Actividad Módulo 28: Introducción a la Clasificación

1. Importación de un modelo de datos de RRHH (general_data.csv)
2. EDA Completo siguiendo los modelos anteriores
3. Feature Engineering - Estandarización, eliminación de columnas innecesarias o que no aportan al modelo, creación de nuevas columnas
4. Aplicación del modelo de regresión logística a la variable Attrition
5. Entrenamiento, evaluación y predicción de resultados del modelo
6. Análisis de sensibilidad e importancia de variables
7. Visualización de resultados del modelo de clasificación
8. Matríz de confusión y medidas del modelo

Descripción de las columnas del datset:

- Attrition - *Variable dependiente*, indica si el empleado salió o no
- BuisnessTravel - Definiciones de la frecuencia de viaje
- Department - Unidad de negocio
- DistanceFromHome - Distancia en millas de su casa
- Education - Nivel Educacional (#)
- EducationField - Campo de educación
- Employee Count - 1
- Gender - Masculino/Femenino
- JobLevel - Nivel de trabajo
- JobRole - Cargo
- MaritalStatus - Estado civil
- MonthlyIncome - Ingreso mensual
- NumCompaniesWorked - No. de empresas en las que trabajó antes
- Over18 - Si tiene más de 18 años
- PercentSalaryHike - El último alza de salario
- StandardHours - Cuántas horas estándar (#)
- StockOptionLevel - Nivel de entrega de ESOP (#)
- TotalWorkingYears - # de años trabajados en la empresa
- TrainintTimesLastYear - Número de capacitaciones que tomó el año anterior
- YearsAtCompany - Años en la empresa (#)
- YearsSinceLastPromotion - Alos desde la última promoción
- YearsWithCurrManager - Años con su manager actual

- La idea del modelo es predecir si un empleado se va a ir de la empresa o no (Attrition)

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
# 1. Importación de datos
df = pd.read_csv('general_data.csv')
df.sample(5)
```

Out[ ]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationFielc
1450	32	No	Travel_Rarely	Human Resources	13	4	Life Science:
2435	47	Yes	Travel_Frequently	Research & Development	1	3	Medica
877	46	No	Travel_Rarely	Research & Development	12	4	Technica Degree
2232	47	No	Travel_Rarely	Research & Development	23	4	Life Science:
1622	30	No	Travel_Rarely	Sales	25	3	Marketinc

5 rows × 24 columns

EDA

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4410 entries, 0 to 4409
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      4410 non-null   int64
 1   Attrition                4410 non-null   object
 2   BusinessTravel           4410 non-null   object
 3   Department               4410 non-null   object
 4   DistanceFromHome         4410 non-null   int64
 5   Education                4410 non-null   int64
 6   EducationField           4410 non-null   object
 7   EmployeeCount            4410 non-null   int64
 8   EmployeeID               4410 non-null   int64
 9   Gender                   4410 non-null   object
 10  JobLevel                 4410 non-null   int64
 11  JobRole                  4410 non-null   object
 12  MaritalStatus            4410 non-null   object
 13  MonthlyIncome            4410 non-null   int64
 14  NumCompaniesWorked       4391 non-null   float64
 15  Over18                   4410 non-null   object
 16  PercentSalaryHike        4410 non-null   int64
 17  StandardHours            4410 non-null   int64
 18  StockOptionLevel         4410 non-null   int64
 19  TotalWorkingYears        4401 non-null   float64
 20  TrainingTimesLastYear    4410 non-null   int64
 21  YearsAtCompany           4410 non-null   int64
 22  YearsSinceLastPromotion  4410 non-null   int64
 23  YearsWithCurrManager     4410 non-null   int64
dtypes: float64(2), int64(14), object(8)
memory usage: 827.0+ KB
```

