Ejercicio 2 - AVANZADO

April 20, 2025

1 Nombre: Miller Alexis Quintero García

Observación: Personalmente modifiqué ligeramente un poco esta sección inicial de preparación de los datos, para tener mayor noción de que es lo que tengo, obtener un poco más de información, visualizar los dataframes en cada etapa y revisar si existe algún desbalanceo en los datos. Todo lo comenté ya sea con comentarios de línea en Python, o con texto en celdas Markdown.

En mi computador de escritorio que tiene un procesador AMD Ryzen 5 3400G, 16 GB de RAM y no tiene tarjeta gráfica, el notebook tardó aproximadamente 6 minutos en correr de inicio a fin.

De antemano, muchas gracias.

1.1 EJERCICIO 2 - AVANZADO

El fichero `hr_train.csv` contiene una tabla sobre los empleados de una empresa. Entre otros incluye información de su nivel de satisfacción, de la última evaluación obtenida, del número de proyecto en el que se encuentra, del número medio de horas por semana, del número de años que lleva en la compañia, si ha tenido o no un accidente en el trabajo, **si se ha ido (left = 1)** de la compañia, si ha tenido una promoción en los últimos 5 años, y por último su nivel salarial (descrito como medio, alto o bajo) y el departamento en que se encuentra (sales/technnical …).

El departamento de RRHH está solicitando un clasificador que pueda predecir si alguien esta en riesgo de irse de la compañia para poder actuar proactivamente.

Notas: - La columna “left” indica si alguien se ha ido de la compañia, es la **“y”** del problema y el objetivo de la predicción. - Las columnas “sales” y “salary” son alfanuméricas. La mayoría de clasificadores (aunque no todos) no trabajan bien con este tipo de variables y requieren valores numéricos. Dado el caso dispones de diferentes alternativas: (1) Eliminar esas columnas y trabajar con el resto. (2) Traducir esas columnas en columnas binarias; por ejemplo si “salary” tiene tres valores, eliminar la columna original y sustituirla por tres columnas salary_low, salary_medium, salary_high donde se pondra un 1 en el tipo de salario y un 0 en las otras dos columnas restantes. Lo mismo para “sales”, donde se transformara esa columna en tantas columnas como departamentos y un 1 en el departamento en que se encuentre el trabajador. Este es un preproceso muy típico.

Nota de evaluación:

En este ejercicio más que el resultado final (que también) se evaluarán los pasos y el razonamiento utilizado en cada decisión que se toma para la implementación del modelo.

1.2 Preparación de los datos

Desde mi punto de vista es importante incluir los datos relacionados con el salario y el departamento. Por un lado el salario siempre es importante en cualquier trabajo, cualquier persona razona un poco antes de renunciar si su salario es destacado según su sector y profesión; por otro lado el departamento al que pertenece un empleado también es importante, no todos los empleados se sienten igual en el mismo departamento.

Debido a las razones anteriores, considero importante incluir dichas variables de forma apropiada para los modelos de aprendizaje automático, para esto es necesario entonces convertir dichas variables alfanuméricas a númericas categóricas agregando la serie de columnas necesarias.

[1]:
```
import pandas as pd
import numpy as np

file1 = r'hr_train.csv'

ci = pd.read_csv(file1)
ci
```

[1]:

	satisfaction_level	last_evaluation	number_project	\
0	0.42	0.46	2	
1	0.66	0.77	2	
2	0.55	0.49	5	
3	0.22	0.88	4	
4	0.20	0.72	6	
…	…	…	…	
10494	0.82	0.84	3	
10495	0.85	0.81	3	
10496	0.32	0.95	5	
10497	0.51	0.76	4	
10498	0.80	0.68	4	

	average_montly_hours	time_spend_company	Work_accident	left	\
0	150	3	0	1	
1	171	2	0	0	
2	240	3	0	0	
3	213	3	1	0	
4	224	4	0	1	
…	…	…	…	…	
10494	237	2	0	0	
10495	205	3	0	0	
10496	172	2	0	1	
10497	140	3	0	1	
10498	199	2	0	0	

	promotion_last_5years	sales	salary
0	0	sales	medium

```
1                          0  technical  medium
2                          0  technical    high
3                          0  technical  medium
4                          0  technical  medium
…                        …        …     …
10494                      0  technical     low
10495                      0  marketing    high
10496                      0      sales     low
10497                      0    support     low
10498                      0         IT  medium

[10499 rows x 10 columns]
```

[2]:
```
# Vemos que variedad de valores alfanuméricos tiene la variable 'salary'
print(ci.salary.value_counts())
# Y la cantidad de valores únicos
print(f"\nHay {ci.salary.nunique()} valores únicos en la variable 'salary'")
```

```
salary
low       5104
medium    4515
high       880
Name: count, dtype: int64

Hay 3 valores únicos en la variable 'salary'
```

[3]:
```
# Vemos que variedad de valores alfanúmericos tiene la variable 'sales'
print(ci.sales.value_counts())
# Y la cantidad de valores únicos
print(f"\nHay {ci.sales.nunique()} valores únicos en la variable 'sales'")
```

```
sales
sales          2935
technical      1890
support        1556
IT              825
product_mng     639
marketing       614
hr              532
RandD           530
accounting      527
management      451
Name: count, dtype: int64

Hay 10 valores únicos en la variable 'sales'
```

Como hay 3 valores únicos en ‘salary’ y 10 en ‘sales’, implica que una vez convertidos los datos a categóricos numéricos, el dataframe tendrá 13 columnas nuevas.

[4]:
```
# Importamos el módulo de OneHotEncoder que permite convertir variables␣
 ↪alfanuméricas en variables binarias
from sklearn.preprocessing import OneHotEncoder

# Creamos el objeto OneHotEncoder
encoder = OneHotEncoder(sparse_output = False, handle_unknown = 'ignore', dtype␣
 ↪= np.int8)
# Ajustamos el objeto a las variables categóricas
encoder.fit(ci[['salary', 'sales']])

# Obtenemos el nombre de las columnas que se generarán
columns = encoder.get_feature_names_out(['salary', 'sales'])
# Creamos un DataFrame con las columnas generadas
df = pd.DataFrame(encoder.transform(ci[['salary', 'sales']]), columns=columns)
# Concatenamos el DataFrame original con el nuevo DataFrame
ci = pd.concat([ci, df], axis=1)

ci
```

[4]:

	satisfaction_level	last_evaluation	number_project	…
0	0.42	0.46	2	
1	0.66	0.77	2	
2	0.55	0.49	5	
3	0.22	0.88	4	
4	0.20	0.72	6	
…	…	…	…	
10494	0.82	0.84	3	
10495	0.85	0.81	3	
10496	0.32	0.95	5	
10497	0.51	0.76	4	
10498	0.80	0.68	4	

	average_montly_hours	time_spend_company	Work_accident	left	…
0	150	3	0	1	
1	171	2	0	0	
2	240	3	0	0	
3	213	3	1	0	
4	224	4	0	1	
…	…	…	…	…	…
10494	237	2	0	0	
10495	205	3	0	0	
10496	172	2	0	1	
10497	140	3	0	1	
10498	199	2	0	0	

	promotion_last_5years	sales	salary	…	sales_IT	sales_RandD	…
0	0	sales	medium	…	0	0	

```
1                        0  technical  medium  …         0            0
2                        0  technical    high  …         0            0
3                        0  technical  medium  …         0            0
4                        0  technical  medium  …         0            0
…                      …       …     …  …       …          …
10494                    0  technical     low  …         0            0
10495                    0  marketing    high  …         0            0
10496                    0      sales     low  …         0            0
10497                    0    support     low  …         0            0
10498                    0         IT  medium  …         1            0

       sales_accounting  sales_hr  sales_management  sales_marketing  \
0                     0         0                 0                0
1                     0         0                 0                0
2                     0         0                 0                0
3                     0         0                 0                0
4                     0         0                 0                0
…                   …       …                 …              …
10494                 0         0                 0                0
10495                 0         0                 0                1
10496                 0         0                 0                0
10497                 0         0                 0                0
10498                 0         0                 0                0

       sales_product_mng  sales_sales  sales_support  sales_technical
0                      0            1              0                0
1                      0            0              0                1
2                      0            0              0                1
3                      0            0              0                1
4                      0            0              0                1
…                    …          …              …              …
10494                  0            0              0                1
10495                  0            0              0                0
10496                  0            1              0                0
10497                  0            0              1                0
10498                  0            0              0                0

[10499 rows x 23 columns]
```

