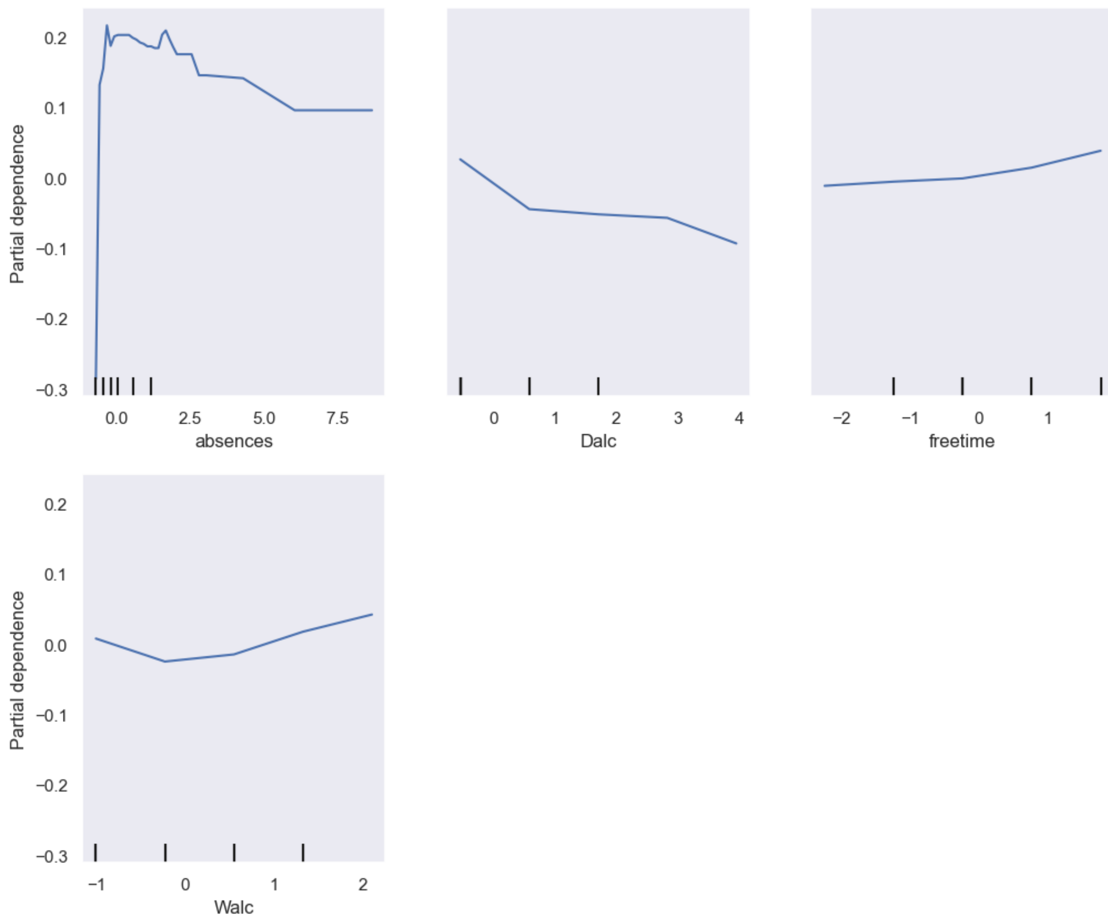
Con esto en mente, planteamos el modelo de Machine Learning.

Nos decantamos por Stocasting Gradient Boosting, dentro de Supervised Machine Learning.

[18]:
```python
random.seed(42)

dataset_cleaned = pd.read_csv('dataset_cleaned.csv')

#Encoding de variables categóricas
label_encoders = {}
for column in dataset_cleaned.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    dataset_cleaned[column] = le.fit_transform(dataset_cleaned[column])
    label_encoders[column] = le

#Ajuste de escala de las variables numéricas
scaler = StandardScaler()
numerical_cols = dataset_cleaned.select_dtypes(include=['int64', 'float64']).
 ↪columns
dataset_cleaned[numerical_cols] = scaler.
 ↪fit_transform(dataset_cleaned[numerical_cols])

#Separación del dataset en training y testing
X = dataset_cleaned.drop('G3', axis=1)
y = dataset_cleaned['G3']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

#Ajuste del modelo Stochastic Gradient Boosting
model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,␣
 ↪max_depth=3, subsample=0.8, random_state=42)
model.fit(X_train, y_train)

#Predicción y Evaluación
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

#Análisis de importancia de los features
feature_importance = model.feature_importances_
print("Feature Importances:")
for feature, importance in zip(X.columns, feature_importance):
    print(f"{feature}: {importance:.3f}")

#Primera interpretación del modelo con dependency plots
features = ['absences', 'Dalc', 'freetime', 'Walc']
fig, ax = plt.subplots(figsize=(12, 10))
PartialDependenceDisplay.from_estimator(model, X_train, features, ax=ax)
plt.show()
```

```
Mean Squared Error: 0.16196197412939672
```

```
Feature Importances:
Medu: 0.003
Fedu: 0.009
G2: 0.770
G1: 0.030
Walc: 0.003
age: 0.030
Dalc: 0.005
freetime: 0.007
absences: 0.143
```



Resultados:

Vemos que MSE es bastante bajo, lo que indica el modelo Stochastic Gradient es un buen modelo.

En cuanto a feature importances, tenemos que G2 (Segundo Período) es la variable predictiva más relevante (valor de 0.77), si bien no podemos influir en ella de forma directa. En cuanto a los features sobre los que sí podemos tener control, podemos destacar absences (ausencias, con un valor de 0.14). Por tanto, podríamos mejorar el rendimiento final si conseguimos tener menos ausencias, algo que concuerda con la expertise planteada al inicio.

Esto se confirma también en los dependency plots: podemos ver el relativamente fuerte impacto de las ausencias. Dalc (consumo de alcohol entre semana) tiene algo de influencia, pero muy poca en términos comparativos.

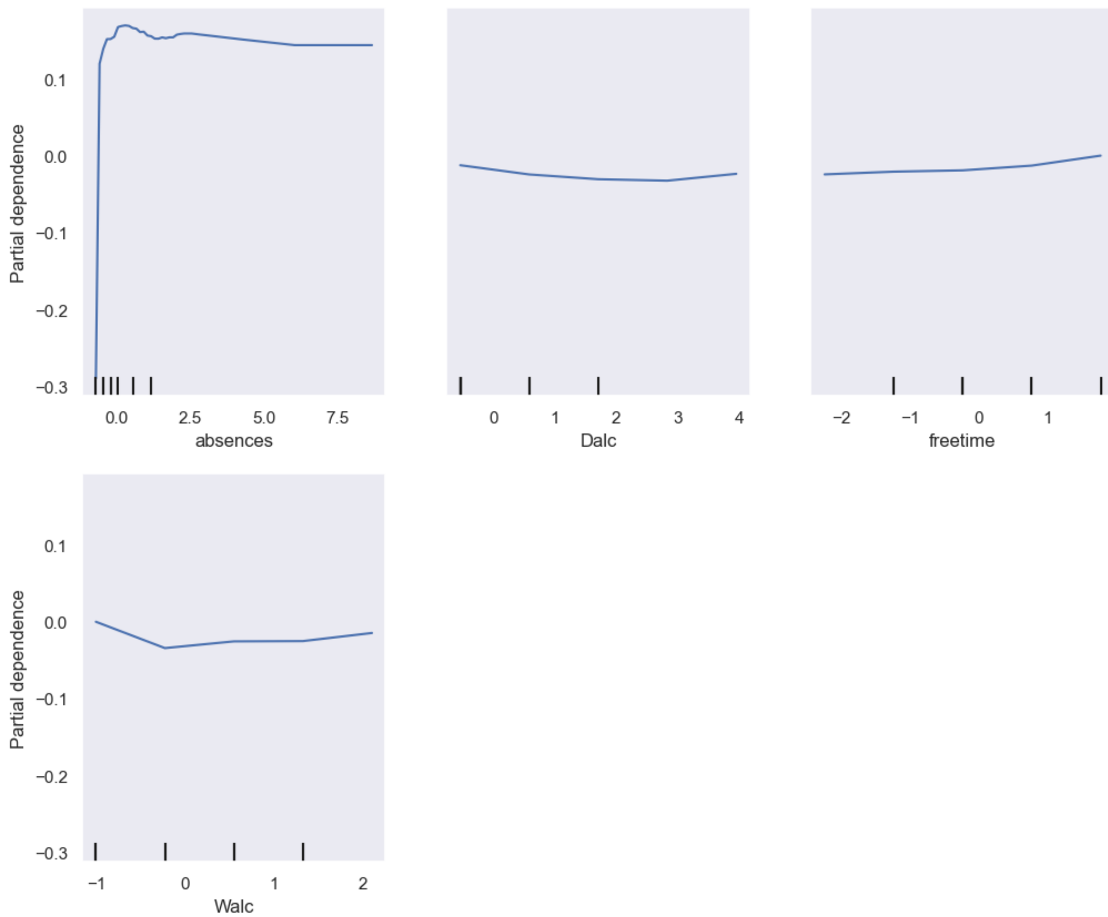
Podemos plantear otro modelo de Supervised Machine Learning a efectos de contrastar e intentar mejorar los resultados.

Teniendo en cuenta que no podemos plantear feature importances o dependency plots en caso de Support Vector Machines (debido a su estructura) y disponiendo de un dataset pequeño, podemos optar por Random Forest:

[19]:
```
random.seed(42)

dataset_cleaned = pd.read_csv('dataset_cleaned.csv')

label_encoders = {}
for column in dataset_cleaned.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    dataset_cleaned[column] = le.fit_transform(dataset_cleaned[column])
    label_encoders[column] = le

scaler = StandardScaler()
numerical_cols = dataset_cleaned.select_dtypes(include=['int64', 'float64']).
 ↪columns
dataset_cleaned[numerical_cols] = scaler.
 ↪fit_transform(dataset_cleaned[numerical_cols])

X = dataset_cleaned.drop('G3', axis=1)
y = dataset_cleaned['G3']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

#Ajuste del modelo Random Forest
model = RandomForestRegressor(n_estimators=100, max_depth=10, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

feature_importance = model.feature_importances_
print("Feature Importances:")
for feature, importance in zip(X.columns, feature_importance):
    print(f"{feature}: {importance:.3f}")

features = ['absences', 'Dalc', 'freetime', 'Walc']
fig, ax = plt.subplots(figsize=(12, 10))
PartialDependenceDisplay.from_estimator(model, X_train, features, ax=ax)
```

```
plt.show()
```

```
Mean Squared Error: 0.15273262437119808
Feature Importances:
Medu: 0.008
Fedu: 0.016
G2: 0.797
G1: 0.016
Walc: 0.008
age: 0.029
Dalc: 0.004
freetime: 0.008
absences: 0.114
```



Obtenemos unos resultados muy similares:

MSE es muy ligeramente más bajo, de 0.15, lo que nos indica que Random Forest es un modelo marginalmente mejor en su conjunto que Stochastic Gradient Boosting.