In [ ]:
```
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%
Age	4410.0	36.923810	9.133301	18.0	30.00	36.0	43.00
DistanceFromHome	4410.0	9.192517	8.105026	1.0	2.00	7.0	14.00
Education	4410.0	2.912925	1.023933	1.0	2.00	3.0	4.00
EmployeeCount	4410.0	1.000000	0.000000	1.0	1.00	1.0	1.00
EmployeeID	4410.0	2205.500000	1273.201673	1.0	1103.25	2205.5	3307.75
JobLevel	4410.0	2.063946	1.106689	1.0	1.00	2.0	3.00
MonthlyIncome	4410.0	65029.312925	47068.888559	10090.0	29110.00	49190.0	83800.00
NumCompaniesWorked	4391.0	2.694830	2.498887	0.0	1.00	2.0	4.00
PercentSalaryHike	4410.0	15.209524	3.659108	11.0	12.00	14.0	18.00
StandardHours	4410.0	8.000000	0.000000	8.0	8.00	8.0	8.00
StockOptionLevel	4410.0	0.793878	0.851883	0.0	0.00	1.0	1.00
TotalWorkingYears	4401.0	11.279936	7.782222	0.0	6.00	10.0	15.00
TrainingTimesLastYear	4410.0	2.799320	1.288978	0.0	2.00	3.0	3.00
YearsAtCompany	4410.0	7.008163	6.125135	0.0	3.00	5.0	9.00
YearsSinceLastPromotion	4410.0	2.187755	3.221699	0.0	0.00	1.0	3.00
YearsWithCurrManager	4410.0	4.123129	3.567327	0.0	2.00	3.0	7.00

In [ ]:
```
df.shape
```

Out[ ]:
```
(4410, 24)
```

In [ ]:
```
# Número de valores diferentes
df.nunique()
```

Out[ ]:
```
Age                         43
Attrition                    2
BusinessTravel               3
Department                   3
DistanceFromHome            29
Education                    5
EducationField               6
EmployeeCount                1
EmployeeID                4410
Gender                       2
JobLevel                     5
JobRole                      9
MaritalStatus                3
MonthlyIncome             1349
NumCompaniesWorked          10
Over18                       1
PercentSalaryHike           15
StandardHours                1
StockOptionLevel             4
TotalWorkingYears           40
TrainingTimesLastYear        7
YearsAtCompany              37
YearsSinceLastPromotion     16
YearsWithCurrManager        18
dtype: int64
```

Insights

- Será necesaria imputación de datos para algunos registros en NumCompaniesWorked y TotalWorkedYears
- Los campos EmployeeCount, Over18 y StandardHours se pueden eliminar; no van a aportar nada al modelo por presentar el mismo valor para todos los registros
- EmployeeID es el id del empleado, no aporta información al modelo, se puede eliminar

In [ ]:
```
# Validación de datos

# Datos nulos
df.isnull().sum()
```

Out[ ]:
```
Age                        0
Attrition                  0
BusinessTravel             0
Department                 0
DistanceFromHome           0
Education                  0
EducationField             0
EmployeeCount              0
EmployeeID                 0
Gender                     0
JobLevel                   0
JobRole                    0
MaritalStatus              0
MonthlyIncome              0
NumCompaniesWorked        19
Over18                     0
PercentSalaryHike          0
StandardHours              0
StockOptionLevel           0
TotalWorkingYears          9
TrainingTimesLastYear      0
YearsAtCompany             0
YearsSinceLastPromotion    0
YearsWithCurrManager       0
dtype: int64
```

Imputación de Datos

- Se decidió tomar tanto para NumCompaniesWorked como apra TotalWorkinYears, el número promedio para imputar en los registros donde faltan estos campos

In [ ]:
```
# Imputación de datos
# Reemplazar los registros con nulos por el promedio general
df['NumCompaniesWorked'] = df['NumCompaniesWorked'].fillna(int(np.mean(a=df['NumCom
df['TotalWorkingYears'] = df['TotalWorkingYears'].fillna(int(np.mean(a=df['TotalWor
```

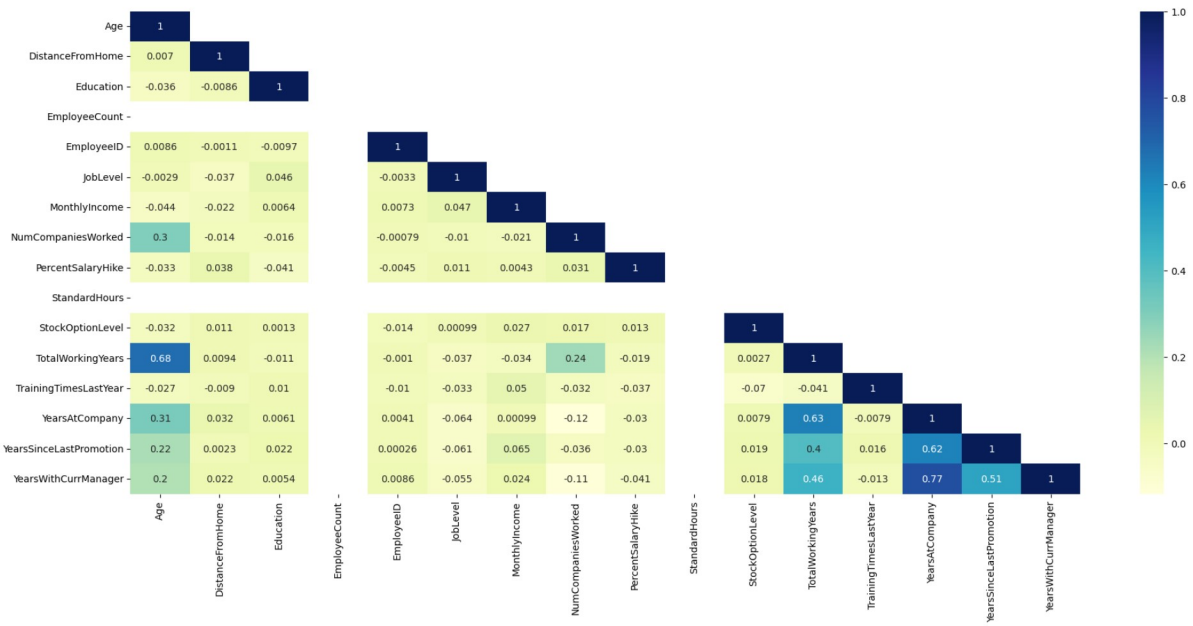
In [ ]:
```
print(f'Total number of null spaces {df.isnull().sum().sum()}')
```