[5]:
```
# Eliminamos las columnas de las variables alfanuméricas originales
ci.drop(['salary', 'sales'], axis=1, inplace=True)
# Mostramos el DataFrame resultante
ci
```

[5]:
```
       satisfaction_level  last_evaluation  number_project  \
0                    0.42             0.46               2
1                    0.66             0.77               2
```

```
2                    0.55            0.49               5
3                    0.22            0.88               4
4                    0.20            0.72               6
…                    …               …                  …
10494                0.82            0.84               3
10495                0.85            0.81               3
10496                0.32            0.95               5
10497                0.51            0.76               4
10498                0.80            0.68               4

       average_montly_hours  time_spend_company  Work_accident  left  \
0                       150                   3              0     1
1                       171                   2              0     0
2                       240                   3              0     0
3                       213                   3              1     0
4                       224                   4              0     1
…                       …                     …              …     …
10494                   237                   2              0     0
10495                   205                   3              0     0
10496                   172                   2              0     1
10497                   140                   3              0     1
10498                   199                   2              0     0

       promotion_last_5years  salary_high  salary_low  …  sales_IT  \
0                          0            0           0  …         0
1                          0            0           0  …         0
2                          0            1           0  …         0
3                          0            0           0  …         0
4                          0            0           0  …         0
…                          …            …           …  …         …
10494                      0            0           1  …         0
10495                      0            1           0  …         0
10496                      0            0           1  …         0
10497                      0            0           1  …         0
10498                      0            0           0  …         1

       sales_RandD  sales_accounting  sales_hr  sales_management  \
0                0                 0         0                 0
1                0                 0         0                 0
2                0                 0         0                 0
3                0                 0         0                 0
4                0                 0         0                 0
…                …                 …         …                 …
10494            0                 0         0                 0
10495            0                 0         0                 0
10496            0                 0         0                 0
10497            0                 0         0                 0
```

```
10498            0                 0         0                 0

       sales_marketing  sales_product_mng  sales_sales  sales_support  \
0                    0                  0            1              0
1                    0                  0            0              0
2                    0                  0            0              0
3                    0                  0            0              0
4                    0                  0            0              0
…                  …                  …            …              …
10494                0                  0            0              0
10495                1                  0            0              0
10496                0                  0            1              0
10497                0                  0            0              1
10498                0                  0            0              0

       sales_technical
0                    0
1                    1
2                    1
3                    1
4                    1
…                  …
10494                1
10495                0
10496                0
10497                0
10498                0

[10499 rows x 21 columns]
```

[6]:
```
# Verificamos si hay o no valores de nulidad o NaN
print(ci.isna().sum())
```

```
satisfaction_level       0
last_evaluation          0
number_project           0
average_montly_hours     0
time_spend_company       0
Work_accident            0
left                     0
promotion_last_5years    0
salary_high              0
salary_low               0
salary_medium            0
sales_IT                 0
sales_RandD              0
sales_accounting         0
sales_hr                 0
```

```
sales_management         0
sales_marketing          0
sales_product_mng        0
sales_sales              0
sales_support            0
sales_technical          0
dtype: int64
```

No hay valores None o NaN en las columnas de las variables

[7]:
```
ci.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10499 entries, 0 to 10498
Data columns (total 21 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     10499 non-null  float64
 1   last_evaluation        10499 non-null  float64
 2   number_project         10499 non-null  int64
 3   average_montly_hours   10499 non-null  int64
 4   time_spend_company     10499 non-null  int64
 5   Work_accident          10499 non-null  int64
 6   left                   10499 non-null  int64
 7   promotion_last_5years  10499 non-null  int64
 8   salary_high            10499 non-null  int8
 9   salary_low             10499 non-null  int8
 10  salary_medium          10499 non-null  int8
 11  sales_IT               10499 non-null  int8
 12  sales_RandD            10499 non-null  int8
 13  sales_accounting       10499 non-null  int8
 14  sales_hr               10499 non-null  int8
 15  sales_management       10499 non-null  int8
 16  sales_marketing        10499 non-null  int8
 17  sales_product_mng      10499 non-null  int8
 18  sales_sales            10499 non-null  int8
 19  sales_support          10499 non-null  int8
 20  sales_technical        10499 non-null  int8
dtypes: float64(2), int64(6), int8(13)
memory usage: 789.6 KB
```

[8]:
```
ci.describe()
```

[8]:
```
       satisfaction_level  last_evaluation  number_project  \
count        10499.000000     10499.000000    10499.000000
mean             0.612683         0.717131        3.808553
std              0.248578         0.171483        1.230572
min              0.090000         0.360000        2.000000
```

```
25%              0.440000         0.560000        3.000000
50%              0.640000         0.720000        4.000000
75%              0.820000         0.870000        5.000000
max              1.000000         1.000000        7.000000

       average_montly_hours  time_spend_company  Work_accident          left  \
count          10499.000000        10499.000000   10499.000000  10499.000000
mean             201.059815            3.494238       0.144299      0.292885
std               49.959332            1.453227       0.351410      0.455108
min               96.000000            2.000000       0.000000      0.000000
25%              156.000000            3.000000       0.000000      0.000000
50%              200.000000            3.000000       0.000000      0.000000
75%              245.000000            4.000000       0.000000      1.000000
max              310.000000           10.000000       1.000000      1.000000

       promotion_last_5years   salary_high    salary_low  …      sales_IT  \
count           10499.000000  10499.000000  10499.000000  …  10499.000000
mean                0.021716      0.083818      0.486142  …      0.078579
std                 0.145763      0.277127      0.499832  …      0.269093
min                 0.000000      0.000000      0.000000  …      0.000000
25%                 0.000000      0.000000      0.000000  …      0.000000
50%                 0.000000      0.000000      0.000000  …      0.000000
75%                 0.000000      0.000000      1.000000  …      0.000000
max                 1.000000      1.000000      1.000000  …      1.000000

        sales_RandD  sales_accounting      sales_hr  sales_management  \
count  10499.000000      10499.000000  10499.000000      10499.000000
mean       0.050481          0.050195      0.050671          0.042956
std        0.218946          0.218358      0.219336          0.202769
min        0.000000          0.000000      0.000000          0.000000
25%        0.000000          0.000000      0.000000          0.000000
50%        0.000000          0.000000      0.000000          0.000000
75%        0.000000          0.000000      0.000000          0.000000
max        1.000000          1.000000      1.000000          1.000000

       sales_marketing  sales_product_mng  sales_sales  sales_support  \
count     10499.000000       10499.000000  10499.00000   10499.000000
mean          0.058482           0.060863      0.27955       0.148205
std           0.234663           0.239090      0.44880       0.355320
min           0.000000           0.000000      0.00000       0.000000
25%           0.000000           0.000000      0.00000       0.000000
50%           0.000000           0.000000      0.00000       0.000000
75%           0.000000           0.000000      1.00000       0.000000
max           1.000000           1.000000      1.00000       1.000000

       sales_technical
count     10499.000000
```

```
mean          0.180017
std           0.384220
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           1.000000

[8 rows x 21 columns]
```

Vamos a separar ahora las variable objetivo **'left'** de los datos.