En cuanto a features, tenemos que Random Forest otorga aún más importancia a G2 (Segundo

Período, igual a 0.8). Con respecto a absences, la única variable significativa en la cual podemos influir a nivel de Sistema Experto, su relevancia es algo más baja aquí, de 0.11.

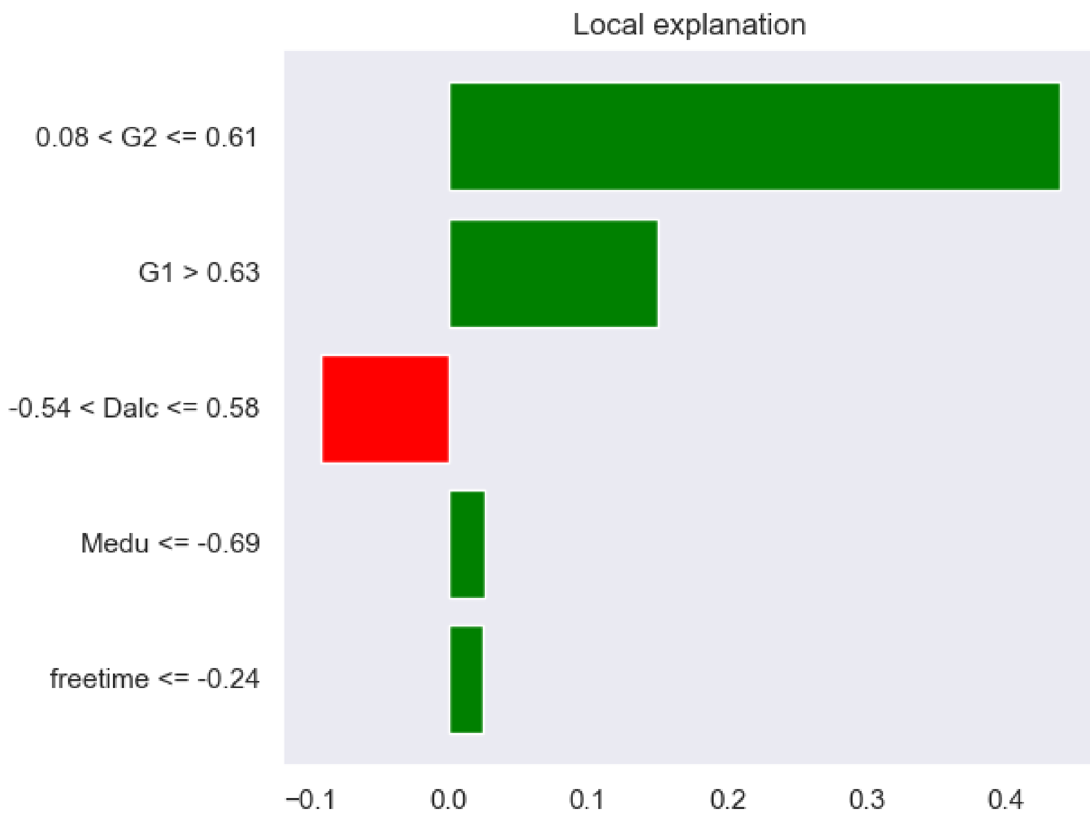
Tanto por el hecho de que Random Forest es un método usado demasiado frecuentemente como porque tenemos más margen de maniobra con el modelo anterior, nos vamos a quedar con Stochastic Gradient Boosting, es decir, el modelo anterior.

A continuación, vamos a plantear código para LIME y comentar este enfoque de Interpretable Machine Learning:

[20]:
```
random.seed(42)

dataset_cleaned = pd.read_csv('dataset_cleaned.csv')

label_encoders = {}
for column in dataset_cleaned.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    dataset_cleaned[column] = le.fit_transform(dataset_cleaned[column])
    label_encoders[column] = le

scaler = StandardScaler()
numerical_cols = dataset_cleaned.select_dtypes(include=['int64', 'float64']).
 ↪columns
dataset_cleaned[numerical_cols] = scaler.
 ↪fit_transform(dataset_cleaned[numerical_cols])

X = dataset_cleaned.drop('G3', axis=1)
y = dataset_cleaned['G3']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

#Como hemos dicho, nos quedamos con Stochastic Gradient Boosting
model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,␣
 ↪max_depth=3, subsample=0.8, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

#Definimos función para evitar un warning debido a que LIME permuta los datos␣
 ↪de forma interna
def model_predict(data_as_array):
    data_as_df = pd.DataFrame(data_as_array, columns=X_train.columns)
    return model.predict(data_as_df)

#Explicador LIME
explainer = LimeTabularExplainer(X_train.values,
```

```
                                    feature_names=X.columns,
                                    class_names=['G3'],
                                    mode='regression')

#Explicación de predicciones
i = 1
exp = explainer.explain_instance(X_test.iloc[i].values, model_predict,␣
 ↪num_features=5)

#Visualizacion de la explicación
fig = exp.as_pyplot_figure()
fig.tight_layout()
plt.show()
```

```
Mean Squared Error: 0.16196197412939672
```



En el caso de LIME, cabe tener en cuenta que es un enfoque post-hoc (la evaluación se realiza una vez que se tienen los resultados), agnóstico (se adapta a cualquier algoritmo de Machine Learning previo: en este caso, Stochastic Gradient Boosting) y local (no se generaliza a la totalidad de la distribución).

Para mayor adaptabilidad al lector, hemos fijado el parámetro de Explicación de Predicciones a 1.

Es decir, planteamos una única regresión. Si bien es perfectamente posible elaborar más regresiones, cada vez que quisieramos simular código nos aparecería un resultado diferente. Luego, no habría concordancia con la explicación que aquí se da.

Así pues, vemos que G2 (Segundo Período) y, en menor medida, G1 (Primer Período) y Medu (Nivel Educativo de la Madre) son los features que determinan un mejor resultado final (mayor G3, Nota Final).

Por otra parte, absences (ausencias) y Walc (consumo de alcohol los fines de semana) disminuyen el rendimiento final.

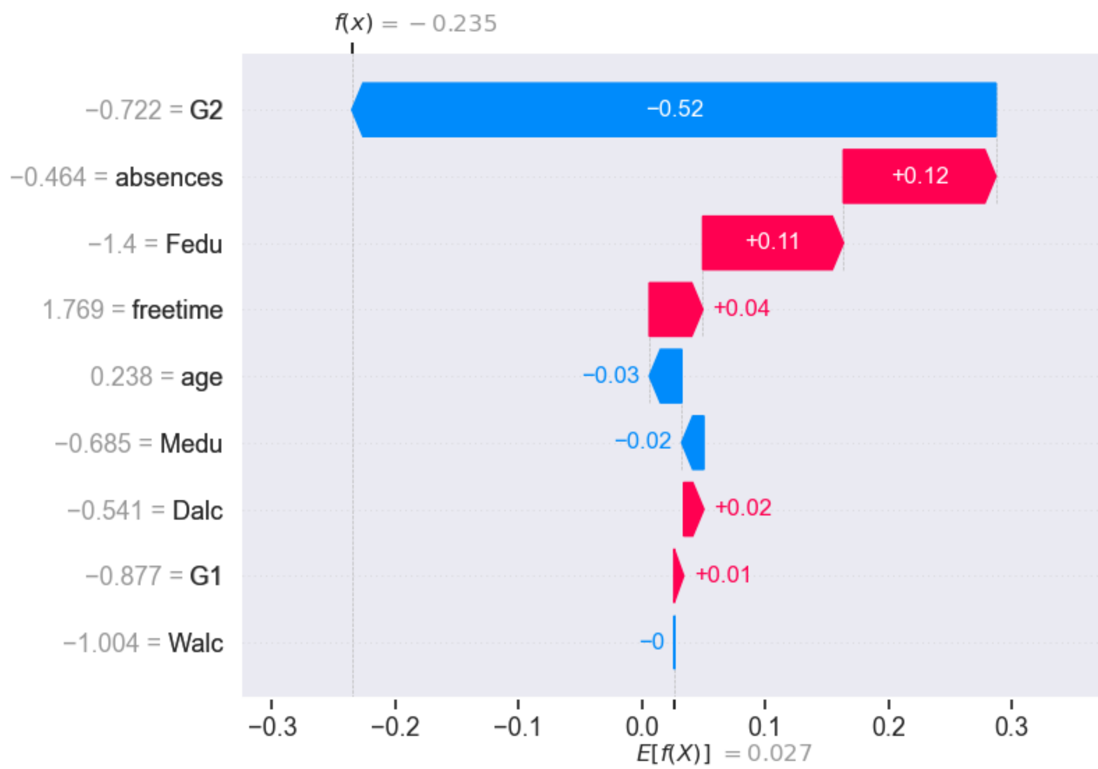
Esta interpretación nos permite ampliar la expertise inicial, donde no estábamos del todo seguros sobre el impacto de la variable Walc. Gracias al método LIME, podemos extender nuestra expertise y afirmar que no sólo influyen negativamente las ausencias y Dalc (consumo de alcohol entre semana), sino que también lo hace Walc (consumo de alcohol entre los fines de semana).

Como desventajas del LIME, podemos mencionar que se trata de una métrica basada en la distribución normal y, al ser una regresión, las relaciones que se plantean son lineales.