```
Total number of null spaces 0
```

Análisis de Correlación

In [ ]:
```
plt.figure(figsize=(22,9))
corr_df = df.corr(method='pearson')
df_lt = corr_df.where(np.tril(np.ones(corr_df.shape)).astype(bool))
hmp = sns.heatmap(data=df_lt, cmap='YlGnBu', annot=True)
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_18088\3473999359.py:2: FutureWarning: Th
e default value of numeric_only in DataFrame.corr is deprecated. In a future versio
n, it will default to False. Select only valid columns or specify the value of nume
ric_only to silence this warning.
  corr_df = df.corr(method='pearson')
```



Insights

- No existe multicolinealidad, pero existen relaciones curiosas entre los campos, TotalWorkingYears - YearsAtCompany - YearsWithCurrManager

---

Gráfico por Distribución

In [ ]:
```
numerics = ['int16', 'int32', 'int64', 'float64']
df.select_dtypes(include=numerics).sample(5)
```

Out[ ]:

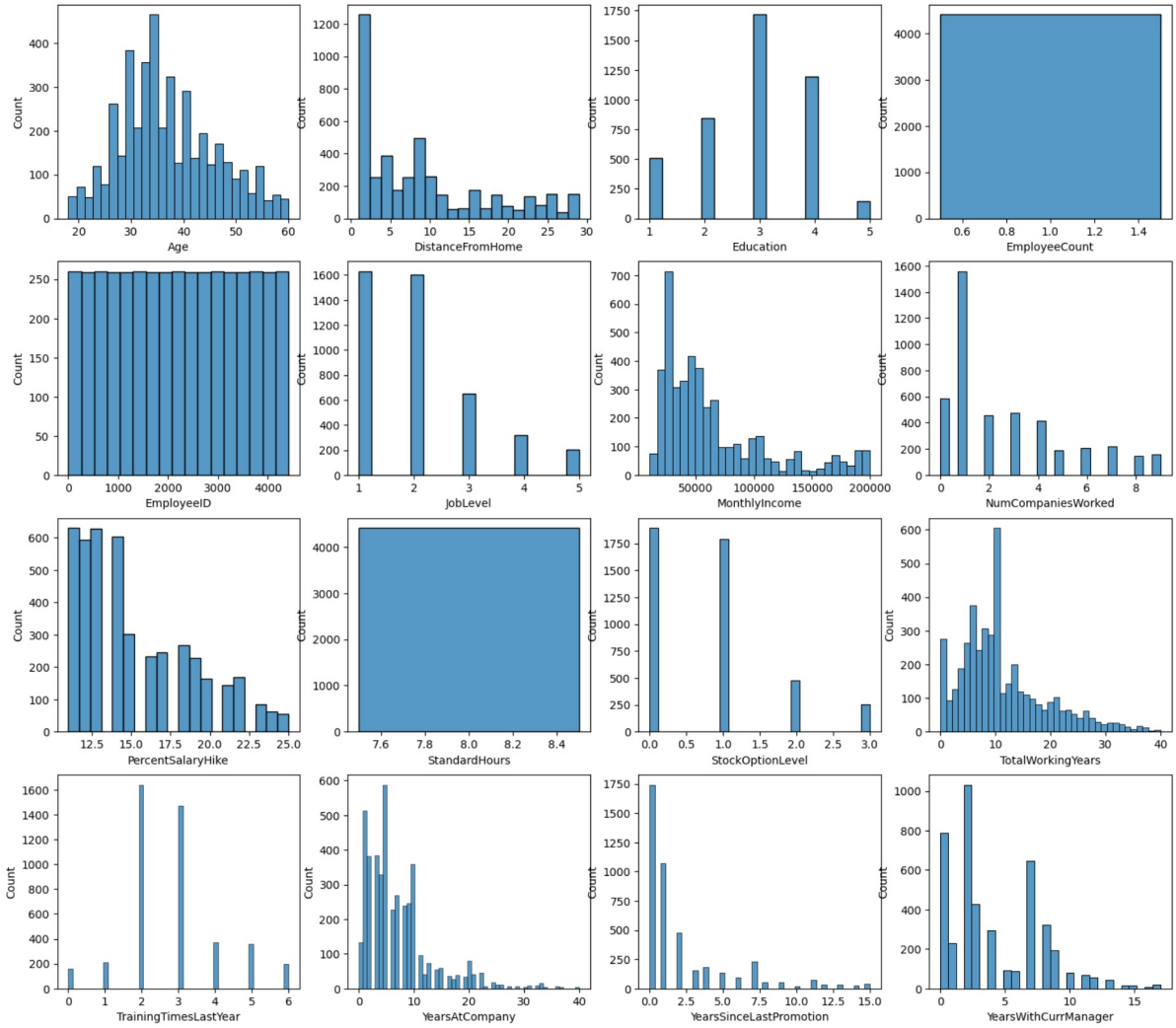
	Age	DistanceFromHome	Education	EmployeeCount	EmployeeID	JobLevel	MonthlyIncome
3062	24	1	3	1	3063	5	2897(
2150	58	27	3	1	2151	2	2187(
2889	36	1	3	1	2890	2	2305(
2342	35	4	3	1	2343	2	7403(
1455	40	2	3	1	1456	1	2718(

In [ ]:
```
# Se seleccional as columnas de tipo numérico
cols_num = df.columns[[np.issubdtype(dt, np.number) for dt in df.dtypes]]
cols_num
```

Out[ ]:
```
Index(['Age', 'DistanceFromHome', 'Education', 'EmployeeCount', 'EmployeeID',
       'JobLevel', 'MonthlyIncome', 'NumCompaniesWorked', 'PercentSalaryHike',
       'StandardHours', 'StockOptionLevel', 'TotalWorkingYears',
       'TrainingTimesLastYear', 'YearsAtCompany', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [ ]:
```
# Gráfico exploratoria de todas las columnas numéricas (16 - cols_num)

fig, ax = plt.subplots(nrows=4, ncols=4, figsize=(18,16))
for i, column in enumerate(cols_num):
    sns.histplot(data=df, x=column, ax=ax[i//4, i%4], kde=False)
```