[9]:
```
# Tomamos la columna 'left' como variable dependiente y el resto como variables independientes
X = ci.drop(['left'], axis=1)
y = ci['left']
X
```

[9]:
```
       satisfaction_level  last_evaluation  number_project  \
0                    0.42             0.46               2
1                    0.66             0.77               2
2                    0.55             0.49               5
3                    0.22             0.88               4
4                    0.20             0.72               6
…                     …                …                …
10494                0.82             0.84               3
10495                0.85             0.81               3
10496                0.32             0.95               5
10497                0.51             0.76               4
10498                0.80             0.68               4

       average_montly_hours  time_spend_company  Work_accident  \
0                       150                   3              0
1                       171                   2              0
2                       240                   3              0
3                       213                   3              1
4                       224                   4              0
…                        …                    …              …
10494                   237                   2              0
10495                   205                   3              0
10496                   172                   2              0
10497                   140                   3              0
10498                   199                   2              0

       promotion_last_5years  salary_high  salary_low  salary_medium  \
0                          0            0           0              1
1                          0            0           0              1
2                          0            1           0              0
```

```
3                        0            0           0              1
4                        0            0           0              1
…                      …            …           …              …
10494                    0            0           1              0
10495                    0            1           0              0
10496                    0            0           1              0
10497                    0            0           1              0
10498                    0            0           0              1

       sales_IT  sales_RandD  sales_accounting  sales_hr  sales_management  \
0             0            0                 0         0                 0
1             0            0                 0         0                 0
2             0            0                 0         0                 0
3             0            0                 0         0                 0
4             0            0                 0         0                 0
…           …            …               …         …               …
10494         0            0                 0         0                 0
10495         0            0                 0         0                 0
10496         0            0                 0         0                 0
10497         0            0                 0         0                 0
10498         1            0                 0         0                 0

       sales_marketing  sales_product_mng  sales_sales  sales_support  \
0                    0                  0            1              0
1                    0                  0            0              0
2                    0                  0            0              0
3                    0                  0            0              0
4                    0                  0            0              0
…                  …                  …            …            …
10494                0                  0            0              0
10495                1                  0            0              0
10496                0                  0            1              0
10497                0                  0            0              1
10498                0                  0            0              0

       sales_technical
0                    0
1                    1
2                    1
3                    1
4                    1
…                  …
10494                1
10495                0
10496                0
10497                0
10498                0
```

```
[10499 rows x 20 columns]
```

[10]:
```
y
```

[10]:
```
0        1
1        0
2        0
3        0
4        1
        ..
10494    0
10495    0
10496    1
10497    1
10498    0
Name: left, Length: 10499, dtype: int64
```

[11]:
```
# Verificamos si hay desbalance en la variable de salida
y.value_counts()
```

[11]:
```
left
0    7424
1    3075
Name: count, dtype: int64
```

Como hay un desbalance entre las clases a clasificar, en la función `train_test_split` especificamos que se repartan los datos guardando la proporción con el argumento `stratify = y`.

[12]:
```
from sklearn.model_selection import train_test_split
# Dividimos el conjunto de datos en un 80% para entrenamiento y un 20% para test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 42, stratify = y)

# Reseteamos los índices de los DataFrames
X_train.reset_index(drop = True, inplace = True)
X_test.reset_index(drop = True, inplace = True)
y_train.reset_index(drop = True, inplace = True)
y_test.reset_index(drop = True, inplace = True)
```

[13]:
```
X_train
```

[13]:
```
      satisfaction_level  last_evaluation  number_project  \
0                   0.66             0.79               5
1                   0.93             0.97               3
2                   0.86             0.71               3
3                   0.74             0.99               4
4                   0.84             0.64               2
```

```
…                    …                …                …
8394                0.99             0.54                3
8395                0.94             0.73                4
8396                0.95             0.74                4
8397                0.43             0.56                2
8398                0.11             0.83                6

      average_montly_hours  time_spend_company  Work_accident  \
0                      134                   3              0
1                      256                   2              1
2                      235                   3              0
3                      233                   5              0
4                      211                   3              0
…                      …                     …              …
8394                   247                   3              0
8395                   204                   2              0
8396                   258                   3              0
8397                   129                   3              0
8398                   244                   4              0

      promotion_last_5years  salary_high  salary_low  salary_medium  sales_IT  \
0                         0            1           0              0         0
1                         0            0           1              0         0
2                         0            0           0              1         1
3                         0            0           1              0         1
4                         0            0           0              1         0
…                         …            …           …              …         …
8394                      0            0           0              1         0
8395                      0            0           1              0         0
8396                      0            0           0              1         0
8397                      0            0           1              0         0
8398                      0            0           0              1         0

      sales_RandD  sales_accounting  sales_hr  sales_management  \
0               0                 0         0                 1
1               0                 0         0                 0
2               0                 0         0                 0
3               0                 0         0                 0
4               0                 0         0                 0
…               …                 …         …                 …
8394            1                 0         0                 0
8395            0                 0         0                 0
8396            0                 0         0                 0
8397            0                 0         0                 0
8398            0                 0         1                 0

      sales_marketing  sales_product_mng  sales_sales  sales_support  \
```

```
0                    0                    0            0              0
1                    0                    1            0              0
2                    0                    0            0              0
3                    0                    0            0              0
4                    0                    0            1              0
…                  …                    …            …              …
8394                 0                    0            0              0
8395                 0                    0            1              0
8396                 0                    0            0              0
8397                 0                    0            1              0
8398                 0                    0            0              0

      sales_technical
0                    0
1                    0
2                    0
3                    0
4                    0
…                  …
8394                 0
8395                 0
8396                 1
8397                 0
8398                 0

[8399 rows x 20 columns]
```

[14]:
```
X_test
```

[14]:
```
      satisfaction_level  last_evaluation  number_project  \
0                   0.82             0.98               5
1                   0.98             0.84               4
2                   0.44             0.54               2
3                   0.76             0.84               5
4                   0.60             0.65               2
…                   …                …                  …
2095                0.64             0.38               2
2096                0.67             0.97               4
2097                0.86             0.84               3
2098                0.34             0.46               5
2099                0.20             0.90               6

      average_montly_hours  time_spend_company  Work_accident  \
0                      234                   5              0
1                      200                   2              0
2                      151                   3              0
3                      249                   3              0
```

```
4                      225                   10              0
…                      …                     …               …
2095                   269                    5              0
2096                   186                    3              0
2097                   177                    3              0
2098                   131                    3              0
2099                   138                    3              0

      promotion_last_5years  salary_high  salary_low  salary_medium  sales_IT  \
0                         0            0           0              1         0
1                         0            0           0              1         0
2                         0            0           1              0         0
3                         0            0           1              0         0
4                         0            1           0              0         0
…                         …            …           …              …         …
2095                      0            0           1              0         0
2096                      0            0           1              0         0
2097                      0            0           0              1         0
2098                      0            0           0              1         0
2099                      0            0           1              0         0

      sales_RandD  sales_accounting  sales_hr  sales_management  \
0               0                 0         0                 0
1               0                 0         0                 0
2               0                 0         0                 1
3               0                 0         0                 0
4               0                 0         0                 1
…               …                 …         …                 …
2095            0                 0         0                 0
2096            0                 0         1                 0
2097            0                 0         0                 0
2098            0                 0         0                 0
2099            0                 0         0                 0

      sales_marketing  sales_product_mng  sales_sales  sales_support  \
0                   1                  0            0              0
1                   0                  1            0              0
2                   0                  0            0              0
3                   0                  0            0              1
4                   0                  0            0              0
…                   …                  …            …              …
2095                0                  0            0              0
2096                0                  0            0              0
2097                0                  0            1              0
2098                1                  0            0              0
2099                0                  0            0              0
```

```
      sales_technical
0                   0
1                   0
2                   0
3                   0
4                   0
…                 …
2095                1
2096                0
2097                0
2098                0
2099                1

[2100 rows x 20 columns]
```

[15]:
```
y_train
```

[15]:
```
0       0
1       0
2       0
3       1
4       0
       ..
8394    1
8395    0
8396    1
8397    1
8398    1
Name: left, Length: 8399, dtype: int64
```

[16]:
```
y_test
```

[16]:
```
0       1
1       0
2       1
3       0
4       0
       ..
2095    1
2096    0
2097    0
2098    0
2099    1
Name: left, Length: 2100, dtype: int64
```

1.3 Definición de métrica clave

La prioridad de RRHH es tomar acciones frente a empleados con alta posibilidad de renunciar, dicha información está contenida en los datos objetivo **y**, de manera que un empleado que renuncio es un **1**, por tal motivo del modelo se debe centrar en detectar lo mejor posible los casos positivos, de manera que tener falsos negativos es peor que tener falsos positivos, pues es preferible tomar acciones inclusive hasta en algunos empleados que probablemente no se iban a ir, que no tomar acciones en empleados que probablemente si se van a ir.