Procedemos a continuación con SHAP:

[21]:
```
random.seed(42)

dataset_cleaned = pd.read_csv('dataset_cleaned.csv')

label_encoders = {}
for column in dataset_cleaned.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    dataset_cleaned[column] = le.fit_transform(dataset_cleaned[column])
    label_encoders[column] = le

scaler = StandardScaler()
numerical_cols = dataset_cleaned.select_dtypes(include=['int64', 'float64']).
 ↪columns
dataset_cleaned[numerical_cols] = scaler.
 ↪fit_transform(dataset_cleaned[numerical_cols])

X = dataset_cleaned.drop('G3', axis=1)
y = dataset_cleaned['G3']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,␣
 ↪max_depth=3, subsample=0.8, random_state=42)
model.fit(X_train, y_train)

#Inicialización de SHAP
explainer = shap.Explainer(model, X_train)

#Cálculos de las Shapley Values
```

```python
shap_values = explainer(X_test, check_additivity=False)

#Visualización de las Shapley Values para la primera predicción
shap.plots.waterfall(shap_values[0])
```



SHAP es un método originado en base al trabajo de Lloyd Shapley en teoría de juegos cooperativos.

En contexto de Machine Learning, el juego es la tarea de predicción que hemos realizado (en este caso, con Stochastic Gradient Boosting), los jugadores son los features que hemos utilizado, y la matriz de pagos sería el resultado obtenido.

La lógica de SHAP es la siguiente:

1 - En primer lugar, simulamos todas las interacciones posibles entre las diferentes variables. La simulación es de carácter exponencial (es el mínimo de interacciones, es decir 2, elevado a n posibilidades). Evidentemente, tendríamos un problema a nivel computacional si la cantidad de variables fuese muy grande, si bien no es el caso en nuestro dataset.

2 - En segundo lugar, teniendo en cómputo todas las interacciones posibles entre variables que se pueden dar, computamos la media (ponderada) para cada variable (sujeta al total de interacciones).

SHAP se considera un método más robusto que LIME, puesto que tenemos una generalización (a diferencia de una regresión local) y hay más riqueza de datos.

Así pues, en cuanto a interpretación de resultados concretos, tenemos que absences (ausencias) y Fedu (nivel educativo del padre) tienen mayor relevancia que la predicha en el modelo original (Stochastic Gradient Boosting), mientras que G2 (Segundo Período) tiene una importancia significativamente menor.

A efectos prácticos, podríamos por ejemplo asignar un mayor peso a la variable absences (ausencias) para reflejar de una forma más certera su mayor relevancia.

Deep Learning:

En el contexto de nuestro dataset, lo que nos pueden aportar las técnicas de Deep Learning son, sobre todo, un mejor manejo de las relaciones no lineales y un mejor aprendizaje en cuanto a cómo se representan los diferentes features.

Así pues, podemos proponer un modelo de redes neuronales con la función de activación relu:

[22]:
```
random.seed(42)

dataset_cleaned = pd.read_csv('dataset_cleaned.csv')

label_encoders = {}
for column in dataset_cleaned.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    dataset_cleaned[column] = le.fit_transform(dataset_cleaned[column])
    label_encoders[column] = le

scaler = StandardScaler()
numerical_cols = dataset_cleaned.select_dtypes(include=['int64', 'float64']).
 ↪columns
dataset_cleaned[numerical_cols] = scaler.
 ↪fit_transform(dataset_cleaned[numerical_cols])

X = dataset_cleaned.drop('G3', axis=1)
y = dataset_cleaned['G3']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

#Construcción de la red neuronal con el optimizador adam
model = Sequential([
    Input(shape=(X_train.shape[1],)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')

#Entrenamiento de modelo, donde epochs es el número de iteraciones
history = model.fit(X_train, y_train, epochs=50, validation_split=0.2)
```

```python
#Evaluación y predicción
mse = model.evaluate(X_test, y_test)
print(f'Test MSE: {mse}')

predictions = model.predict(X_test)
```

```
Epoch 1/50
8/8              2s 29ms/step - loss:
0.8269 - val_loss: 0.4055
Epoch 2/50
8/8              0s 6ms/step - loss:
0.4507 - val_loss: 0.2732
Epoch 3/50
8/8              0s 9ms/step - loss:
0.2844 - val_loss: 0.2814
Epoch 4/50
8/8              0s 10ms/step - loss:
0.2110 - val_loss: 0.2307
Epoch 5/50
8/8              0s 8ms/step - loss:
0.2056 - val_loss: 0.1917
Epoch 6/50
8/8              0s 5ms/step - loss:
0.1237 - val_loss: 0.1710
Epoch 7/50
8/8              0s 8ms/step - loss:
0.1316 - val_loss: 0.1700
Epoch 8/50
8/8              0s 7ms/step - loss:
0.1079 - val_loss: 0.1686
Epoch 9/50
8/8              0s 8ms/step - loss:
0.1012 - val_loss: 0.1611
Epoch 10/50
8/8              0s 8ms/step - loss:
0.1078 - val_loss: 0.1591
Epoch 11/50
8/8              0s 8ms/step - loss:
0.0972 - val_loss: 0.1667
Epoch 12/50
8/8              0s 8ms/step - loss:
0.0693 - val_loss: 0.1668
Epoch 13/50
8/8              0s 8ms/step - loss:
0.0738 - val_loss: 0.1595
Epoch 14/50
8/8              0s 7ms/step - loss:
0.0692 - val_loss: 0.1655
```

```
Epoch 15/50
8/8            0s 7ms/step - loss:
0.0605 - val_loss: 0.1594
Epoch 16/50
8/8            0s 7ms/step - loss:
0.0661 - val_loss: 0.1573
Epoch 17/50
8/8            0s 8ms/step - loss:
0.0579 - val_loss: 0.1703
Epoch 18/50
8/8            0s 7ms/step - loss:
0.0556 - val_loss: 0.1595
Epoch 19/50
8/8            0s 8ms/step - loss:
0.0595 - val_loss: 0.1597
Epoch 20/50
8/8            0s 7ms/step - loss:
0.0463 - val_loss: 0.1678
Epoch 21/50
8/8            0s 7ms/step - loss:
0.0478 - val_loss: 0.1545
Epoch 22/50
8/8            0s 8ms/step - loss:
0.0384 - val_loss: 0.1584
Epoch 23/50
8/8            0s 7ms/step - loss:
0.0478 - val_loss: 0.1584
Epoch 24/50
8/8            0s 7ms/step - loss:
0.0375 - val_loss: 0.1625
Epoch 25/50
8/8            0s 8ms/step - loss:
0.0323 - val_loss: 0.1529
Epoch 26/50
8/8            0s 8ms/step - loss:
0.0358 - val_loss: 0.1711
Epoch 27/50
8/8            0s 6ms/step - loss:
0.0230 - val_loss: 0.1501
Epoch 28/50
8/8            0s 8ms/step - loss:
0.0275 - val_loss: 0.1644
Epoch 29/50
8/8            0s 7ms/step - loss:
0.0265 - val_loss: 0.1490
Epoch 30/50
8/8            0s 7ms/step - loss:
0.0208 - val_loss: 0.1583
```

```
Epoch 31/50
8/8            0s 8ms/step - loss:
0.0219 - val_loss: 0.1483
Epoch 32/50
8/8            0s 8ms/step - loss:
0.0209 - val_loss: 0.1523
Epoch 33/50
8/8            0s 8ms/step - loss:
0.0184 - val_loss: 0.1590
Epoch 34/50
8/8            0s 8ms/step - loss:
0.0215 - val_loss: 0.1466
Epoch 35/50
8/8            0s 7ms/step - loss:
0.0140 - val_loss: 0.1523
Epoch 36/50
8/8            0s 6ms/step - loss:
0.0146 - val_loss: 0.1504
Epoch 37/50
8/8            0s 7ms/step - loss:
0.0155 - val_loss: 0.1511
Epoch 38/50
8/8            0s 8ms/step - loss:
0.0119 - val_loss: 0.1449
Epoch 39/50
8/8            0s 8ms/step - loss:
0.0186 - val_loss: 0.1435
Epoch 40/50
8/8            0s 8ms/step - loss:
0.0115 - val_loss: 0.1379
Epoch 41/50
8/8            0s 8ms/step - loss:
0.0122 - val_loss: 0.1489
Epoch 42/50
8/8            0s 7ms/step - loss:
0.0100 - val_loss: 0.1428
Epoch 43/50
8/8            0s 7ms/step - loss:
0.0082 - val_loss: 0.1450
Epoch 44/50
8/8            0s 8ms/step - loss:
0.0115 - val_loss: 0.1305
Epoch 45/50
8/8            0s 7ms/step - loss:
0.0091 - val_loss: 0.1474
Epoch 46/50
8/8            0s 8ms/step - loss:
0.0081 - val_loss: 0.1350
```

```
Epoch 47/50
8/8             0s 8ms/step - loss:
0.0082 - val_loss: 0.1527
Epoch 48/50
8/8             0s 7ms/step - loss:
0.0075 - val_loss: 0.1253
Epoch 49/50
8/8             0s 8ms/step - loss:
0.0076 - val_loss: 0.1423
Epoch 50/50
8/8             0s 8ms/step - loss:
0.0046 - val_loss: 0.1279
3/3             0s 5ms/step - loss:
0.2263
Test MSE: 0.20375557243824005
3/3             0s 30ms/step
```

Habiendo hecho diferentes ajustes de epochs, obtenemos un MSE de 0.17 con 50 epochs. Por otra parte, no podemos computar feature importances debido a que la estructura no es una basada en árboles de decisión. El resultado previo está casi a la par con el del obtenido con Stochastic Gradient Boosting (igual a 0.16), si bien no lo mejora. La explicación puede estar en el hecho de que nuestro dataset no es lo suficientemente complejo como para que haya necesidad clara de recurrir a métodos de Deep Learning. No obstante, vamos a probar a continuación el método de Long Short-Term Memory (LSTM):