Insights

- Las columas YearsAtCompany, TotalWorkingYears, MonthlyIncome parecen estar sesgadas
- El rango de la variable MonthlyIncome se podría estandaizar (está en cientos de miles a comparación de otras que estan en decenas o unidades)
- Las columnas StandardHours, EmployeeId no aportan ningún valor al modelo, se pueden eliminar
- EmployeeCount siendo siempre uno seguramente es utilizado para hacer cuentas de empleados y no usar el EmployeeID u otros valores

---

Análisis Bivriado

In [ ]:
```
df.groupby(['TrainingTimesLastYear']).mean().T
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_18088\3805342632.py:1: FutureWarning: Th
e default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select
only columns which should be valid for the function.
  df.groupby(['TrainingTimesLastYear']).mean().T
```

Out[ ]:

TrainingTimesLastYear	0	1	2	3	4	
Age	37.222222	38.492958	37.208410	36.576375	36.528455	
DistanceFromHome	9.833333	10.985915	8.804388	9.274949	9.471545	
Education	2.796296	3.056338	2.839122	2.979633	3.024390	
EmployeeCount	1.000000	1.000000	1.000000	1.000000	1.000000	
EmployeeID	2183.611111	2309.929577	2205.654479	2200.089613	2235.268293	2
JobLevel	1.907407	2.042254	2.102377	2.112016	1.983740	
MonthlyIncome	60866.666667	70142.394366	63903.071298	64080.040733	56839.024390	72
NumCompaniesWorked	3.425926	2.718310	2.630104	2.719620	2.940379	
PercentSalaryHike	15.500000	15.408451	15.341865	15.002037	15.821138	
StandardHours	8.000000	8.000000	8.000000	8.000000	8.000000	
StockOptionLevel	0.981481	0.845070	0.842779	0.733198	0.886179	
TotalWorkingYears	12.055556	11.760563	11.497867	11.272912	10.910569	
YearsAtCompany	7.462963	7.492958	7.073126	6.828921	7.056911	
YearsSinceLastPromotion	1.814815	2.323944	2.117002	2.264766	2.479675	
YearsWithCurrManager	4.759259	4.507042	4.062157	4.079430	4.040650	