Por tal motivo considero que la métrica de decisión debe ser el ***recall*** ya que si observamos su ecuación:

$$Recall = \frac{TP}{TP + FN}$$

Vemos que esta métrica se puede maximizar si los casos de falsos negativos (empleados que si es probable que se vayan pero se predijo que no) se disminuyen.

[17]:
```
# Importamos la métrica de 'accuracy' y 'recall'
from sklearn.metrics import accuracy_score, recall_score
```

1.4 Normalización de datos

Ya que vamos a probar variedad de modelos, algunos de estos requerirán datos normalizados, como en el análisis exploratorio de los datos no se vieron outliers de ningún tipo, por lo que se usará simplemente un MinMaxScaler.

[18]:
```
# Importamos el normalizador MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

# Creamos el objeto MinMaxScaler
scaler = MinMaxScaler()
# Ajustamos el objeto a los datos de entrenamiento
scaler.fit(X_train)
# Transformamos los datos de entrenamiento y test
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[19]:
```
X_train_scaled
```

[19]:
```
array([[0.62637363, 0.671875  , 0.6       , …, 0.        , 0.        ,
        0.        ],
       [0.92307692, 0.953125  , 0.2       , …, 0.        , 0.        ,
        0.        ],
       [0.84615385, 0.546875  , 0.2       , …, 0.        , 0.        ,
        0.        ],
       …,
       [0.94505495, 0.59375   , 0.4       , …, 0.        , 0.        ,
        1.        ],
```

```
        [0.37362637, 0.3125    , 0.        , …, 1.        , 0.        ,
         0.        ],
        [0.02197802, 0.734375  , 0.8       , …, 0.        , 0.        ,
         0.        ]])
```

[20]:
```
X_test_scaled
```

[20]:
```
array([[0.8021978 , 0.96875   , 0.6       , …, 0.        , 0.        ,
        0.        ],
       [0.97802198, 0.75      , 0.4       , …, 0.        , 0.        ,
        0.        ],
       [0.38461538, 0.28125   , 0.        , …, 0.        , 0.        ,
        0.        ],
       …,
       [0.84615385, 0.75      , 0.2       , …, 1.        , 0.        ,
        0.        ],
       [0.27472527, 0.15625   , 0.6       , …, 0.        , 0.        ,
        0.        ],
       [0.12087912, 0.84375   , 0.8       , …, 0.        , 0.        ,
        1.        ]])
```

1.5 Modelos

1.5.1 Logistic Regression

Desde un punto de vista de pensarlo como un problema de predecir probabilidades vamos a empezar utilizando regresión logistica.

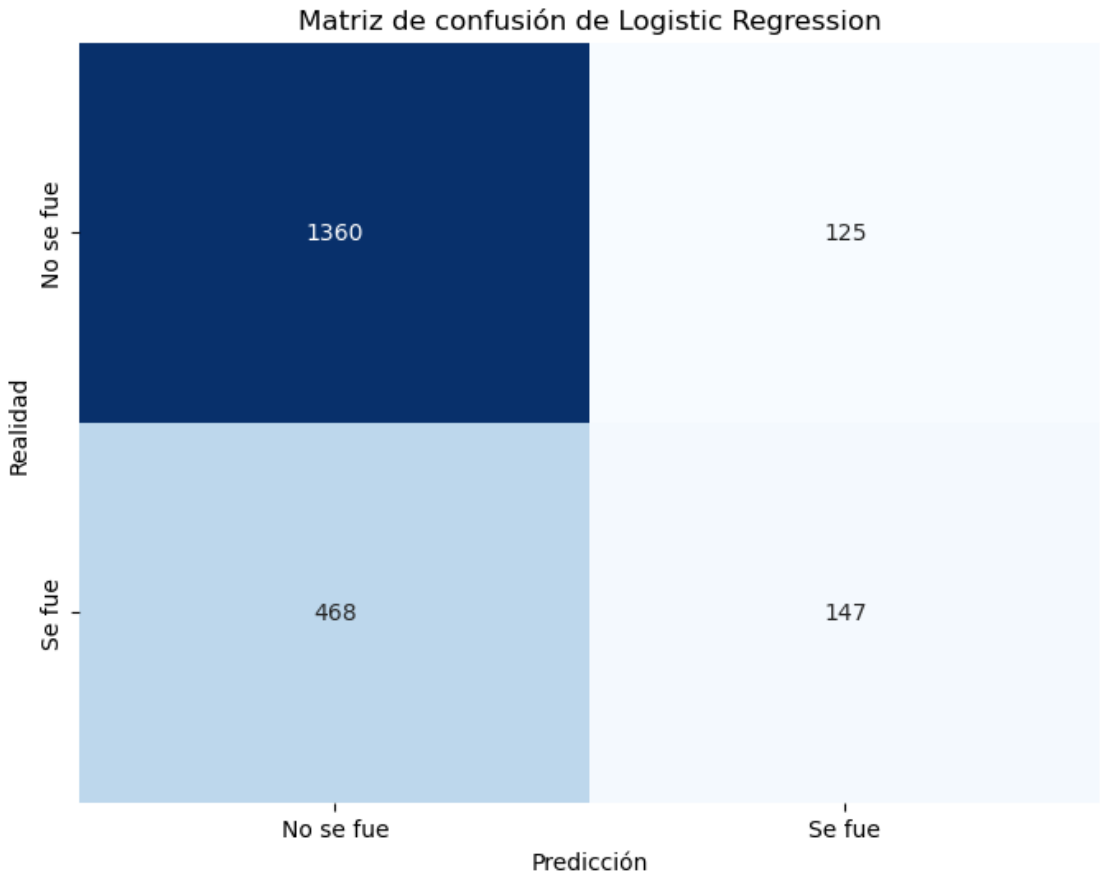
[21]:
```
# Importamos el módulo de LogisticRegression
from sklearn.linear_model import LogisticRegression

# Creamos el objeto LogisticRegression
model = LogisticRegression(max_iter = 1000, random_state = 42)
# Ajustamos el modelo a los datos de entrenamiento
model.fit(X_train_scaled, y_train)

# Predecimos los valores de la variable dependiente para el conjunto de test
y_pred = model.predict(X_test_scaled)

# Calculamos la métrica de 'accuracy' en train para ver que no hay overfitting
y_train_pred = model.predict(X_train_scaled)
print(f"Accuracy en train: {accuracy_score(y_train, y_train_pred)}")
# Calculamos la métrica de 'accuracy' en test
print(f"Accuracy en test: {accuracy_score(y_test, y_pred)}")

# Calculamos la métrica de 'recall' en test
print(f"Recall en test: {recall_score(y_test, y_pred)}")
```

```
Accuracy en train: 0.7213954042147874
Accuracy en test: 0.7176190476190476
Recall en test: 0.23902439024390243
```

[22]:
```python
# Mostramos la probabilidad de pertenecer a cada clase
y_pred_proba = model.predict_proba(X_test_scaled)
# Vemos cuantas instancias del conjunto de test tienen una probabilidad mayor o igual a 0.5 de pertenecer a la clase 1
print(f"Hay {np.sum(y_pred_proba[:, 1] >= 0.5)} instancias del conjunto de test con probabilidad mayor o igual a 0.5 de pertenecer a la clase 1")
# Vemos cuantas instancias del conjunto de test tienen una probabilidad mayor o igual a 0.5 de pertenecer a la clase 0
print(f"Hay {np.sum(y_pred_proba[:, 0] >= 0.5)} instancias del conjunto de test con probabilidad mayor o igual a 0.5 de pertenecer a la clase 0")