[23]:
```
random.seed(42)

dataset_cleaned = pd.read_csv('dataset_cleaned.csv')

label_encoders = {}
for column in dataset_cleaned.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    dataset_cleaned[column] = le.fit_transform(dataset_cleaned[column])
    label_encoders[column] = le

scaler = StandardScaler()
numerical_cols = dataset_cleaned.select_dtypes(include=['int64', 'float64']).
 ↪columns
dataset_cleaned[numerical_cols] = scaler.
 ↪fit_transform(dataset_cleaned[numerical_cols])

X = dataset_cleaned.drop('G3', axis=1)
y = dataset_cleaned['G3']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

#Formateamos los datos de forma acorde con LSTM
```

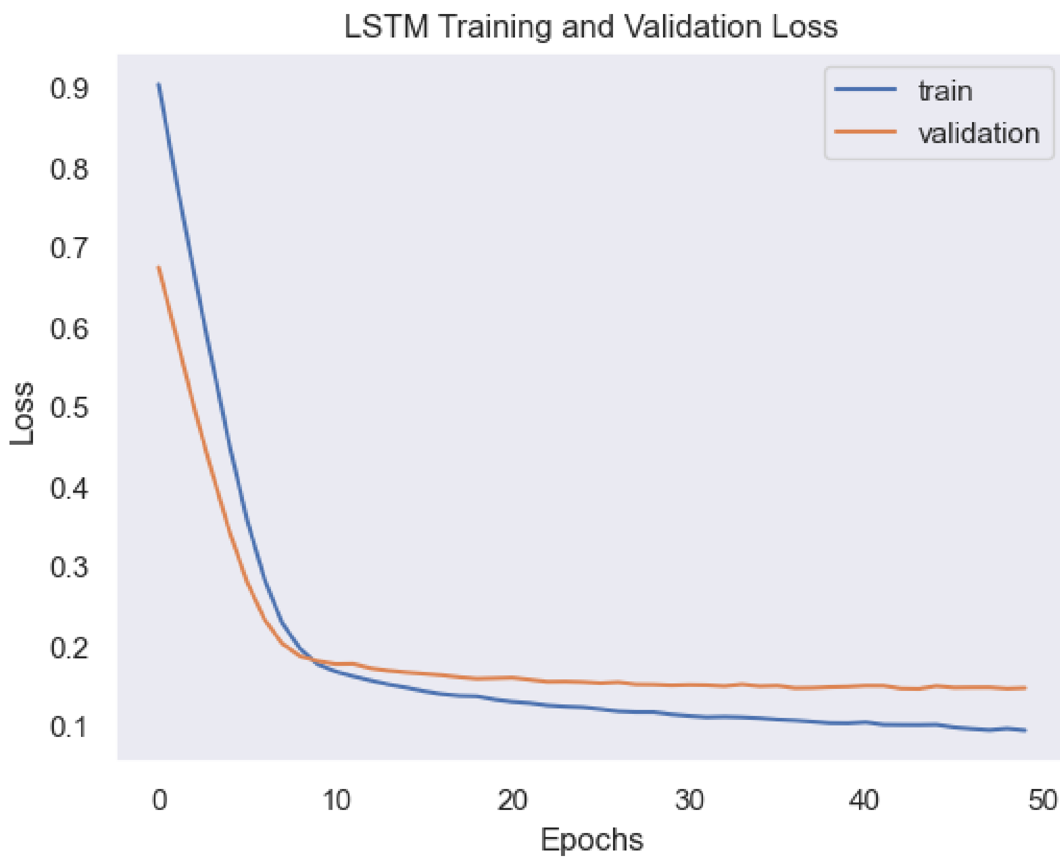
```python
X_train_reshaped = X_train.values.reshape((X_train.shape[0], 1, X_train.
 ↪shape[1]))
X_test_reshaped = X_test.values.reshape((X_test.shape[0], 1, X_test.shape[1]))

#Construcción del modelo LSTM
model_lstm = Sequential([
    Input(shape=(X_train_reshaped.shape[1], X_train_reshaped.shape[2])),
    LSTM(50),
    Dense(20, activation='relu'),
    Dense(1)
])

model_lstm.compile(optimizer='adam', loss='mse')

history_lstm = model_lstm.fit(X_train_reshaped, y_train, epochs=50,␣
 ↪validation_split=0.2)

mse_lstm = model_lstm.evaluate(X_test_reshaped, y_test)
print(f'Test MSE: {mse_lstm}')

predictions_lstm = model_lstm.predict(X_test_reshaped)

plt.plot(history_lstm.history['loss'], label='train')
plt.plot(history_lstm.history['val_loss'], label='validation')
plt.title('LSTM Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Epoch 1/50
8/8              4s 50ms/step - loss:
0.9595 - val_loss: 0.6742
Epoch 2/50
8/8              0s 7ms/step - loss:
0.8596 - val_loss: 0.5872
Epoch 3/50
8/8              0s 6ms/step - loss:
0.7253 - val_loss: 0.4988
Epoch 4/50
8/8              0s 10ms/step - loss:
0.5930 - val_loss: 0.4183
Epoch 5/50
8/8              0s 8ms/step - loss:
0.4857 - val_loss: 0.3432
Epoch 6/50
8/8              0s 9ms/step - loss:
```

```
0.3903 - val_loss: 0.2793
Epoch 7/50
8/8              0s 7ms/step - loss:
0.2663 - val_loss: 0.2318
Epoch 8/50
8/8              0s 6ms/step - loss:
0.2617 - val_loss: 0.2018
Epoch 9/50
8/8              0s 6ms/step - loss:
0.2248 - val_loss: 0.1863
Epoch 10/50
8/8              0s 6ms/step - loss:
0.1677 - val_loss: 0.1802
Epoch 11/50
8/8              0s 11ms/step - loss:
0.1413 - val_loss: 0.1763
Epoch 12/50
8/8              0s 7ms/step - loss:
0.1912 - val_loss: 0.1767
Epoch 13/50
8/8              0s 7ms/step - loss:
0.1405 - val_loss: 0.1711
Epoch 14/50
8/8              0s 9ms/step - loss:
0.1745 - val_loss: 0.1682
Epoch 15/50
8/8              0s 8ms/step - loss:
0.1495 - val_loss: 0.1660
Epoch 16/50
8/8              0s 9ms/step - loss:
0.1538 - val_loss: 0.1644
Epoch 17/50
8/8              0s 9ms/step - loss:
0.1331 - val_loss: 0.1625
Epoch 18/50
8/8              0s 9ms/step - loss:
0.1174 - val_loss: 0.1600
Epoch 19/50
8/8              0s 9ms/step - loss:
0.1328 - val_loss: 0.1581
Epoch 20/50
8/8              0s 9ms/step - loss:
0.1448 - val_loss: 0.1586
Epoch 21/50
8/8              0s 11ms/step - loss:
0.1408 - val_loss: 0.1593
Epoch 22/50
8/8              0s 7ms/step - loss:
```

```
0.1307 - val_loss: 0.1568
Epoch 23/50
8/8            0s 10ms/step - loss:
0.1176 - val_loss: 0.1542
Epoch 24/50
8/8            0s 9ms/step - loss:
0.1106 - val_loss: 0.1545
Epoch 25/50
8/8            0s 8ms/step - loss:
0.1282 - val_loss: 0.1538
Epoch 26/50
8/8            0s 9ms/step - loss:
0.1090 - val_loss: 0.1524
Epoch 27/50
8/8            0s 9ms/step - loss:
0.1095 - val_loss: 0.1536
Epoch 28/50
8/8            0s 9ms/step - loss:
0.1245 - val_loss: 0.1509
Epoch 29/50
8/8            0s 8ms/step - loss:
0.0969 - val_loss: 0.1507
Epoch 30/50
8/8            0s 6ms/step - loss:
0.1532 - val_loss: 0.1498
Epoch 31/50
8/8            0s 12ms/step - loss:
0.1313 - val_loss: 0.1503
Epoch 32/50
8/8            0s 12ms/step - loss:
0.1114 - val_loss: 0.1500
Epoch 33/50
8/8            0s 9ms/step - loss:
0.1324 - val_loss: 0.1488
Epoch 34/50
8/8            0s 9ms/step - loss:
0.1183 - val_loss: 0.1510
Epoch 35/50
8/8            0s 9ms/step - loss:
0.1009 - val_loss: 0.1487
Epoch 36/50
8/8            0s 12ms/step - loss:
0.1250 - val_loss: 0.1493
Epoch 37/50
8/8            0s 10ms/step - loss:
0.0918 - val_loss: 0.1462
Epoch 38/50
8/8            0s 11ms/step - loss:
```

```
0.1057 - val_loss: 0.1465
Epoch 39/50
8/8              0s 10ms/step - loss:
0.1044 - val_loss: 0.1475
Epoch 40/50
8/8              0s 8ms/step - loss:
0.0928 - val_loss: 0.1481
Epoch 41/50
8/8              0s 7ms/step - loss:
0.0852 - val_loss: 0.1493
Epoch 42/50
8/8              0s 8ms/step - loss:
0.1050 - val_loss: 0.1492
Epoch 43/50
8/8              0s 9ms/step - loss:
0.0893 - val_loss: 0.1456
Epoch 44/50
8/8              0s 13ms/step - loss:
0.0736 - val_loss: 0.1452
Epoch 45/50
8/8              0s 7ms/step - loss:
0.0943 - val_loss: 0.1489
Epoch 46/50
8/8              0s 9ms/step - loss:
0.1034 - val_loss: 0.1468
Epoch 47/50
8/8              0s 8ms/step - loss:
0.0971 - val_loss: 0.1471
Epoch 48/50
8/8              0s 8ms/step - loss:
0.1058 - val_loss: 0.1471
Epoch 49/50
8/8              0s 9ms/step - loss:
0.0861 - val_loss: 0.1455
Epoch 50/50
8/8              0s 9ms/step - loss:
0.1292 - val_loss: 0.1463
3/3              0s 4ms/step - loss:
0.2418
Test MSE: 0.20432646572589874
3/3              1s 177ms/step
```



El método de LSTM consiste, esencialmente, en un pipe (Long Memory) que se controla a través del mecanismo de Short-Term Memory a efectos de evitar o bien gradient vanishing o bien gradient explosion, una problemática común en Deep Learning. Al mismo tiempo, los datos se presentan de forma secuencial.