In [ ]:
```
df.groupby(['JobLevel']).mean().T
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_18088\1067281539.py:1: FutureWarning: Th
e default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select
only columns which should be valid for the function.
  df.groupby(['JobLevel']).mean().T
```

Out[ ]:

JobLevel	1	2	3	4	5
Age	36.775322	37.367041	36.266055	36.377358	37.579710
DistanceFromHome	9.381215	9.415730	8.784404	8.273585	8.681159
Education	2.841621	2.945693	2.940367	3.047170	2.927536
EmployeeCount	1.000000	1.000000	1.000000	1.000000	1.000000
EmployeeID	2189.810313	2237.709738	2196.220183	2142.886792	2205.202899
MonthlyIncome	62677.421731	65506.479401	63545.321101	77940.754717	64698.405797
NumCompaniesWorked	2.669122	2.742821	2.772171	2.443396	2.603865
PercentSalaryHike	14.963168	15.475655	15.412844	14.783019	15.101449
StandardHours	8.000000	8.000000	8.000000	8.000000	8.000000
StockOptionLevel	0.823204	0.758427	0.793578	0.716981	0.956522
TotalWorkingYears	11.348066	11.647940	10.723242	10.990566	10.086957
TrainingTimesLastYear	2.882136	2.769663	2.651376	2.801887	2.840580
YearsAtCompany	7.338858	7.106742	6.426606	7.169811	5.231884
YearsSinceLastPromotion	2.418048	2.155431	1.876147	2.245283	1.521739
YearsWithCurrManager	4.384899	3.998127	3.926606	4.481132	3.101449

In [ ]:
```
df.groupby(['StockOptionLevel']).mean()[['Age', 'DistanceFromHome', 'MonthlyIncome'
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_18088\2000770076.py:1: FutureWarning: Th
e default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select
only columns which should be valid for the function.
  df.groupby(['StockOptionLevel']).mean()[['Age', 'DistanceFromHome', 'MonthlyIncom
e', 'TotalWorkingYears']]
```

Out[ ]:

	Age	DistanceFromHome	MonthlyIncome	TotalWorkingYears
StockOptionLevel				
0	36.939778	9.169572	63077.606973	11.159007
1	37.278523	9.156040	66549.630872	11.378076
2	36.354430	9.025316	66898.924051	11.810127
3	35.376471	9.929412	65382.470588	10.494118

Feature Engineering

In [ ]:
```
# Genera una copia del df
df2 = df.copy()
```

In [ ]:
```
# ELIMINAR COLUMNAS NO NECESARIAS
df2.drop(columns=['EmployeeID', 'StandardHours', 'Over18', 'EmployeeCount'], inplac
df2.sample(3)
```

Out[ ]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationFielc
722	31	Yes	Travel_Frequently	Research & Development	2	2	Medica
228	37	No	Travel_Rarely	Sales	9	4	Marketinc
2949	25	No	Non-Travel	Research & Development	7	4	Medica

In [ ]:
```
# CAMBIAR COLUMNAS OBJECT POR CATEGORY
df2['BusinessTravel'] = df2['BusinessTravel'].astype('category')
df2['Department'] = df2['Department'].astype('category')
df2['EducationField'] = df2['EducationField'].astype('category')
df2['Gender'] = df2['Gender'].astype('category')

# Cambiar columnas NumCompaniesWorked a entero
df2['NumCompaniesWorked'] = df2['NumCompaniesWorked'].fillna(value=0).astype('int')
```