# Mostramos la cantidad de instancias totales del conjunto de test
print(f"Hay {len(y_test)} instancias en total en el conjunto de test")
```

```
Hay 272 instancias del conjunto de test con probabilidad mayor o igual a 0.5 de pertenecer a la clase 1
Hay 1828 instancias del conjunto de test con probabilidad mayor o igual a 0.5 de pertenecer a la clase 0
Hay 2100 instancias en total en el conjunto de test
```

[23]:
```python
# Veamos ahora la matriz de confusión
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Creamos la matriz de confusión
cm = confusion_matrix(y_test, y_pred)
# Creamos un DataFrame con la matriz de confusión
cm_df = pd.DataFrame(cm, index = ['No se fue', 'Se fue'], columns = ['No se fue', 'Se fue'])
# Mostramos la matriz de confusión
plt.figure(figsize = (8, 6))
sns.heatmap(cm_df, annot = True, fmt = 'd', cmap = 'Blues', cbar = False)
plt.title('Matriz de confusión de Logistic Regression')
plt.xlabel('Predicción')
plt.ylabel('Realidad')
plt.show()
```



En general vemos que el modelo tiene un error importante al observar los falsos negativos, de manera que hay muchas instancias en las que predijo que el empleado no se fue, cuando en realidad si se fue, esto es grave ya que nos aleja del objetivo que tiene el departamento de RRHH con el modelo, el cuál es predecir lo mejor posible dichos casos, y esa falencia del modelo se ve reflejada en su baja métrica de **recall**.

1.5.2 Gradient Boosting Classifier

Este es otro modelo importante a considerar, pasando por encima del Decission Tree y el Random Forest, ya que tiene usualmente ventajas frente al primero en desempeño, y en rendimiento frente al segundo.

Para hacerlo lo mejor posible vamos a probar diferentes learning rate y número de estimadores con GridSearchCV.

[24]:
```
# Importamos el módulo de GradientBoostingClassifier y GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV

# Creamos el objeto GradientBoostingClassifier
```

```
gbc = GradientBoostingClassifier(random_state = 42)
# Definimos los parámetros a ajustar
params = {
    'learning_rate': [0.3, 0.5, 1.0]
    , 'n_estimators': [100, 200, 300]
}

# Creamos el objeto GridSearchCV optimizando la métrica de 'recall'
grid = GridSearchCV(estimator = gbc, param_grid = params, scoring = 'recall',␣
 ↪cv = 10)
# Ajustamos el objeto a los datos de entrenamiento, considerando que un GBC no␣
 ↪requiere normalización
grid.fit(X_train, y_train)
```

[24]:
```
GridSearchCV(cv=10, estimator=GradientBoostingClassifier(random_state=42),
             param_grid={'learning_rate': [0.3, 0.5, 1.0],
                         'n_estimators': [100, 200, 300]},
             scoring='recall')
```

[25]:
```
# Vemos los resultados del GridSearchCV
grid_results = pd.DataFrame(grid.cv_results_).loc[:, ['params',␣
 ↪'mean_test_score', 'std_test_score', 'rank_test_score']]
print("Los mejores parámetros son:", grid.best_params_)
grid_results
```

```
Los mejores parámetros son: {'learning_rate': 0.5, 'n_estimators': 200}
```

[25]:

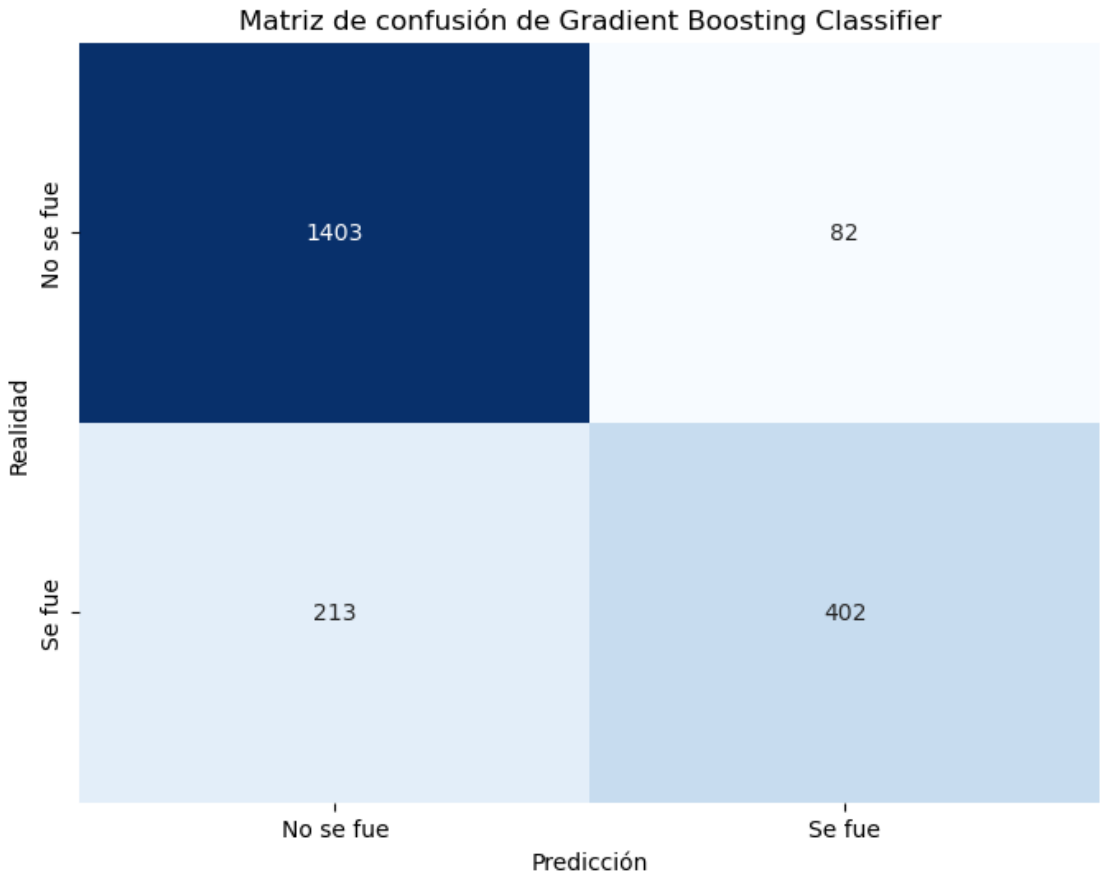
	params	mean_test_score	std_test_score	rank_test_score
0	{'learning_rate': 0.3, 'n_estimators': 100}	0.691463	0.024080	8
1	{'learning_rate': 0.3, 'n_estimators': 200}	0.693089	0.024542	7
2	{'learning_rate': 0.3, 'n_estimators': 300}	0.694715	0.030117	4
3	{'learning_rate': 0.5, 'n_estimators': 100}	0.690244	0.025177	9
4	{'learning_rate': 0.5, 'n_estimators': 200}	0.695935	0.029189	1
5	{'learning_rate': 0.5, 'n_estimators': 300}	0.695122	0.026219	3
6	{'learning_rate': 1.0, 'n_estimators': 100}	0.694309	0.024104	6
7	{'learning_rate': 1.0, 'n_estimators': 200}	0.694715	0.021079	4
8	{'learning_rate': 1.0, 'n_estimators': 300}	0.695935		

```
8         0.028210                  1
```

[26]:
```
# Tomamos el mejor modelo del GridSearchCV
gbc_best = grid.best_estimator_

# Predecimos los valores de la variable dependiente para el conjunto de test
y_pred_gbc = gbc_best.predict(X_test)

# Calculamos la métrica de 'accuracy' en train para ver que no hay overfitting
y_train_pred_gbc = gbc_best.predict(X_train)
print(f"Accuracy en train: {accuracy_score(y_train, y_train_pred_gbc)}")
# Calculamos la métrica de 'accuracy' en test
print(f"Accuracy en test: {accuracy_score(y_test, y_pred_gbc)}")
# Calculamos la métrica de 'recall' en test
print(f"Recall en test: {recall_score(y_test, y_pred_gbc)}")
```