Obtenemos un MSE mejor (más bajo) mediante LSTM, de 0.20 en 50 epocs (que parece ser el número de iteraciones óptimo, siendo los otros epochs probados 25 y 100).

Este resultado sigue sin mejorar el obtenido por Stochastic Gradient Boosting. Por otra parte, no tendría sentido recurrir a Convolutional Neural Networks (CNNs) puesto que se utilizan normalmente en otro dominio (procesamiento de imágenes).

Puesto que ni LIME ni SHAP están planteados para trabajar con datos secuenciales (están optimizados para datos tabulares o imagenes), vamos a plantear la interpretabilidad para la Red Neuronal anterior:

[24]:
```
random.seed(42)

dataset_cleaned = pd.read_csv('dataset_cleaned.csv')

label_encoders = {}
```

```python
for column in dataset_cleaned.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    dataset_cleaned[column] = le.fit_transform(dataset_cleaned[column])
    label_encoders[column] = le

scaler = StandardScaler()
numerical_cols = dataset_cleaned.select_dtypes(include=['int64', 'float64']).
 ↪columns
dataset_cleaned[numerical_cols] = scaler.
 ↪fit_transform(dataset_cleaned[numerical_cols])

X = dataset_cleaned.drop('G3', axis=1)
y = dataset_cleaned['G3']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

#Construcción de la red neuronal con el optimizador adam
model = Sequential([
    Input(shape=(X_train.shape[1],)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')

#Entrenamiento de modelo, donde epochs es el número de iteraciones
history = model.fit(X_train, y_train, epochs=50, validation_split=0.2)

#Evaluación y predicción
mse = model.evaluate(X_test, y_test)
print(f'Test MSE: {mse}')

predictions = model.predict(X_test)

#Ajuste de formato
def nn_predict(input_data):
    return model.predict(input_data).flatten()

#Inicialización de LIME
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,
    feature_names=X_train.columns.tolist(),
    mode='regression'
)
```

```python
#Explicación de la primera instancia
instance_index = 0
exp = explainer.explain_instance(
    data_row=X_test.iloc[instance_index].values,
    predict_fn=nn_predict,
    num_features=5
)

exp.show_in_notebook(show_table=True)
```

```
Epoch 1/50
8/8               2s 31ms/step - loss:
0.8303 - val_loss: 0.4030
Epoch 2/50
8/8               0s 8ms/step - loss:
0.3899 - val_loss: 0.2769
Epoch 3/50
8/8               0s 8ms/step - loss:
0.3137 - val_loss: 0.2777
Epoch 4/50
8/8               0s 9ms/step - loss:
0.2189 - val_loss: 0.2270
Epoch 5/50
8/8               0s 9ms/step - loss:
0.1977 - val_loss: 0.1903
Epoch 6/50
8/8               0s 12ms/step - loss:
0.1469 - val_loss: 0.1814
Epoch 7/50
8/8               0s 7ms/step - loss:
0.1275 - val_loss: 0.1664
Epoch 8/50
8/8               0s 11ms/step - loss:
0.1268 - val_loss: 0.1654
Epoch 9/50
8/8               0s 10ms/step - loss:
0.1046 - val_loss: 0.1627
Epoch 10/50
8/8               0s 9ms/step - loss:
0.0933 - val_loss: 0.1598
Epoch 11/50
8/8               0s 9ms/step - loss:
0.1012 - val_loss: 0.1549
Epoch 12/50
8/8               0s 10ms/step - loss:
0.0809 - val_loss: 0.1629
Epoch 13/50
8/8               0s 8ms/step - loss:
```

```
0.0780 - val_loss: 0.1551
Epoch 14/50
8/8              0s 7ms/step - loss:
0.0821 - val_loss: 0.1570
Epoch 15/50
8/8              0s 8ms/step - loss:
0.0662 - val_loss: 0.1573
Epoch 16/50
8/8              0s 8ms/step - loss:
0.0578 - val_loss: 0.1668
Epoch 17/50
8/8              0s 9ms/step - loss:
0.0580 - val_loss: 0.1558
Epoch 18/50
8/8              0s 8ms/step - loss:
0.0573 - val_loss: 0.1541
Epoch 19/50
8/8              0s 8ms/step - loss:
0.0500 - val_loss: 0.1665
Epoch 20/50
8/8              0s 8ms/step - loss:
0.0469 - val_loss: 0.1483
Epoch 21/50
8/8              0s 8ms/step - loss:
0.0424 - val_loss: 0.1625
Epoch 22/50
8/8              0s 8ms/step - loss:
0.0408 - val_loss: 0.1514
Epoch 23/50
8/8              0s 8ms/step - loss:
0.0342 - val_loss: 0.1499
Epoch 24/50
8/8              0s 8ms/step - loss:
0.0312 - val_loss: 0.1567
Epoch 25/50
8/8              0s 7ms/step - loss:
0.0305 - val_loss: 0.1583
Epoch 26/50
8/8              0s 8ms/step - loss:
0.0320 - val_loss: 0.1514
Epoch 27/50
8/8              0s 8ms/step - loss:
0.0334 - val_loss: 0.1518
Epoch 28/50
8/8              0s 8ms/step - loss:
0.0336 - val_loss: 0.1469
Epoch 29/50
8/8              0s 10ms/step - loss:
```

```
0.0205 - val_loss: 0.1451
Epoch 30/50
8/8            0s 6ms/step - loss:
0.0327 - val_loss: 0.1487
Epoch 31/50
8/8            0s 9ms/step - loss:
0.0187 - val_loss: 0.1544
Epoch 32/50
8/8            0s 9ms/step - loss:
0.0193 - val_loss: 0.1428
Epoch 33/50
8/8            0s 10ms/step - loss:
0.0216 - val_loss: 0.1412
Epoch 34/50
8/8            0s 9ms/step - loss:
0.0233 - val_loss: 0.1514
Epoch 35/50
8/8            0s 9ms/step - loss:
0.0208 - val_loss: 0.1402
Epoch 36/50
8/8            0s 9ms/step - loss:
0.0196 - val_loss: 0.1469
Epoch 37/50
8/8            0s 6ms/step - loss:
0.0155 - val_loss: 0.1465
Epoch 38/50
8/8            0s 14ms/step - loss:
0.0136 - val_loss: 0.1452
Epoch 39/50
8/8            0s 10ms/step - loss:
0.0124 - val_loss: 0.1415
Epoch 40/50
8/8            0s 10ms/step - loss:
0.0126 - val_loss: 0.1517
Epoch 41/50
8/8            0s 10ms/step - loss:
0.0096 - val_loss: 0.1437
Epoch 42/50
8/8            0s 9ms/step - loss:
0.0073 - val_loss: 0.1381
Epoch 43/50
8/8            0s 9ms/step - loss:
0.0086 - val_loss: 0.1457
Epoch 44/50
8/8            0s 9ms/step - loss:
0.0095 - val_loss: 0.1401
Epoch 45/50
8/8            0s 10ms/step - loss:
```

```
0.0076 - val_loss: 0.1445
Epoch 46/50
8/8            0s 10ms/step - loss:
0.0067 - val_loss: 0.1483
Epoch 47/50
8/8            0s 9ms/step - loss:
0.0069 - val_loss: 0.1408
Epoch 48/50
8/8            0s 9ms/step - loss:
0.0069 - val_loss: 0.1335
Epoch 49/50
8/8            0s 8ms/step - loss:
0.0061 - val_loss: 0.1485
Epoch 50/50
8/8            0s 9ms/step - loss:
0.0060 - val_loss: 0.1464
3/3            0s 3ms/step - loss:
0.2394
Test MSE: 0.2157924771308899
WARNING:tensorflow:5 out of the last 7 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x00000179811B2520> triggered tf.function retracing. Tracing is expensive and
the excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
1/3            0s
93ms/stepWARNING:tensorflow:6 out of the last 9 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x00000179811B2520> triggered tf.function retracing. Tracing is expensive and
the excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
3/3            0s 31ms/step
157/157            0s 1ms/step
```

```
<IPython.core.display.HTML object>
```

Si bien, como hemos visto anteriormente, las redes neuronales no nos ofrecen la mejor predicción posible, si son bastante ilustrativas a efectos de interpretabilidad.

Así, debido a su naturaleza de capas múltiples, podemos ver que tanto G1 (Primer Período) como G2

(Segundo Período) parecen estar muy ligeramente sobreestimados respecto a los valores obtenidos sin este enfoque interpretable. Nuevamente, cabe matizar que en el marco de nuestro Sistema Experto no podemos ejercer influencia directa sobre ninguna de estas dos variables.

De esta forma, la conclusión que podemos sacar es que mientras que el MSE es ligeramente más alto en el caso de las Redes Neuronales, el ajuste parece ser más representativo, en tanto que hay muy poca diferencia entre el modelo LIME y el original.

[25]:
```
random.seed(42)

dataset_cleaned = pd.read_csv('dataset_cleaned.csv')

label_encoders = {}
for column in dataset_cleaned.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    dataset_cleaned[column] = le.fit_transform(dataset_cleaned[column])
    label_encoders[column] = le

scaler = StandardScaler()
numerical_cols = dataset_cleaned.select_dtypes(include=['int64', 'float64']).
 ↪columns
dataset_cleaned[numerical_cols] = scaler.
 ↪fit_transform(dataset_cleaned[numerical_cols])

X = dataset_cleaned.drop('G3', axis=1)
y = dataset_cleaned['G3']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

model = Sequential([
    Input(shape=(X_train.shape[1],)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')
history = model.fit(X_train, y_train, epochs=50, validation_split=0.2)
mse = model.evaluate(X_test, y_test)
print(f'Test MSE: {mse}')
predictions = model.predict(X_test)

predictions = model.predict(X_test)

#Utilizamos GradientExplainer en vez de DeepExplainer debido a falta de soporte␣
 ↪de DeepExplainer en TensorFlow > 2.4.0
background = X_train.sample(min(100, len(X_train)))
```

```
explainer = shap.GradientExplainer(model, background)

test_sample = X_test.sample(min(100, len(X_test)))
test_sample_features = test_sample.values
shap_values = explainer.shap_values(test_sample_features)

print(shap_values)
```

```
Epoch 1/50
8/8           2s 34ms/step - loss:
0.9108 - val_loss: 0.3988
Epoch 2/50
8/8           0s 7ms/step - loss:
0.4264 - val_loss: 0.2757
Epoch 3/50
8/8           0s 7ms/step - loss:
0.2822 - val_loss: 0.2709
Epoch 4/50
8/8           0s 9ms/step - loss:
0.1981 - val_loss: 0.2328
Epoch 5/50
8/8           0s 8ms/step - loss:
0.1615 - val_loss: 0.1894
Epoch 6/50
8/8           0s 9ms/step - loss:
0.1447 - val_loss: 0.1747
Epoch 7/50
8/8           0s 8ms/step - loss:
0.1302 - val_loss: 0.1629
Epoch 8/50
8/8           0s 7ms/step - loss:
0.0941 - val_loss: 0.1711
Epoch 9/50
8/8           0s 8ms/step - loss:
0.1336 - val_loss: 0.1580
Epoch 10/50
8/8           0s 8ms/step - loss:
0.0910 - val_loss: 0.1657
Epoch 11/50
8/8           0s 9ms/step - loss:
0.0778 - val_loss: 0.1626
Epoch 12/50
8/8           0s 8ms/step - loss:
0.0709 - val_loss: 0.1680
Epoch 13/50
8/8           0s 8ms/step - loss:
0.0710 - val_loss: 0.1556
Epoch 14/50
```

```
8/8            0s 11ms/step - loss:
0.0738 - val_loss: 0.1716
Epoch 15/50
8/8            0s 18ms/step - loss:
0.0663 - val_loss: 0.1611
Epoch 16/50
8/8            0s 8ms/step - loss:
0.0522 - val_loss: 0.1632
Epoch 17/50
8/8            0s 7ms/step - loss:
0.0546 - val_loss: 0.1652
Epoch 18/50
8/8            0s 8ms/step - loss:
0.0638 - val_loss: 0.1538
Epoch 19/50
8/8            0s 6ms/step - loss:
0.0408 - val_loss: 0.1644
Epoch 20/50
8/8            0s 7ms/step - loss:
0.0576 - val_loss: 0.1585
Epoch 21/50
8/8            0s 8ms/step - loss:
0.0458 - val_loss: 0.1659
Epoch 22/50
8/8            0s 8ms/step - loss:
0.0436 - val_loss: 0.1638
Epoch 23/50
8/8            0s 10ms/step - loss:
0.0311 - val_loss: 0.1679
Epoch 24/50
8/8            0s 10ms/step - loss:
0.0302 - val_loss: 0.1590
Epoch 25/50
8/8            0s 9ms/step - loss:
0.0320 - val_loss: 0.1564
Epoch 26/50
8/8            0s 9ms/step - loss:
0.0316 - val_loss: 0.1698
Epoch 27/50
8/8            0s 7ms/step - loss:
0.0384 - val_loss: 0.1563
Epoch 28/50
8/8            0s 8ms/step - loss:
0.0281 - val_loss: 0.1648
Epoch 29/50
8/8            0s 9ms/step - loss:
0.0272 - val_loss: 0.1610
Epoch 30/50
```

```
8/8             0s 11ms/step - loss:
0.0246 - val_loss: 0.1552
Epoch 31/50
8/8             0s 9ms/step - loss:
0.0207 - val_loss: 0.1582
Epoch 32/50
8/8             0s 7ms/step - loss:
0.0218 - val_loss: 0.1725
Epoch 33/50
8/8             0s 8ms/step - loss:
0.0216 - val_loss: 0.1508
Epoch 34/50
8/8             0s 8ms/step - loss:
0.0213 - val_loss: 0.1527
Epoch 35/50
8/8             0s 8ms/step - loss:
0.0141 - val_loss: 0.1609
Epoch 36/50
8/8             0s 7ms/step - loss:
0.0156 - val_loss: 0.1617
Epoch 37/50
8/8             0s 7ms/step - loss:
0.0188 - val_loss: 0.1514
Epoch 38/50
8/8             0s 9ms/step - loss:
0.0114 - val_loss: 0.1519
Epoch 39/50
8/8             0s 8ms/step - loss:
0.0133 - val_loss: 0.1534
Epoch 40/50
8/8             0s 8ms/step - loss:
0.0108 - val_loss: 0.1638
Epoch 41/50
8/8             0s 8ms/step - loss:
0.0102 - val_loss: 0.1557
Epoch 42/50
8/8             0s 8ms/step - loss:
0.0097 - val_loss: 0.1574
Epoch 43/50
8/8             0s 7ms/step - loss:
0.0099 - val_loss: 0.1488
Epoch 44/50
8/8             0s 8ms/step - loss:
0.0090 - val_loss: 0.1524
Epoch 45/50
8/8             0s 9ms/step - loss:
0.0094 - val_loss: 0.1555
Epoch 46/50
```

```
8/8             0s 9ms/step - loss:
0.0056 - val_loss: 0.1543
Epoch 47/50
8/8             0s 8ms/step - loss:
0.0072 - val_loss: 0.1551
Epoch 48/50
8/8             0s 7ms/step - loss:
0.0061 - val_loss: 0.1489
Epoch 49/50
8/8             0s 5ms/step - loss:
0.0075 - val_loss: 0.1460
Epoch 50/50
8/8             0s 5ms/step - loss:
0.0069 - val_loss: 0.1633
3/3             0s 5ms/step - loss:
0.2528
Test MSE: 0.23003347218036652
3/3             0s 33ms/step
3/3             0s 3ms/step
[[[ 2.90112160e-02]
  [ 8.20284486e-02]
  [ 3.91857296e-01]
  [ 1.19833209e-01]
  [ 1.66959707e-02]
  [-3.76915820e-02]
  [-1.41599908e-01]
  [ 1.27676979e-01]
  [-9.03578177e-02]]

 [[ 3.04919153e-01]
  [-1.20210208e-01]
  [-9.61534500e-01]
  [-1.77292198e-01]
  [-6.22484908e-02]
  [-3.05906124e-03]
  [ 1.47437513e-01]
  [ 3.71097289e-02]
  [ 1.44476265e-01]]

 [[-3.11932396e-02]
  [-1.86060760e-02]
  [-8.47220004e-01]
  [-4.19955224e-01]
  [ 4.79156226e-02]
  [ 2.20841542e-02]
  [ 2.62478262e-01]
  [-1.87981352e-02]
  [-2.92911649e-01]]
```

```
[[-1.70274600e-01]
 [ 1.31244967e-02]
 [-1.04149997e+00]
 [-1.01374552e-01]
 [-2.26109430e-01]
 [ 3.17267388e-01]
 [ 1.06554836e-01]
 [ 4.50981185e-02]
 [-3.57285649e-01]]

[[-2.62426380e-02]
 [ 3.05778701e-02]
 [ 7.80632913e-01]
 [ 1.97123304e-01]
 [-4.25626002e-02]
 [ 3.37222479e-02]
 [ 4.69609685e-02]
 [-1.89462882e-02]
 [-1.86220389e-02]]

[[ 9.16957930e-02]
 [-8.61497894e-02]
 [ 4.80147809e-01]
 [ 2.19703130e-02]
 [-9.99475643e-03]
 [ 1.52517473e-02]
 [ 5.00223823e-02]
 [ 9.77111757e-02]
 [ 3.20623256e-02]]

[[ 1.33920424e-02]
 [ 1.17714502e-01]
 [ 2.58808825e-02]
 [ 1.79140102e-02]
 [ 2.24555489e-02]
 [ 4.17630039e-02]
 [ 1.67061966e-02]
 [ 1.34838715e-01]
 [ 7.82928318e-02]]

[[ 2.43320409e-02]
 [-1.88259706e-02]
 [-1.25035930e+00]
 [-5.47817528e-01]
 [ 3.44172828e-02]
 [-1.16057517e-02]
 [ 1.88175380e-01]
```

```
 [-3.44208106e-02]
 [ 3.12736601e-01]]

[[-2.65266925e-01]
 [ 6.38393313e-02]
 [-5.64333975e-01]
 [-9.20422524e-02]