In [ ]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4410 entries, 0 to 4409
Data columns (total 20 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       4410 non-null   int64
 1   Attrition                 4410 non-null   object
 2   BusinessTravel            4410 non-null   category
 3   Department                4410 non-null   category
 4   DistanceFromHome          4410 non-null   int64
 5   Education                 4410 non-null   int64
 6   EducationField            4410 non-null   category
 7   Gender                    4410 non-null   category
 8   JobLevel                  4410 non-null   int64
 9   JobRole                   4410 non-null   object
 10  MaritalStatus             4410 non-null   object
 11  MonthlyIncome             4410 non-null   int64
 12  NumCompaniesWorked        4410 non-null   int32
 13  PercentSalaryHike         4410 non-null   int64
 14  StockOptionLevel          4410 non-null   int64
 15  TotalWorkingYears         4410 non-null   float64
 16  TrainingTimesLastYear     4410 non-null   int64
 17  YearsAtCompany            4410 non-null   int64
 18  YearsSinceLastPromotion   4410 non-null   int64
 19  YearsWithCurrManager      4410 non-null   int64
dtypes: category(4), float64(1), int32(1), int64(11), object(3)
memory usage: 552.0+ KB
```

Balanceo de Clases

In [ ]:
```
# BALANCEO DE CLASES - PASAR LAS CLASES QUE TENGAN REPRESENTACIÓN EN MENOS DEL 5% D
print(df2['EducationField'].value_counts(normalize=True),'\n')
print(df2['BusinessTravel'].value_counts(normalize=True), '\n')
print(df2['Department'].value_counts(normalize=True), '\n')
print(df2['MaritalStatus'].value_counts(normalize=True), '\n')
print(df2['JobRole'].value_counts(normalize=True), '\n')
print(df2['Gender'].value_counts(normalize=True), '\n')
```

```
Life Sciences       0.412245
Medical             0.315646
Marketing           0.108163
Technical Degree    0.089796
Other               0.055782
Human Resources     0.018367
Name: EducationField, dtype: float64

Travel_Rarely        0.709524
Travel_Frequently    0.188435
Non-Travel           0.102041
Name: BusinessTravel, dtype: float64

Research & Development    0.653741
Sales                     0.303401
Human Resources           0.042857
Name: Department, dtype: float64

Married     0.457823
Single      0.319728
Divorced    0.222449
Name: MaritalStatus, dtype: float64

Sales Executive              0.221769
Research Scientist           0.198639
Laboratory Technician        0.176190
Manufacturing Director       0.098639
Healthcare Representative    0.089116
Manager                      0.069388
Sales Representative         0.056463
Research Director            0.054422
Human Resources              0.035374
Name: JobRole, dtype: float64

Male      0.6
Female    0.4
Name: Gender, dtype: float64
```

In [ ]:
```
# EducationField - Agregar Human Resources a Other
df2['EducationField'] = df2['EducationField'].replace(to_replace={'Human Resources'
```

In [ ]:
```
print(df2['EducationField'].value_counts(normalize=True))
```

```
Life Sciences       0.412245
Medical             0.315646
Marketing           0.108163
Technical Degree    0.089796
Other               0.074150
Name: EducationField, dtype: float64
```

In [ ]:
```
# Cambio de tipo de columnas categóricas - Automático
# Se decidió utilizar variables dummy para BuisenessTravel, Department, EducationFi
# Para Gender y Attrition, se decidió utilizar reemplazar los valores por 1/0

df2['Gender'] = df2['Gender'].replace(to_replace={'Male':0, 'Female':1})
df2['Attrition'] = df2['Attrition'].replace(to_replace={'Yes':1, 'No':0})

df2 = pd.get_dummies(data=df2)

df2
```

Out[ ]:

	Age	Attrition	DistanceFromHome	Education	JobLevel	MonthlyIncome	NumCompaniesW
0	51	0	6	2	1	131160	
1	31	1	10	1	1	41890	
2	32	0	17	4	4	193280	
3	38	0	2	5	3	83210	
4	32	0	10	1	1	23420	
...	...	...	...	...	...	...	
4405	42	0	5	4	1	60290	
4406	29	0	2	4	1	26790	
4407	25	0	25	2	2	37020	
4408	42	0	18	2	1	23980	
4409	40	0	28	3	2	54680	

4410 rows × 39 columns

In [ ]:
```
# Estandarizar las columnas que lo necesiten (Income)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df2['MonthlyIncome_scale'] = scaler.fit_transform(df2[['MonthlyIncome']])
df2.sample(3)
```

Out[ ]:

	Age	Attrition	DistanceFromHome	Education	JobLevel	MonthlyIncome	NumCompaniesW
3151	32	1	7	2	2	32980	
1242	33	0	3	3	1	43730	
3879	20	0	1	3	1	47770	

3 rows × 40 columns

In [ ]:
```
# Opcional - Eliminar la variable que se acaba de estandarizar
df2.drop(columns=['MonthlyIncome'], inplace=True)
```

Data Split

In [ ]:
```
from sklearn.model_selection import train_test_split

x = df2.drop(columns=['Attrition'])
y = df2['Attrition']