```
Accuracy en train: 0.9097511608524824
Accuracy en test: 0.8595238095238096
Recall en test: 0.6536585365853659
```

[27]:
```
# Creamos la matriz de confusión
cm_gbc = confusion_matrix(y_test, y_pred_gbc)
# Creamos un DataFrame con la matriz de confusión
cm_df_gbc = pd.DataFrame(cm_gbc, index = ['No se fue', 'Se fue'], columns =␣
 ↪['No se fue', 'Se fue'])
# Mostramos la matriz de confusión
plt.figure(figsize = (8, 6))
sns.heatmap(cm_df_gbc, annot = True, fmt = 'd', cmap = 'Blues', cbar = False)
plt.title('Matriz de confusión de Gradient Boosting Classifier')
plt.xlabel('Predicción')
plt.ylabel('Realidad')
plt.show()
```



Vemos en general que este modelo con búsqueda del mejor learning rate y número de estimadores, resultó mucho mejor en sus predicciones, tanto la métrica de recall como en la de accuracy, si lo comparamos con el de regresión logística.

1.5.3 Multi-layer Perceptron Classifier

Estos modelos aunque son costos computacionalmente, representan desde su arquitectura, una forma muy personalizable de construir el modelo, buscando desde el conocimiento teórico y de los datos, el mejor modelo posible.

Ahora bien, considerando que los datos de entrada son vectores de características de 20 elementos y que la salida es binaria. Las capas de entrada y salida tienen 20 neuronas y 1 neurona respectivamente. Entonces, una serie de buenas arquitecturas base podrían ser:

- 4 capas ocultas de la forma (40, 20, 10, 5), donde se disponen primero de una cantidad de neuronas igual al doble de las dimensiones de las entradas, con la intención de captar complejidad, y luego va disminuyendo en pirámide para simplificar hacia la salida.
- 3 capas ocultas de la forma (20, 10, 5), donde desde el inicio comienza disminuyendo la complejidad.
- 2 capas ocultas de la forma (10, 5), con una mayor disminución de complejidad desde el inicio para ver si converge mejor la salida.

[28]:
```
# Importamos el módulo de Multi-Layer Perceptron
from sklearn.neural_network import MLPClassifier

# Creamos el diccionario de parámetros a ajustar
params = {
    'hidden_layer_sizes': [(40, 20, 10, 5), (20, 10, 5), (10, 5)]}

# Creamos el objeto MLPClassifier
mlp = MLPClassifier(max_iter = 1000, random_state = 42)
# Creamos el objeto GridSearchCV optimizando la métrica de 'recall', y con los␣
 ↪5 folds por defecto
grid = GridSearchCV(estimator = mlp, param_grid = params, scoring = 'recall')
# Ajustamos el objeto a los datos de entrenamiento escalados, ya que esto␣
 ↪beneficia a los MLP
grid.fit(X_train_scaled, y_train)
```

[28]:
```
GridSearchCV(estimator=MLPClassifier(max_iter=1000, random_state=42),
             param_grid={'hidden_layer_sizes': [(40, 20, 10, 5), (20, 10, 5),
                                                (10, 5)]},
             scoring='recall')
```

[29]:
```
# Vemos los resultados del GridSearchCV
grid_results = pd.DataFrame(grid.cv_results_).loc[:, ['params',␣
 ↪'mean_test_score', 'std_test_score', 'rank_test_score']]
print("El mejor parámetro de 'hidden_layer_sizes' es:", grid.best_params_)
grid_results
```

```
El mejor parámetro de 'hidden_layer_sizes' es: {'hidden_layer_sizes': (10, 5)}
```

[29]:

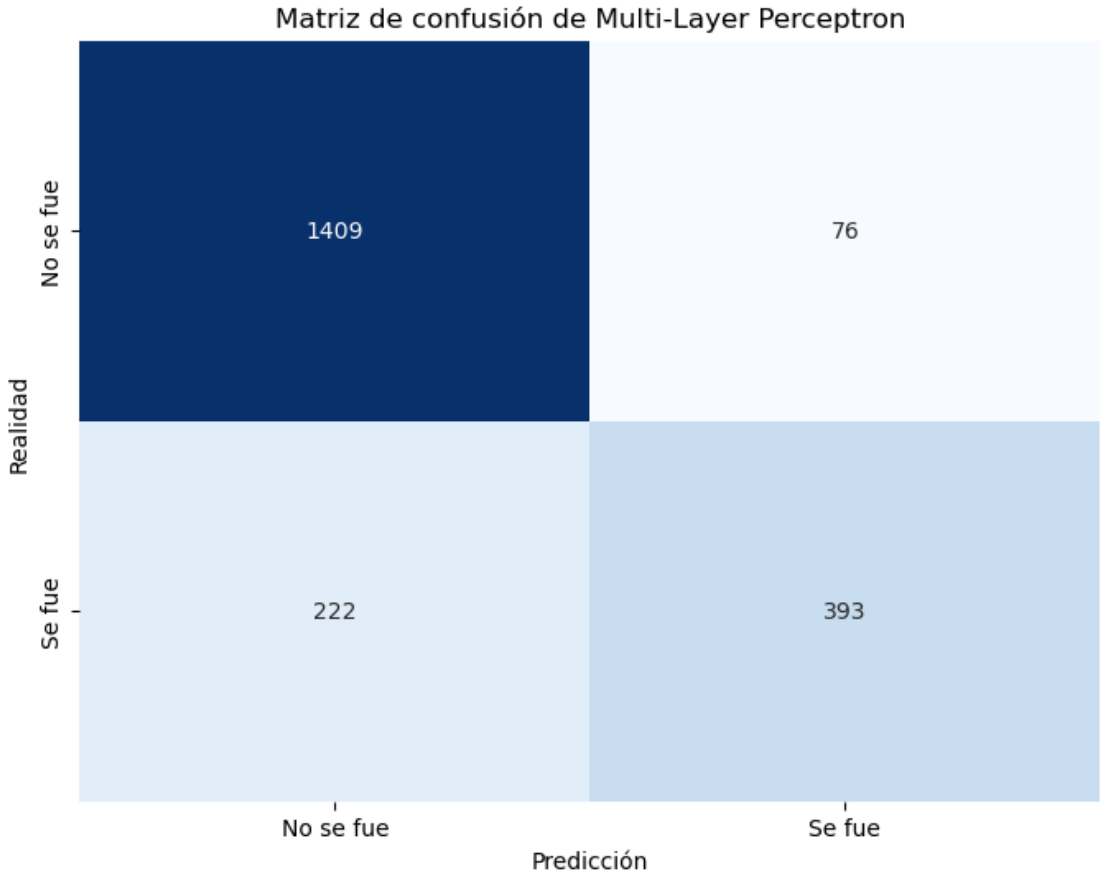
	params	mean_test_score	std_test_score	rank_test_score
0	{'hidden_layer_sizes': (40, 20, 10, 5)}	0.636179	0.035228	3
1	{'hidden_layer_sizes': (20, 10, 5)}	0.663821	0.019436	2
2	{'hidden_layer_sizes': (10, 5)}	0.669512	0.020022	1

[30]:
```
# Tomamos el mejor modelo del GridSearchCV
mlp_best = grid.best_estimator_

# Predecimos los valores de la variable dependiente para el conjunto de test
y_pred_mlp = mlp_best.predict(X_test_scaled)

# Calculamos la métrica de 'accuracy' en train para ver que no hay overfitting
y_train_pred_mlp = mlp_best.predict(X_train_scaled)
print(f"Accuracy en train: {accuracy_score(y_train, y_train_pred_mlp)}")
```

```
# Calculamos la métrica de 'accuracy' en test
print(f"Accuracy en test: {accuracy_score(y_test, y_pred_mlp)}")
# Calculamos la métrica de 'recall' en test
print(f"Recall en test: {recall_score(y_test, y_pred_mlp)}")
```

```
Accuracy en train: 0.8711751398976069
Accuracy en test: 0.8580952380952381
Recall en test: 0.6390243902439025
```

[31]:
```
# Creamos la matriz de confusión
cm_mlp = confusion_matrix(y_test, y_pred_mlp)
# Creamos un DataFrame con la matriz de confusión
cm_df_mlp = pd.DataFrame(cm_mlp, index = ['No se fue', 'Se fue'], columns =␣
 ↪['No se fue', 'Se fue'])
# Mostramos la matriz de confusión
plt.figure(figsize = (8, 6))
sns.heatmap(cm_df_mlp, annot = True, fmt = 'd', cmap = 'Blues', cbar = False)
plt.title('Matriz de confusión de Multi-Layer Perceptron')
plt.xlabel('Predicción')
plt.ylabel('Realidad')
plt.show()
```



Vemos en general que el modelo óptimo de Perceptrón unos resultados muy similares al de Gradient Boosting, sin embargo la cantidad de falsos negativos es un poco mayor, lo cual genera esa diferencia en la métrica clave de 'recall'.

1.5.4 Random Forest

Debido a una inconformidad de los resultados en torno a la métrica de **'recall'**, se opta por buscar una alternativa en este modelo, al consistir en diversos árboles de decisión que crean reglas de separación de clases y que hacen votación, suponemos podría dar buenos resultados.

[32]:
```
# Importamos el módulo de RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

# Creamos el diccionario de parámetros a ajustar
params = {
    # Número de árboles impar para evitar empates en votación
    'n_estimators': [51, 101]
    , 'max_depth': [15, 20]
    , 'min_samples_split': [2, 5]
}
# Creamos el objeto RandomForestClassifier
rfc = RandomForestClassifier(random_state = 42)

# Creamos el objeto GridSearchCV optimizando la métrica de 'recall' y con los 5␣
 ↪folds por defecto
grid = GridSearchCV(estimator = rfc, param_grid = params, scoring = 'recall')
# Ajustamos el objeto a los datos de entrenamiento, considerando que un RFC no␣
 ↪requiere normalización
grid.fit(X_train, y_train)
```

[32]:
```
GridSearchCV(estimator=RandomForestClassifier(random_state=42),
             param_grid={'max_depth': [15, 20], 'min_samples_split': [2, 5],
                         'n_estimators': [51, 101]},
             scoring='recall')
```

[33]:
```
# Creamos un DataFrame con los resultados del GridSearchCV
grid_results = pd.DataFrame(grid.cv_results_).loc[:, ['params',␣
 ↪'mean_test_score', 'std_test_score', 'rank_test_score']]
print("Los mejores parámetros son:", grid.best_params_)
grid_results
```

```
Los mejores parámetros son: {'max_depth': 20, 'min_samples_split': 2,
'n_estimators': 101}
```

[33]:
```
                                              params  mean_test_score  \
0  {'max_depth': 15, 'min_samples_split': 2, 'n_e…         0.686585
```

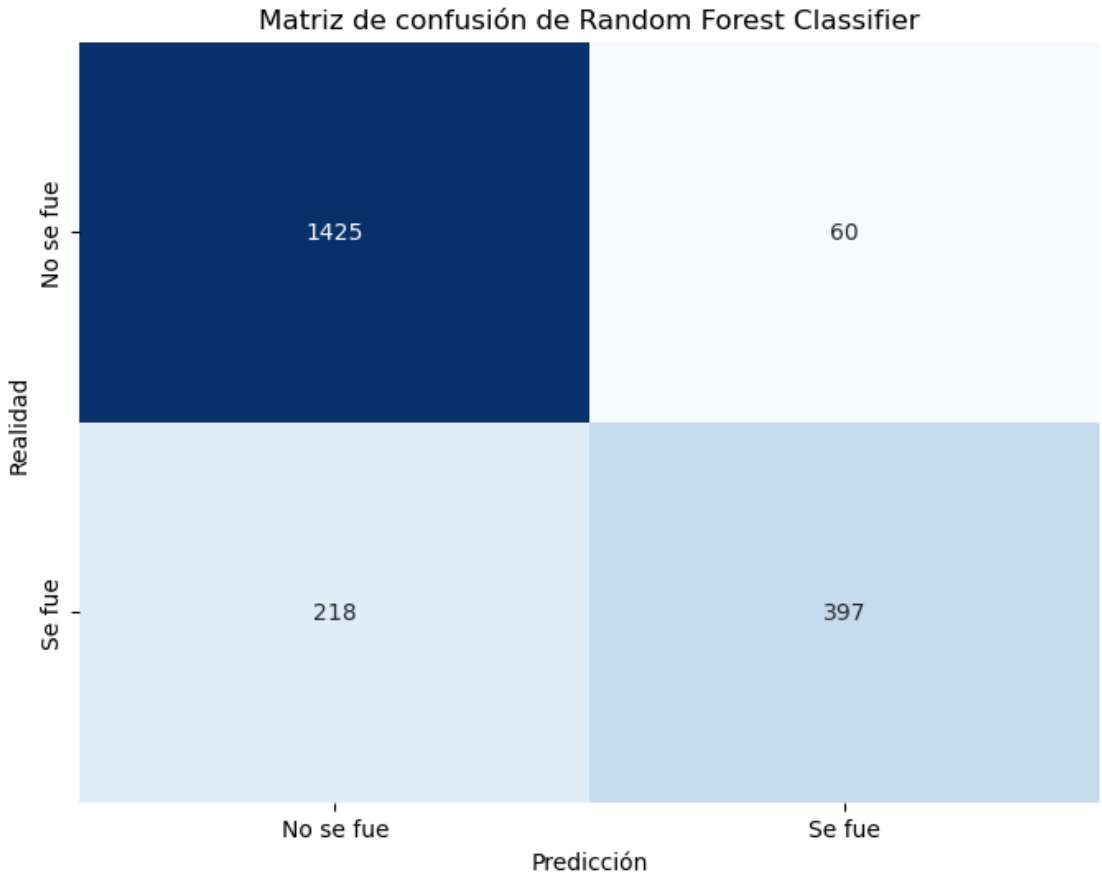
```
1  {'max_depth': 15, 'min_samples_split': 2, 'n_e…         0.687398
2  {'max_depth': 15, 'min_samples_split': 5, 'n_e…         0.683740
3  {'max_depth': 15, 'min_samples_split': 5, 'n_e…         0.684146
4  {'max_depth': 20, 'min_samples_split': 2, 'n_e…         0.681707
5  {'max_depth': 20, 'min_samples_split': 2, 'n_e…         0.687398
6  {'max_depth': 20, 'min_samples_split': 5, 'n_e…         0.684553
7  {'max_depth': 20, 'min_samples_split': 5, 'n_e…         0.684146

   std_test_score  rank_test_score
0        0.022317                3
1        0.020544                2
2        0.025921                7
3        0.025857                5
4        0.017179                8
5        0.017877                1
6        0.023046                4
7        0.024343                5
```

[34]:
```python
# Tomamos el mejor modelo del GridSearchCV
rfc_best = grid.best_estimator_

# Predecimos los valores de la variable dependiente para el conjunto de test
y_pred_rfc = rfc_best.predict(X_test)

# Calculamos la métrica de 'accuracy' en train para ver que no hay overfitting
y_train_pred_rfc = rfc_best.predict(X_train)
print(f"Accuracy en train: {accuracy_score(y_train, y_train_pred_rfc)}")
# Calculamos la métrica de 'accuracy' en test
print(f"Accuracy en test: {accuracy_score(y_test, y_pred_rfc)}")

# Calculamos la métrica de 'recall' en test
print(f"Recall en test: {recall_score(y_test, y_pred_rfc)}")
```

```
Accuracy en train: 0.9621383498035481
Accuracy en test: 0.8676190476190476
Recall en test: 0.6455284552845528
```

[35]:
```python
# Creamos la matriz de confusión
cm_rfc = confusion_matrix(y_test, y_pred_rfc)
# Creamos un DataFrame con la matriz de confusión
cm_df_rfc = pd.DataFrame(cm_rfc, index = ['No se fue', 'Se fue'], columns =␣
 ↪['No se fue', 'Se fue'])
# Mostramos la matriz de confusión
plt.figure(figsize = (8, 6))
sns.heatmap(cm_df_rfc, annot = True, fmt = 'd', cmap = 'Blues', cbar = False)
plt.title('Matriz de confusión de Random Forest Classifier')
plt.xlabel('Predicción')
```

```
plt.ylabel('Realidad')
plt.show()
```



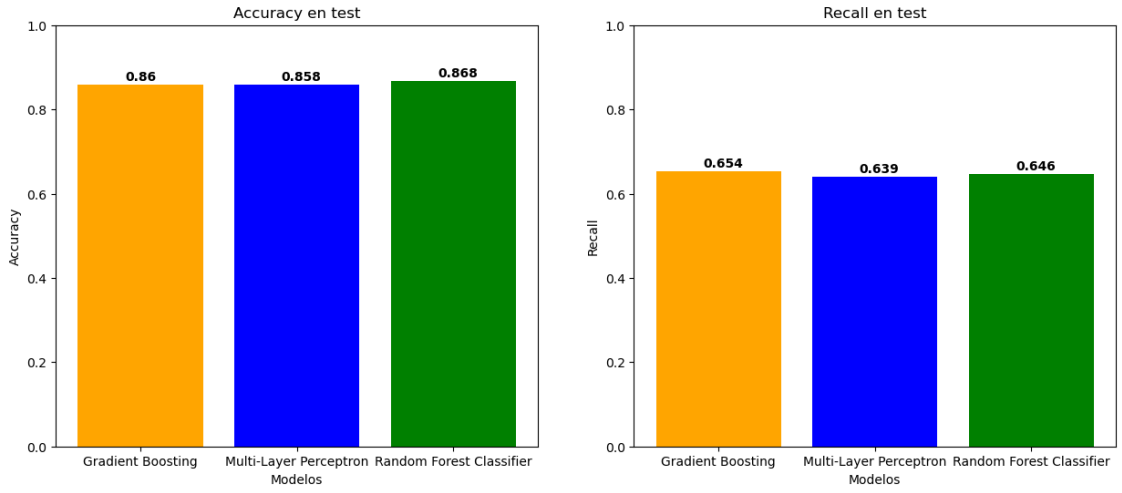
A simple vista el desempeño del Random Forest termina siendo ligeramente peor que los modelos anteriores, pero se mantiene muy cerca.

1.6 Decisión del modelo y conclusiones

Vamos a comparar los modelos presentados anteriormente, exceptuando el de regresión logística que de partida tuvo un mal desempeño en el 'recall', a parte de que no hay muchos parámetros para variar con seguridad, más allá del de regularización que no necesita explorarse demasiado, ya que el modelo no tuvo overfitting.

[36]:
```
# Comparamos los resultados de los modelos

# Comenzamos creando 2 subplots, cada uno será el gráfico de barras de los␣
  ↪modelos en torno a una métrica
fig, ax = plt.subplots(1, 2, figsize = (15, 6))
# Creamos el gráfico de barras para la métrica de 'accuracy'
```

```
ax[0].bar(['Gradient Boosting', 'Multi-Layer Perceptron', 'Random Forest␣
 ↪Classifier']
          , [accuracy_score(y_test, y_pred_gbc), accuracy_score(y_test,␣
 ↪y_pred_mlp), accuracy_score(y_test, y_pred_rfc)]
          , color = ['orange', 'blue', 'green'])
ax[0].set_title('Accuracy en test')
ax[0].set_ylabel('Accuracy')
ax[0].set_xlabel('Modelos')
ax[0].set_ylim(0, 1)
# Agregamos los valores de las métricas en cada barra
for i, v in enumerate([accuracy_score(y_test, y_pred_gbc),␣
 ↪accuracy_score(y_test, y_pred_mlp), accuracy_score(y_test, y_pred_rfc)]):
    ax[0].text(i - 0.1, v + 0.01, str(round(v, 3)), color = 'black', fontweight␣
 ↪= 'bold')

# Creamos el gráfico de barras para la métrica de 'recall'
ax[1].bar(['Gradient Boosting', 'Multi-Layer Perceptron', 'Random Forest␣
 ↪Classifier']
          , [recall_score(y_test, y_pred_gbc), recall_score(y_test,␣
 ↪y_pred_mlp), recall_score(y_test, y_pred_rfc)]
          , color = ['orange', 'blue', 'green'])
ax[1].set_title('Recall en test')
ax[1].set_ylabel('Recall')
ax[1].set_xlabel('Modelos')
ax[1].set_ylim(0, 1)
# Agregamos los valores de las métricas en cada barra
for i, v in enumerate([recall_score(y_test, y_pred_gbc), recall_score(y_test,␣
 ↪y_pred_mlp), recall_score(y_test, y_pred_rfc)]):
    ax[1].text(i - 0.1, v + 0.01, str(round(v, 3)), color = 'black', fontweight␣
 ↪= 'bold')
plt.show()
```



Se aprecia que los 3 modelos para los que se hizo la limitada búsqueda de parámetros optimizando el 'recall', estan muy parejos entre si. Vemos que el Gradient Boosting tiene ligeramente el mejor puntaje en **'recall'**, y su **'accuracy'** ocupa el segundo lugar, siguiendo de cerca al Random Forest, por estos factores, más la consideración de coste computacional, **se escoge el Gradient Boosting Classifier como el mejor modelo para esta simulación**.

[37]:
```
# Veamos los parámetros del mejor modelo Gradient Boosting Classifier
print("Los mejores parámetros del modelo Gradient Boosting Classifier son:")
for key, value in gbc_best.get_params().items():
    print(f"{key}: {value}")
```

```
Los mejores parámetros del modelo Gradient Boosting Classifier son:
ccp_alpha: 0.0
criterion: friedman_mse
init: None
learning_rate: 0.5
loss: log_loss
max_depth: 3
max_features: None
max_leaf_nodes: None
min_impurity_decrease: 0.0
min_samples_leaf: 1
min_samples_split: 2
min_weight_fraction_leaf: 0.0
n_estimators: 200
n_iter_no_change: None
random_state: 42
subsample: 1.0
tol: 0.0001
validation_fraction: 0.1
verbose: 0
warm_start: False
```

1.6.1 Conclusiones

- A pesar de que todos los modelos alcanzaron un recall alrededor del 66%, esto sigue sin tener un nivel deseado, pues significa que de cada 3 casos de empleados que potencialmente podrían abandonar la compañía, 1 no se lograría detectar para tomar medidas. Resulta en que sí se podrían tomar acciones en la mayoría de casos, pero no a un nivel suficiente para calificar de acertadas las acciones de RRHH con estos modelos.
- El accuracy de todos los modelos fue bastante bueno, se podría decir que aunque en la detección de los casos objetivo de abandono no es tan buena, tampoco resulta en un modelo que mande erroneamente a RRHH a tomar acciones con empleados que en realidad es poco probable que abandonen la empresa. Lo cual en el fondo implica ahorrar dinero y esfuerzo en las estrategias de no deserción.

- El desbalance en los datos es un factor a tomar en cuenta, en todas la matrices de confusión se hizo evidente, de manera que hay menos muestras de empleados que abandonan la compañía, y esto podría afectar el diseño de cualquier modelo, y requerir técnicas más avanzadas para compensarlo; o en su defecto recoger más muestras de estos casos.
- Quizás una búsqueda de parámetros más exhaustiva en los modelos podría llevar a una mejora considerable de la métrica clave, sin embargo esto requiere recursos computacionales más grandes, lo cual se escapa de los límites de mi máquina en concreto, y el tiempo de ejecución razonable para este notebook de simulación.