 [-3.36883456e-01]
 [ 2.33804174e-02]
 [ 8.11063573e-02]
 [-4.74622771e-02]
 [-4.52992260e-01]]

[[-1.79645717e-02]
 [-6.39526546e-02]
 [ 7.92428672e-01]
 [ 2.13763669e-01]
 [-7.12271873e-03]
 [-3.62744927e-02]
 [ 3.21943425e-02]
 [-2.99000554e-02]
 [ 9.56699625e-02]]

[[-1.99698396e-02]
 [ 5.56565262e-03]
 [ 5.04200041e-01]
 [-2.45972490e-03]
 [ 4.44834121e-02]
 [-2.06204001e-02]
 [ 3.01534049e-02]
 [ 1.02653373e-02]
 [ 1.30988643e-01]]

[[ 6.80022836e-02]
 [-1.08660892e-01]
 [-1.22324371e+00]
 [-2.91103631e-01]
 [-6.42609457e-03]
 [-1.20696843e-01]
 [ 1.85616300e-01]
 [-5.84609434e-03]
 [ 2.61530966e-01]]

[[ 1.52094170e-01]
 [-1.20017137e-02]
 [-1.18933752e-01]
 [-1.41021307e-03]
 [ 8.56901705e-02]
```

```
 [-1.00084832e-02]
 [ 8.88629854e-02]
 [ 7.08129117e-03]
 [ 1.46855444e-01]]

[[-1.23444954e-02]
 [-8.75498280e-02]
 [-5.89702606e-01]
 [-8.12704191e-02]
 [-1.52205564e-02]
 [ 1.41708395e-02]
 [ 5.36298044e-02]
 [-1.81455553e-01]
 [ 9.17808786e-02]]

[[ 1.96818728e-02]
 [-8.53664875e-02]
 [-2.46076524e-01]
 [-5.17415144e-02]
 [-9.61163640e-02]
 [ 4.80605057e-03]
 [ 6.03307970e-02]
 [-5.22400588e-02]
 [ 2.20684856e-01]]

[[ 2.08296239e-01]
 [-1.88491959e-02]
 [-1.26801580e-01]
 [ 4.81087789e-02]
 [-1.79734603e-02]
 [-7.45430216e-02]
 [ 7.23465905e-02]
 [-1.93952005e-02]
 [-1.23994693e-01]]

[[ 2.11272482e-02]
 [-1.14036866e-01]
 [ 1.36280334e+00]
 [ 4.60427254e-01]
 [-2.03608628e-02]
 [-7.29907900e-02]
 [ 9.74815059e-03]
 [ 3.49739590e-03]
 [-1.80482306e-02]]

[[-2.39096526e-02]
 [ 2.69445442e-02]
 [-1.05480218e+00]
```

```
 [-1.86867446e-01]
 [ 1.10785052e-01]
 [ 1.16007321e-01]
 [ 9.44724306e-02]
 [ 8.26710165e-02]
 [ 3.18390012e-01]]

[[-2.62362719e-01]
 [ 6.25691175e-01]
 [-7.76326001e-01]
 [-1.59615353e-01]
 [-1.66501582e-01]
 [-7.26624429e-02]
 [ 5.04967645e-02]
 [ 1.31862700e-01]
 [-9.85517502e-02]]

[[ 9.88784581e-02]
 [-7.26830065e-02]
 [ 4.74090129e-01]
 [ 1.69676766e-01]
 [ 1.26284957e-01]
 [ 6.27649724e-02]
 [ 8.08920525e-03]
 [-5.10544218e-02]
 [ 1.45373285e-01]]

[[ 2.81307008e-02]
 [-8.46807659e-02]
 [ 9.46528018e-01]
 [ 2.62553185e-01]
 [-1.65302008e-02]
 [-2.63418201e-02]
 [ 2.89506428e-02]
 [-1.40382210e-02]
 [-1.28107524e-04]]

[[-2.88532730e-02]
 [-2.22350098e-02]
 [ 4.38327879e-01]
 [ 3.06118722e-03]
 [-2.01157816e-02]
 [-2.51425058e-03]
 [ 2.99731251e-02]
 [ 2.72180811e-02]
 [ 9.04267505e-02]]

[[ 2.49155005e-03]
```

```
 [-1.57091498e-01]
 [ 9.39646840e-01]
 [ 6.66103512e-02]
 [ 5.16508706e-02]
 [ 1.12331145e-01]
 [ 3.86471711e-02]
 [-1.70894712e-02]
 [-4.77204956e-02]]

[[ 9.11180899e-02]
 [-6.53002337e-02]
 [-8.30588222e-01]
 [-2.53869385e-01]
 [-4.82414141e-02]
 [-1.78372204e-01]
 [ 1.21829271e-01]
 [-2.10626107e-02]
 [ 5.86118996e-02]]

[[ 1.14015274e-01]
 [ 2.08169475e-01]
 [-6.94454134e-01]
 [ 5.77659272e-02]
 [ 4.40240763e-02]
 [ 6.98717590e-03]
 [ 1.76014546e-02]
 [ 4.20205407e-02]
 [ 6.47358596e-02]]

[[-9.08757150e-02]
 [ 1.10778376e-01]
 [-3.14386129e-01]
 [-3.99289727e-02]
 [-2.92386800e-01]
 [ 2.45331470e-02]
 [-1.63043082e-01]
 [ 4.69599850e-02]
 [ 3.23725045e-01]]

[[-8.53949338e-02]
 [ 2.37687454e-01]
 [-1.41847193e-01]
 [-4.56333011e-02]
 [-2.23421231e-02]
 [ 1.32108390e-01]
 [-1.04312897e-01]
 [-4.49379859e-03]
 [-3.82824428e-02]]
```

```
[[ 7.13013187e-02]
 [-4.99468222e-02]
 [ 6.20004475e-01]
 [ 5.81573211e-02]
 [-1.90806277e-02]
 [ 1.67273264e-02]
 [ 2.16431301e-02]
 [ 2.21223291e-02]
 [ 5.32879792e-02]]

[[-7.29636997e-02]
 [ 1.57597348e-01]
 [-1.83503330e-01]
 [ 1.92760229e-02]
 [-2.77095083e-02]
 [-8.84234309e-02]
 [ 1.18082248e-01]
 [-7.42527237e-03]
 [ 5.14582247e-02]]

[[-2.22131386e-01]
 [ 4.82063532e-01]
 [-8.23998809e-01]
 [-1.84196487e-01]
 [-6.42141104e-02]
 [-7.81996399e-02]
 [ 1.39638320e-01]
 [-7.13496804e-02]
 [-3.73496979e-01]]

[[-5.59736155e-02]
 [ 2.67412752e-01]
 [-9.36697602e-01]
 [-2.06566289e-01]
 [ 8.14006701e-02]
 [ 2.60029584e-01]
 [-8.03662837e-02]
 [ 3.80419157e-02]
 [-5.87982461e-02]]

[[ 2.05712579e-02]
 [-4.53909785e-02]
 [ 7.92040586e-01]
 [ 1.44044653e-01]
 [-1.93305016e-02]
 [ 6.20152988e-02]
 [ 3.82443294e-02]
```

```
 [-2.54630987e-02]
 [-2.01735906e-02]]

[[ 1.73711795e-02]
 [ 1.07804038e-01]
 [-3.50995153e-01]
 [ 3.56894508e-02]
 [ 3.52699310e-02]
 [ 4.11116444e-02]
 [ 4.20949645e-02]
 [ 2.81808954e-02]
 [ 1.48502678e-01]]

[[ 6.60096630e-02]
 [ 1.18286818e-01]
 [-3.80056143e-01]
 [-3.98451537e-02]
 [ 3.10509354e-02]
 [ 9.71030444e-02]
 [ 6.03823438e-02]
 [-2.21631471e-02]
 [ 1.76039711e-01]]

[[-6.14093393e-02]
 [-1.37010857e-01]
 [-1.07154393e+00]
 [-1.00734122e-01]
 [-3.60963829e-02]
 [ 9.55811888e-02]
 [ 8.80006626e-02]
 [ 1.40697360e-01]
 [-2.85808474e-01]]

[[-8.33016410e-02]
 [ 1.24597810e-01]
 [-2.29794681e-01]
 [ 9.31607932e-02]
 [-3.32968533e-02]
 [-8.37563351e-03]
 [ 5.71582690e-02]
 [ 2.25783163e-03]
 [-5.90150468e-02]]

[[-2.57446785e-02]
 [ 4.06768173e-02]
 [-4.54503417e-01]
 [-6.03167340e-02]
 [-3.27416286e-02]
```

```
 [ 5.69549613e-02]
 [-1.21273354e-01]
 [-9.21368971e-03]
 [ 1.93575934e-01]]

[[-3.90425744e-03]
 [-5.38437478e-02]
 [-2.59837270e-01]
 [-2.22749233e-01]
 [ 1.09212466e-01]
 [ 2.33033225e-02]
 [-1.79454267e-01]
 [ 5.35805225e-02]
 [ 2.74111301e-01]]

[[-2.80200951e-02]
 [ 7.50070214e-02]
 [ 2.11237609e-01]
 [ 1.09600775e-01]
 [-6.88057207e-03]
 [ 3.29566896e-02]
 [-5.73240481e-02]
 [-5.08067086e-02]
 [ 1.67262629e-02]]

[[ 5.62944412e-02]
 [-8.61997157e-02]
 [ 8.85668278e-01]
 [ 1.19820252e-01]
 [ 3.67453545e-02]
 [-7.45086819e-02]
 [ 5.26695400e-02]
 [-4.16113362e-02]
 [ 3.08550242e-02]]

[[-8.24291706e-02]
 [ 1.97179131e-02]
 [ 8.54205787e-01]
 [ 3.27187121e-01]
 [-2.88436823e-02]
 [ 1.42093385e-02]
 [ 3.14197503e-02]
 [-3.96473147e-03]
 [ 5.24117611e-02]]

[[-1.02479011e-01]
 [ 4.04006153e-01]
 [-3.42461467e-01]
```

```
 [-1.74979329e-01]
 [-8.93119536e-03]
 [-7.02479109e-02]
 [ 1.03988536e-01]
 [ 3.47461067e-02]
 [-1.88152701e-01]]

[[-2.25806125e-02]
 [-2.86203306e-02]
 [-1.02322924e+00]
 [-5.15830457e-01]
 [ 1.03916928e-01]
 [-2.66162567e-02]
 [ 2.02378422e-01]
 [-2.26787347e-02]
 [ 3.35870087e-02]]

[[-4.45599407e-02]
 [ 4.43764869e-03]
 [ 2.75602162e-01]
 [ 2.54833102e-01]
 [ 1.84115954e-02]
 [-7.65273999e-03]
 [-5.53232878e-02]
 [ 4.40008491e-02]
 [ 3.45706306e-02]]

[[-1.63129464e-01]
 [ 1.95958391e-01]
 [-3.28974426e-01]
 [-9.24027562e-02]
 [ 7.92614743e-02]
 [-1.38683632e-01]
 [ 4.10857014e-02]
 [ 3.73602323e-02]
 [-2.19819903e-01]]

[[ 2.83281151e-02]
 [ 3.63990553e-02]
 [ 1.48365080e+00]
 [ 3.05419266e-01]
 [ 1.62409455e-01]
 [-1.96500029e-02]
 [ 1.14060435e-02]
 [ 5.70261143e-02]
 [ 6.47771582e-02]]

[[ 1.82125717e-01]
```

```
 [-1.94253981e-01]
 [ 9.32897553e-02]
 [-8.67190957e-02]
 [-4.97584678e-02]
 [ 2.47910902e-01]
 [-3.42562854e-01]
 [ 3.06982826e-02]
 [-1.13407001e-01]]

[[-1.24042675e-01]
 [-1.01084411e-01]
 [-5.95066249e-01]
 [ 4.29770984e-02]
 [-4.08995338e-02]
 [ 3.65842283e-01]
 [ 1.15147354e-02]
 [ 1.02119535e-01]
 [-9.29672346e-02]]

[[ 2.92099379e-02]
 [-7.26569444e-02]
 [ 3.88800114e-01]
 [-6.42540213e-03]
 [ 2.98512522e-02]
 [ 9.00092423e-02]
 [-3.21851909e-01]
 [ 3.17855962e-02]
 [ 9.21452716e-02]]

[[-1.15886934e-01]
 [-1.84813544e-01]
 [ 1.33845794e+00]
 [ 3.27523917e-01]
 [ 3.40607241e-02]
 [-1.31786347e-03]
 [-6.57506287e-02]
 [-3.32805738e-02]
 [-9.26429555e-02]]

[[-4.28241864e-03]
 [-4.04734947e-02]
 [ 2.82078296e-01]
 [ 2.75718421e-02]
 [-2.09383089e-02]
 [-7.90426228e-03]
 [ 3.61214429e-02]
 [ 6.24241978e-02]
 [ 1.46229327e-01]]
```

```
[[-3.97546887e-02]
 [ 1.18292226e-02]
 [ 6.49937510e-01]
 [ 1.54437184e-01]
 [ 1.25128925e-01]
 [-3.09429858e-02]
 [ 1.51004903e-02]
 [ 8.34770221e-03]
 [-4.87902900e-03]]

[[ 5.00519648e-02]
 [-2.75820736e-02]
 [ 7.11146235e-01]
 [ 2.06023887e-01]
 [-1.53196752e-02]
 [-1.22547187e-02]
 [ 6.27827947e-04]
 [ 1.30938683e-02]
 [ 9.14000869e-02]]

[[-3.84496152e-02]
 [-3.20481807e-02]
 [ 8.98686051e-01]
 [ 2.80358166e-01]
 [ 4.70355228e-02]
 [-7.30110100e-03]
 [ 5.58451936e-02]
 [-1.36220992e-01]
 [-6.16303384e-02]]

[[ 1.92901656e-01]
 [-4.57237422e-01]
 [-1.20863545e+00]
 [-4.13059592e-01]
 [-2.34972499e-03]
 [-1.28945395e-01]
 [-2.59595633e-01]
 [ 1.41237983e-02]
 [ 3.01384151e-01]]

[[ 6.52388632e-02]
 [ 1.82747399e-03]
 [ 8.56425464e-01]
 [ 1.51146933e-01]
 [-6.91990182e-02]
 [-1.48233725e-02]
 [-4.96737957e-02]
```

```
 [-3.36535810e-03]
 [ 2.95401402e-02]]

[[ 1.03223465e-01]
 [-5.96899800e-02]
 [ 5.52912533e-01]
 [ 4.62218300e-02]
 [-1.77439470e-02]
 [ 3.26024834e-03]
 [ 4.60461825e-02]
 [ 7.66490400e-03]
 [ 6.23002499e-02]]

[[ 4.40346859e-02]
 [-9.37176794e-02]
 [-1.61413753e+00]
 [-8.70501280e-01]
 [-2.47527957e-01]
 [-3.66435125e-02]
 [ 1.93371087e-01]
 [-1.73390564e-02]
 [-4.54560548e-01]]

[[ 2.53936532e-03]
 [ 1.12308428e-01]
 [-9.96655405e-01]
 [-4.10615087e-01]
 [ 2.19300129e-02]
 [-4.25585210e-02]
 [-1.04399815e-01]
 [ 1.50518328e-01]
 [ 3.45692545e-01]]

[[ 3.12334411e-02]
 [-1.11690937e-02]
 [ 8.66684794e-01]
 [ 4.18344527e-01]
 [-1.84924006e-02]
 [-1.32495286e-02]
 [ 4.48045209e-02]
 [ 3.43090645e-03]
 [ 2.13202145e-02]]

[[-1.26973465e-02]
 [ 1.51538715e-01]
 [-9.96313930e-01]
 [-3.24178249e-01]
 [-1.21332798e-02]
```

```
 [ 2.48166993e-02]
 [-6.74945712e-02]
 [ 8.40189680e-02]
 [ 1.50969729e-01]]

[[-1.11374687e-02]
 [-7.97885004e-03]
 [ 6.45427406e-01]
 [ 3.01458299e-01]
 [ 5.67963012e-02]
 [ 1.84577964e-02]
 [ 2.84393504e-02]
 [-8.35322067e-02]
 [-4.66323225e-03]]

[[-2.96615213e-02]
 [-1.67248938e-02]
 [-5.83757833e-02]
 [-7.93541372e-02]
 [-3.71756800e-03]
 [ 7.69000649e-02]
 [ 9.51437578e-02]
 [-5.42932563e-03]
 [-2.06677645e-01]]

[[-1.22151591e-01]
 [ 1.75460219e-01]
 [ 1.91058561e-01]
 [ 3.52677815e-02]
 [ 3.85336503e-02]
 [ 2.66408265e-01]
 [ 4.38224263e-02]
 [ 9.19974409e-03]
 [-1.54168472e-01]]

[[ 7.91258886e-02]
 [-6.85325488e-02]
 [ 1.52760708e+00]
 [ 2.35357419e-01]
 [ 7.53304362e-02]
 [-3.83821432e-03]
 [ 2.19160672e-02]
 [ 2.50799712e-02]
 [ 3.83120626e-02]]

[[-6.99497238e-02]
 [ 3.45090777e-01]
 [-7.17625260e-01]
```

```
 [-1.70270324e-01]
 [-5.85290864e-02]
 [-1.06588848e-01]
 [ 1.67120665e-01]
 [-1.44559637e-01]
 [-2.13531271e-01]]

[[-3.29865277e-01]
 [-4.73160744e-02]
 [ 2.54909694e-01]
 [ 1.80932432e-01]
 [-8.14081728e-02]
 [-1.00204833e-02]
 [ 1.07511990e-01]
 [ 7.36801047e-03]
 [-1.74732536e-01]]

[[-6.23625293e-02]
 [-7.50221033e-03]
 [-5.82452118e-01]
 [-6.16332591e-02]
 [-1.86002422e-02]
 [-3.10141165e-02]
 [ 5.98340742e-02]
 [ 1.04670517e-01]
 [ 1.67910114e-01]]

[[-1.58983283e-02]
 [ 6.50016591e-02]
 [ 6.44541323e-01]
 [ 5.51463850e-02]
 [-1.73434112e-02]
 [ 2.11363751e-03]
 [-7.21645653e-02]
 [ 1.25107333e-01]
 [ 3.41910832e-02]]

[[-5.58807701e-02]
 [ 1.83371842e-01]
 [ 2.26620249e-02]
 [ 2.03521773e-02]
 [-9.18605253e-02]
 [-2.76679210e-02]
 [ 3.93906422e-02]
 [ 5.23809195e-02]
 [-4.82549518e-02]]

[[ 1.07885845e-01]
```

```
 [-1.53850481e-01]
 [ 1.27582884e+00]
 [ 1.89316273e-01]
 [-6.54215273e-03]
 [ 1.54137313e-02]
 [ 4.39750589e-02]
 [ 5.11779301e-02]
 [-2.33806553e-03]]

[[ 2.73555573e-02]
 [-1.97653137e-02]
 [ 6.86943114e-01]
 [ 1.36482060e-01]
 [-3.26942988e-02]
 [ 3.86358169e-03]
 [ 4.71740626e-02]
 [ 1.65281780e-02]
 [ 5.33048771e-02]]

[[-1.86526254e-02]
 [ 1.60317272e-02]
 [ 4.12188441e-01]
 [ 1.06559388e-01]
 [ 4.54528518e-02]
 [-3.44111174e-02]
 [ 1.63464211e-02]
 [ 2.32030209e-02]
 [ 1.06256686e-01]]

[[-1.31760731e-01]
 [ 4.81663719e-02]
 [ 4.44980085e-01]
 [ 1.48928523e-01]
 [ 1.60079058e-02]
 [-6.05304912e-02]
 [-2.65677303e-01]
 [ 1.26897702e-02]
 [-8.98199975e-02]]

[[-1.98412295e-02]
 [-7.40716383e-02]
 [ 1.31690338e-01]
 [ 2.85755340e-02]
 [-2.56076939e-02]
 [ 1.48752369e-02]
 [ 3.89769487e-02]
 [-1.99670330e-01]
 [-1.53622642e-01]]
```

```
[[-2.58039460e-02]
 [ 2.53760647e-02]
 [ 9.99624729e-01]
 [ 1.77400440e-01]
 [-4.21858206e-03]
 [ 1.78889688e-02]
 [-9.45648253e-02]
 [ 6.24645650e-02]
 [ 5.90634234e-02]]

[[-4.06046398e-03]
 [ 8.92381091e-03]
 [-2.58995324e-01]
 [-1.55736119e-01]
 [-7.47717097e-02]
 [-2.92072352e-02]
 [ 1.34472281e-01]
 [ 1.09475963e-01]
 [-3.08919549e-01]]

[[ 2.06515774e-01]
 [-6.36917174e-01]
 [-5.07127643e-01]
 [-2.39635393e-01]
 [ 8.35231990e-02]
 [-1.10846110e-01]
 [-2.94696659e-01]
 [ 1.07934706e-01]
 [ 9.90302674e-03]]

[[ 1.59122925e-02]
 [ 4.96286675e-02]
 [ 9.05069888e-01]
 [ 3.78807724e-01]
 [ 1.90264061e-02]
 [ 2.82904766e-02]
 [-7.14705363e-02]
 [ 5.67957386e-02]
 [-1.12574942e-01]]]
```

Debido a la naturaleza secuencial de los datos y la utilización de una versión de TensorFlow > 2.4.0 no resulta posible conseguir una representación gráfica. En cualquier caso, observando los resultados vemos como en un modelo SHAP (que, como hemos explicado antes, ofrece una mejor generalización de datos) la desviación entre el modelo SHAP y el modelo original (de Redes Neuronales) es muy pequeña.

Como conclusión general de interpretabilidad en Deep Learning, podemos concluir que para este dataset en concreto, los modelos de Machine Learning ofrecen un ajuste más exacto pero son también más proclives a subestimar o sobreestimar la importancia de determinados features, mientras que en los modelos de Deep Learning ocurre lo opuesto.

Nota final: para una mejor comprensión de los modelos LIME y SHAP se han utilizado de referencia los respectivos epígrafes del capítulo “Local Model-Agnostic Methods” del libro Interpretable Machine Learning de Christoph Molnar (2022, 2ª Edición).