# En este caso no es necesario, pero habría que convertir variables categóricas en
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.30, random_sta
```

Regresión Logística

In [ ]:
```
print(df2['Attrition'].value_counts(normalize=True))
```

```
0    0.838776
1    0.161224
Name: Attrition, dtype: float64
```

Como se puede ver, en el dataset hay una pequeña cantidad de respuestas posivitas a comparación de la cantidad de respuestas negativas

https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/

https://machinelearningmastery.com/cost-sensitive-logistic-regression/

In [ ]:
```
from sklearn.linear_model import LogisticRegression

# Creación del modelo
model = LogisticRegression(solver='liblinear')

# Entrenamiento del modelo
model.fit(x_train, y_train)

# Hacer predicción en con x_test
y_predict = model.predict(x_test)
```

In [ ]:
```
# Build a dataset with original and predicted
z = pd.DataFrame()
z['Attrition real'] = y_test
z['Attrition prediction'] = y_predict
z.sample(20)
```

Out[ ]:

	Attrition real	Attrition prediction
453	1	1
2639	0	0
1918	1	0
1501	0	0
3661	0	0
203	0	0
1409	0	0
4069	0	0
2273	0	0
1453	0	0
28	1	0
1296	0	0
3213	0	0
2010	1	0
4195	0	0
1377	0	0
357	0	0
3987	0	0
1232	0	0
2611	0	0

In [ ]:
```
# Tenera los coeficientes de la ecuación
coef_df = pd.DataFrame(model.coef_)
coef_df.columns = x.columns

# Incluir el intercept
coef_df['Intercept'] = model.intercept_
coef_df.T
```

Out[ ]:

	0
Age	-0.030033
DistanceFromHome	-0.000333
Education	-0.041780
JobLevel	-0.033312
NumCompaniesWorked	0.132107
PercentSalaryHike	0.022312
StockOptionLevel	-0.076700
TotalWorkingYears	-0.065211
TrainingTimesLastYear	-0.129351
YearsAtCompany	0.027393
YearsSinceLastPromotion	0.117369
YearsWithCurrManager	-0.148730
BusinessTravel_Non-Travel	-0.686801
BusinessTravel_Travel_Frequently	0.684039
BusinessTravel_Travel_Rarely	-0.020373
Department_Human Resources	0.579358
Department_Research & Development	-0.228419
Department_Sales	-0.374075
EducationField_Life Sciences	0.110802
EducationField_Marketing	-0.135779
EducationField_Medical	0.072018
EducationField_Other	0.035621
EducationField_Technical Degree	-0.105799
Gender_1	-0.071894
Gender_0	0.048759
JobRole_Healthcare Representative	-0.024558
JobRole_Human Resources	-0.408602
JobRole_Laboratory Technician	0.232202
JobRole_Manager	-0.288983
JobRole_Manufacturing Director	-0.470102
JobRole_Research Director	0.457353
JobRole_Research Scientist	0.227239
JobRole_Sales Executive	0.226191

	0
JobRole_Sales Representative	0.026123
MaritalStatus_Divorced	-0.406031
MaritalStatus_Married	-0.255328
MaritalStatus_Single	0.638224
MonthlyIncome_scale	-0.034717
Intercept	-0.023135

In [ ]:
```
print(df['JobRole'].value_counts())
```

```
Sales Executive              978
Research Scientist           876
Laboratory Technician        777
Manufacturing Director       435
Healthcare Representative    393
Manager                      306
Sales Representative         249
Research Director            240
Human Resources              156
Name: JobRole, dtype: int64
```

Análisis de sensibilidad

Algunas de las variables que más influencia tienen sobre la variable de interés Attrition y el efecto que tienen sobre la variable:

- BuisnessTravel - Conforme más viaja la persona, más probable es que deje la compañía
- Department - Cuando la persona trabaja en el departamento de recursos humanos, es más probable que dejen la compañía y cuando trabajan en el área de ventas es menos probable que dejen la compañía
- Age - Entre mayor sea el trabajador, menor la probabilidad de que renuncie
- JobRole - Ser director de manufactura o de recursos humanos disminuye la probabilidad de que la persona renuncie. Por el otro lado, cuando son directores de investigación, es más probable que renuncien
- MaritalStatus - Cuando el empleado es soltero es más probable que se vaya de la empresa
- Monthly Income - Conforme el salario del trabajador aumenta, es menos probable que se vaya de la empresa
- NumCompaniesWorked - Conforme el empleado haya trabajado en más empresas, es un poco más probable que se vaya de la empresa
- YearsSinceLastPromotion - Conforme hayan pasado más años desde su ultima promoción de puesto, es un poco más probable que el empleado deje la empresa
- YearWithCurrManager - Conforme la persona tenga más años con el mismo manager, es un poco menos probable uqe deje la empresa
- Otras variables que no tuvo efecto sobre attrition:
  - DistanceFromHome
  - StockOption
  - TotalWorkingYears
  - YearsAtCompany
  - EducationField
  - Gender

Matriz de confusión

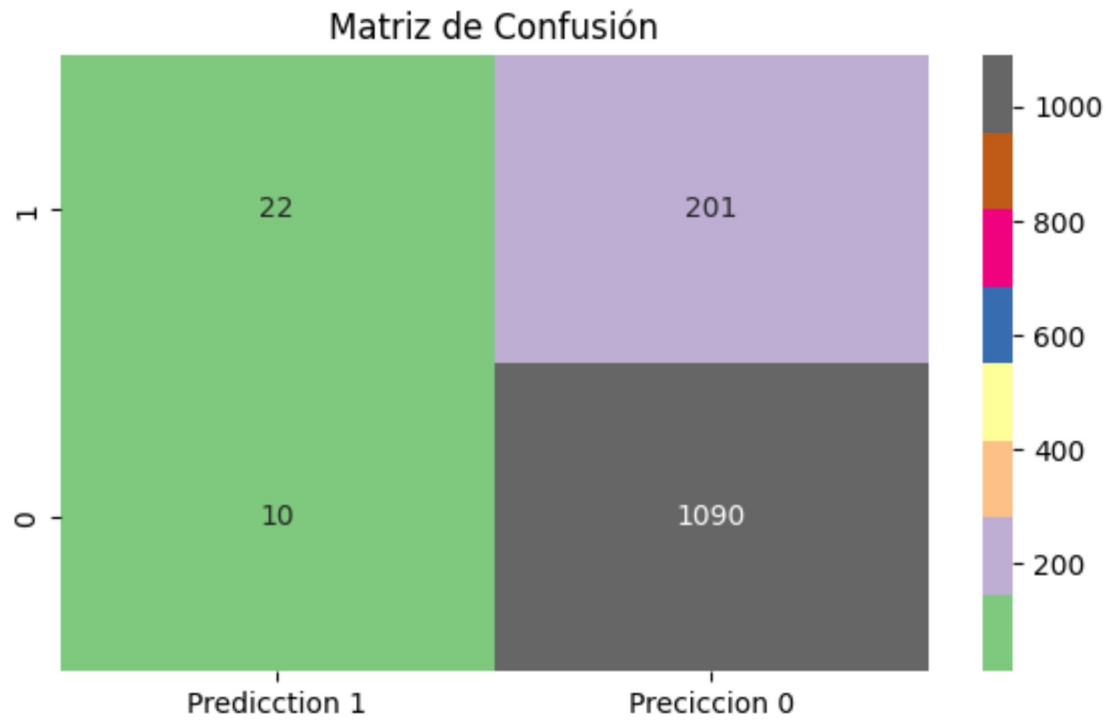
In [ ]:
```
from sklearn import metrics

cm = metrics.confusion_matrix(y_test, y_predict, labels=[1,0])

df_cm = pd.DataFrame(cm, index=[i for i in ['1', '0']],
                     columns=[i for i in ['Predicction 1', 'Preciccion 0']])

plt.figure(figsize=(7,4))
sns.heatmap(df_cm, annot=True, cmap='Accent', fmt='d').set_title('Matriz de Confusi
```

Out[ ]:
```
Text(0.5, 1.0, 'Matriz de Confusión')
```



In [ ]:
```
# Medidas del modelo
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score

model_score = accuracy_score(y_test, y_predict)
recall_score = recall_score(y_test, y_predict)
precision_score = precision_score(y_test, y_predict)
f1_score = f1_score(y_test, y_predict)

print('Medidas del modelo')
print('----------------------')
print('Accuracy = {0:0.2f}'.format(model_score))
print('Precision = {0:0.2f}'.format(precision_score))
print('Recall = {0:0.2f}'.format(recall_score))
print('F1 Score = {0:0.2f}'.format(f1_score))
print('Accuracy = {0:0.2f}'.format(roc_auc_score(y_test, y_predict)))
```

```
Medidas del modelo
----------------------
Accuracy = 0.84
Precision = 0.69
Recall = 0.10
F1 Score = 0.17
Accuracy = 0.54
```

In [ ]: