

## Resolución de prueba técnica (KPI Churn = Tasa de Abandono) de Eduardo Martinez para Deloitte

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True)

### PROYECTO / PRUEBA TÉCNICA

#### CARGA DE LIBRERIAS

Haz doble clic (o pulsa Intro) para editar

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

#### CARGA DE LOS DATOS

- Para la realización de este ejercicio voy a usar el documento llamado "AbandonoEmpleados.csv"

```
df = pd.read_csv('AbandonoEmpleados.csv', sep = ';', index_col= 'id', na_values='#N/D')
```

df

	edad	abandono	viajes	departamento	distancia_casa	educacion	carrera	empleados	satisfaccion_entorno	sexo
id										
1	41	Yes	Travel_Rarely	Sales	1	Universitaria	Life Sciences	1	Media	3.0
2	49	No	Travel_Frequently	Research & Development	8	Secundaria	Life Sciences	1	Alta	2.0
4	37	Yes	Travel_Rarely	Research & Development	2	Secundaria	Other	1	Muy_Alta	2.0
5	33	No	Travel_Frequently	Research & Development	3	Universitaria	Life Sciences	1	Muy_Alta	3.0
7	27	No	Travel_Rarely	Research & Development	2	Universitaria	Medical	1	Baja	3.0
...	...	...	...	...	...	...	...	...	...	...
2061	36	No	Travel_Frequently	Research & Development	23	Master	Medical	1	Alta	4.0
2062	39	No	Travel_Rarely	Research & Development	6	Secundaria	Medical	1	Muy_Alta	2.0
2064	27	No	Travel_Rarely	Research & Development	4	Master	Life Sciences	1	Media	4.0
2065	49	No	Travel_Frequently	Sales	2	Secundaria	Medical	1	Muy_Alta	NaN
2068	34	No	Travel_Rarely	Research & Development	8	NaN	Medical	1	Media	4.0

1470 rows × 31 columns

### APARTADO BUSINESS ANALYTICS

```
df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
Index: 1470 entries, 1 to 2068
Data columns (total 31 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   edad                  1470 non-null   int64
 1   abandono              1470 non-null   object
 2   viajes                1470 non-null   object
 3   departamento          1470 non-null   object
 4   distancia_casa        1470 non-null   int64
 5   educacion             1369 non-null   object
 6   carrera               1470 non-null   object
 7   empleados             1470 non-null   int64
 8   satisfaccion_entorno  1470 non-null   object
 9   sexo                  1271 non-null   float64
10   implicacion           1452 non-null   object
11   nivel_laboral         1470 non-null   int64
12   puesto                1470 non-null   object
13   satisfaccion_trabajo  1394 non-null   object
14   estado_civil          1470 non-null   object
15   salario_mes           1470 non-null   int64
16   num_empresas_anteriores 1470 non-null   int64
17   mayor_edad            1470 non-null   object
18   horas_extra           1470 non-null   object
19   incremento_salario_porcentaje 1470 non-null   int64
20   evaluacion            1470 non-null   object
21   satisfaccion_companeros 1470 non-null   object
22   horas_quincena        1470 non-null   int64
23   nivel_acciones        1470 non-null   int64
24   anos_experiencia      1470 non-null   int64
25   num_formaciones_ult_ano 1470 non-null   int64
26   conciliacion          459 non-null    object
27   anos_compania         1470 non-null   int64
28   anos_en_puesto        232 non-null    float64
29   anos_desde_ult_promocion 1470 non-null   int64
30   anos_con_manager_actual 1470 non-null   int64
dtypes: float64(2), int64(14), object(15)
memory usage: 367.5+ KB

```

## ✓ APARTADO EDA (ANÁLISIS EXPLORATORIO DE LOS DATOS)

### ANÁLISIS DE NULOS

```
df.isna().sum().sort_values(ascending = False)
```

```

anos_en_puesto      1238
conciliacion        1011
sexo                 199
educacion            101
satisfaccion_trabajo  76
implicacion          18
edad                  0
nivel_acciones       0
evaluacion           0
satisfaccion_companeros 0
horas_quincena       0
anos_experiencia     0
horas_extra          0
num_formaciones_ult_ano 0
anos_compania        0
anos_desde_ult_promocion 0
incremento_salario_porcentaje 0
salario_mes          0
mayor_edad           0
num_empresas_anteriores 0
abandono             0
estado_civil         0
puesto              0
nivel_laboral        0
satisfaccion_entorno 0
empleados            0
carrera              0
distancia_casa       0
departamento        0
viajes               0
anos_con_manager_actual 0
dtype: int64

```

## ✓ Conclusiones:

- anos\_en\_puesto y conciliacion ==> tienen demasiados nulos ==> Elimino Variables
- sexo, educacion, satisfaccion\_trabajo e implicacion ==> Imputarlas tras EDA

```
df.drop(columns = ['anos_en_puesto','conciliacion'], inplace = True)
df
```

id	edad	abandono	viajes	departamento	distancia_casa	educacion	carrera	empleados	satisfaccion_entorno	sexo
1	41	Yes	Travel_Rarely	Sales	1	Universitaria	Life Sciences	1	Media	3.0
2	49	No	Travel_Frequently	Research & Development	8	Secundaria	Life Sciences	1	Alta	2.0
4	37	Yes	Travel_Rarely	Research & Development	2	Secundaria	Other	1	Muy_Alta	2.0
5	33	No	Travel_Frequently	Research & Development	3	Universitaria	Life Sciences	1	Muy_Alta	3.0
7	27	No	Travel_Rarely	Research & Development	2	Universitaria	Medical	1	Baja	3.0
...	...	...	...	...	...	...	...	...	...	...
2061	36	No	Travel_Frequently	Research & Development	23	Master	Medical	1	Alta	4.0
2062	39	No	Travel_Rarely	Research & Development	6	Secundaria	Medical	1	Muy_Alta	2.0
2064	27	No	Travel_Rarely	Research & Development	4	Master	Life Sciences	1	Media	4.0
2065	49	No	Travel_Frequently	Sales	2	Secundaria	Medical	1	Muy_Alta	NaN
2068	34	No	Travel_Rarely	Research & Development	8	NaN	Medical	1	Media	4.0

1470 rows × 29 columns

## ✓ EDA VARIABLES CATEGÓRICAS

```
def graficos_eda_categoricos(cat):

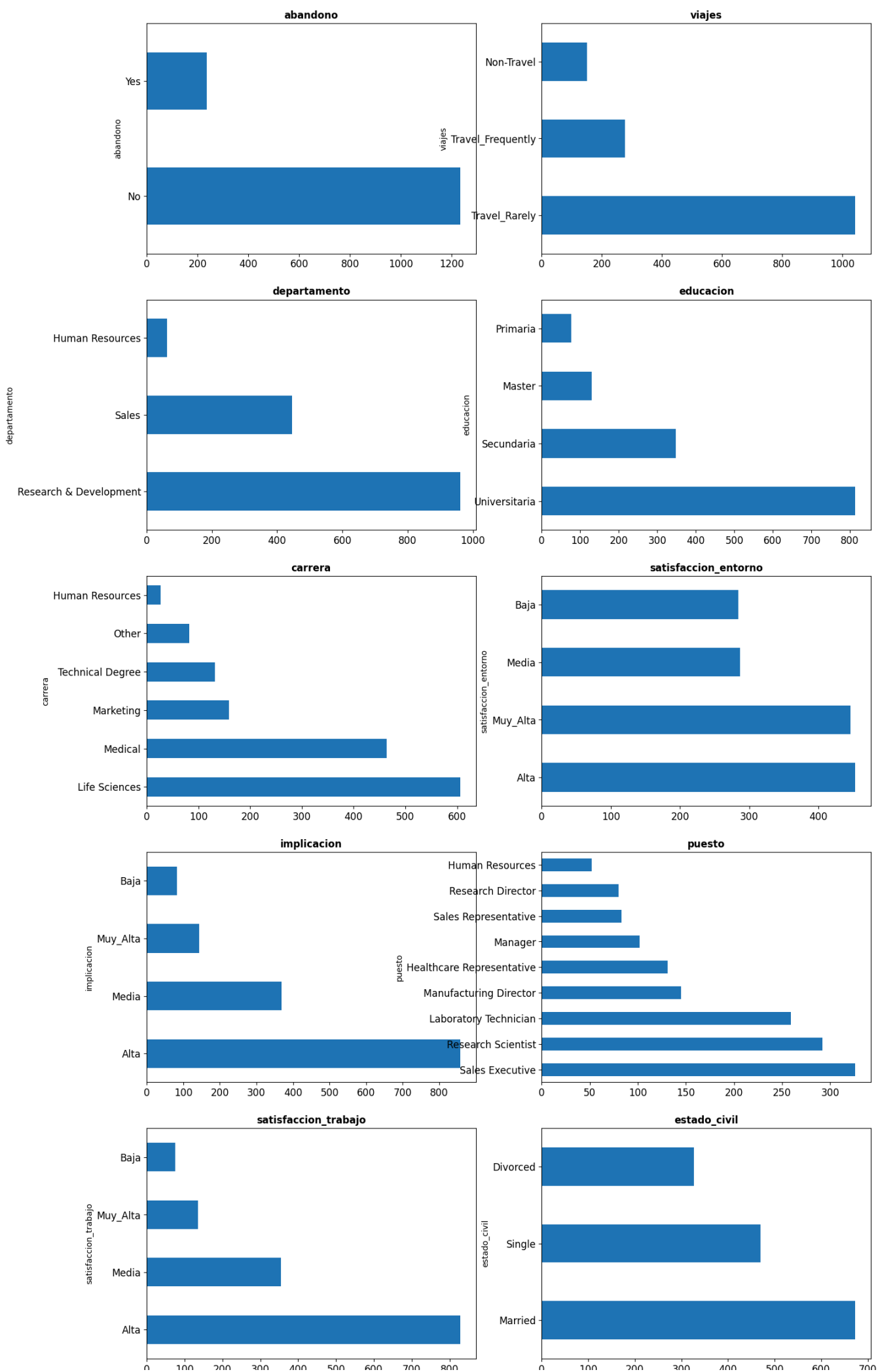
    #Calculo el número de filas que necesito
    from math import ceil
    filas = ceil(cat.shape[1] / 2)

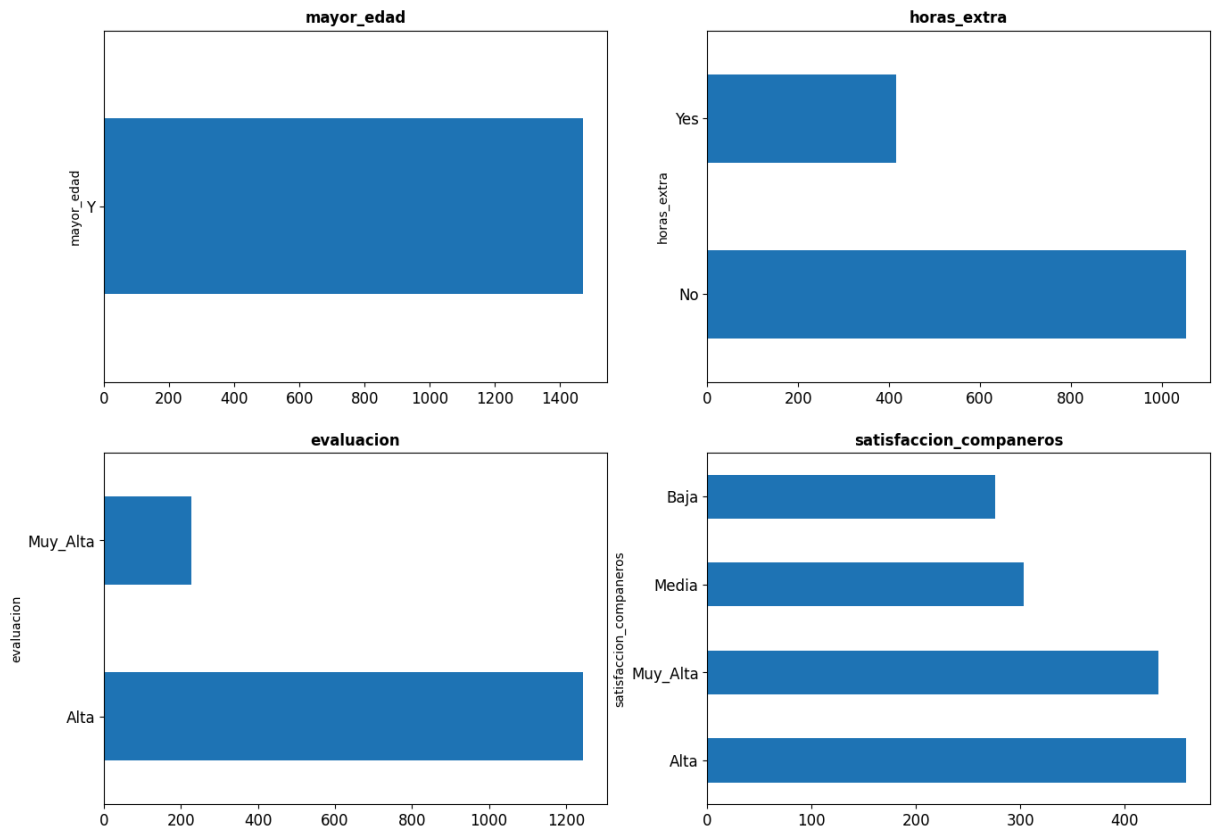
    #Defino el gráfico
    f, ax = plt.subplots(nrows = filas, ncols = 2, figsize = (16, filas * 6))

    #Aplano para iterar por el gráfico como si fuera de 1 dimensión en lugar de 2
    ax = ax.flat

    #Creo el bucle que va añadiendo gráficos
    for cada, variable in enumerate(cat):
        cat[variable].value_counts().plot.barh(ax = ax[cada])
        ax[cada].set_title(variable, fontsize = 12, fontweight = "bold")
        ax[cada].tick_params(labelsize = 12)

graficos_eda_categoricos(df.select_dtypes('O'))
```





## ✓ Conclusiones:

- mayor\_edad solo tiene un valor ==> Elimino
- Sobre las imputaciones pendientes de variables categóricas:
  - educacion: ==> Imputar por 'Universitaria'
  - satisfaccion\_trabajo: ==> Imputar por 'Alta'
  - implicacion: ==> Imputar por 'Alta'

```
df.drop(columns = 'mayor_edad', inplace = True)

df['educacion'] = df['educacion'].fillna('Universitaria')

df['satisfaccion_trabajo'] = df['satisfaccion_trabajo'].fillna('Alta')

df['implicacion'] = df['implicacion'].fillna('Alta')
```

## ✓ EDA VARIABLES NUMÉRICAS

```
def estadisticos_cont(num):
    #Calculo "describe".
    estadisticos = num.describe().T
    #Añado la mediana
    estadisticos['median'] = num.median()
    #Reordeno para que la mediana esté al lado de la media
    estadisticos = estadisticos.iloc[:, [0,1,8,2,3,4,5,6,7]]
    #Lo devuelvo con return.
    return(estadisticos)
```

```
estadisticos_cont(df.select_dtypes('number'))
```

	count	mean	median	std	min	25%	50%	75%	max
<b>edad</b>	1470.0	36.923810	36.0	9.135373	18.0	30.0	36.0	43.0	60.0
<b>distancia_casa</b>	1470.0	9.192517	7.0	8.106864	1.0	2.0	7.0	14.0	29.0
<b>empleados</b>	1470.0	1.000000	1.0	0.000000	1.0	1.0	1.0	1.0	1.0
<b>sexo</b>	1271.0	2.727773	3.0	0.720788	1.0	2.0	3.0	3.0	4.0
<b>nivel_laboral</b>	1470.0	2.063946	2.0	1.106940	1.0	1.0	2.0	3.0	5.0
<b>salario_mes</b>	1470.0	6502.931293	4919.0	4707.956783	1009.0	2911.0	4919.0	8379.0	19999.0
<b>num_empresas_anteriores</b>	1470.0	2.693197	2.0	2.498009	0.0	1.0	2.0	4.0	9.0
<b>incremento_salario_porc</b>	1470.0	15.209524	14.0	3.659938	11.0	12.0	14.0	18.0	25.0
<b>horas_quincena</b>	1470.0	80.000000	80.0	0.000000	80.0	80.0	80.0	80.0	80.0
<b>nivel_acciones</b>	1470.0	0.793878	1.0	0.852077	0.0	0.0	1.0	1.0	3.0
<b>anos_experiencia</b>	1470.0	11.279592	10.0	7.780782	0.0	6.0	10.0	15.0	40.0
<b>num_formaciones_ult_ano</b>	1470.0	2.799320	3.0	1.289271	0.0	2.0	3.0	3.0	6.0
<b>anos_compania</b>	1470.0	7.008163	5.0	6.126525	0.0	3.0	5.0	9.0	40.0
<b>anos_desde_ult_promocion</b>	1470.0	2.187755	1.0	3.222430	0.0	0.0	1.0	3.0	15.0
<b>anos_con_manager_actual</b>	1470.0	4.123129	3.0	3.568136	0.0	2.0	3.0	7.0	17.0

## ✓ Conclusiones:

- Empleados solo tiene un valor ==> Elimino
- Sexo tiene 4 valores ==> Elimino
- Horas quincena solo tiene una valor ==> Elimino

- De los nulos pendientes de imputación que sean numéricas solo está el sexo, pero como la voy a eliminar ya no hay que imputar nada.

```
df.drop(columns = ['empleados', 'sexo', 'horas_quincena'], inplace = True)
df
```



	edad	abandono	viajes	departamento	distancia_casa	educacion	carrera	satisfaccion_entorno	implicacion	nive
id										
1	41	Yes	Travel_Rarely	Sales		1	Universitaria	Life Sciences	Media	Alta
2	49	No	Travel_Frequently	Research & Development		8	Secundaria	Life Sciences	Alta	Media
4	37	Yes	Travel_Rarely	Research & Development		2	Secundaria	Other	Muy_Alta	Media
5	33	No	Travel_Frequently	Research & Development		3	Universitaria	Life Sciences	Muy_Alta	Alta
7	27	No	Travel_Rarely	Research & Development		2	Universitaria	Medical	Baja	Alta
...	...	...	...	...	...	...	...	...	...	...
2061	36	No	Travel_Frequently	Research & Development		23	Master	Medical	Alta	Muy_Alta
2062	39	No	Travel_Rarely	Research & Development		6	Secundaria	Medical	Muy_Alta	Media
2064	27	No	Travel_Rarely	Research & Development		4	Master	Life Sciences	Media	Muy_Alta
2065	49	No	Travel_Frequently	Sales		2	Secundaria	Medical	Muy_Alta	Media
2068	34	No	Travel_Rarely	Research & Development		8	Universitaria	Medical	Media	Muy_Alta


1470 rows × 25 columns



## ✓ APARTADO GENERACIÓN DE INSIGHTS

### ✓ Cuantificación del problema: ¿Cuál es la tasa de abandono?

```
df.abandono.value_counts(normalize = True) * 100
```



abandono	
No	83.877551
Yes	16.122449

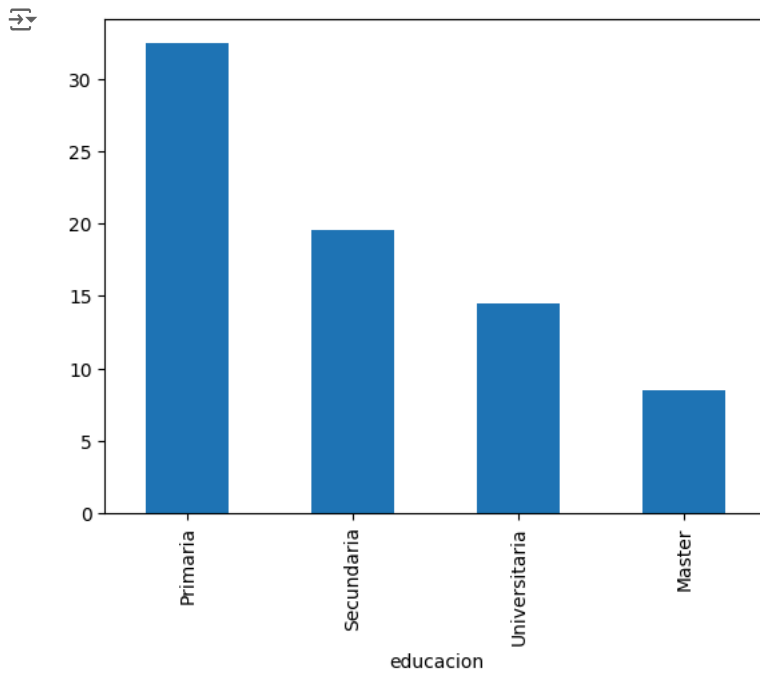
Name: proportion, dtype: float64

Haz doble clic (o pulsa Intro) para editar

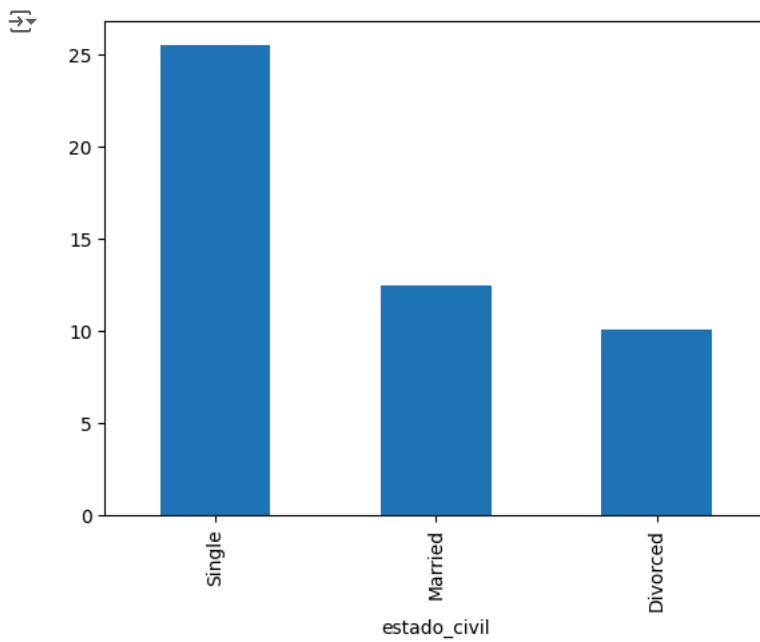
### ✓ ¿Hay un "perfil tipo" de empleado que deja la empresa?

```
# Transformo la variable abandono a numérica.
df['abandono'] = df.abandono.map({'No':0, 'Yes':1})
```

```
# Análisis por educación
temp = df.groupby('educacion').abandono.mean().sort_values(ascending = False) * 100
temp.plot.bar();
```

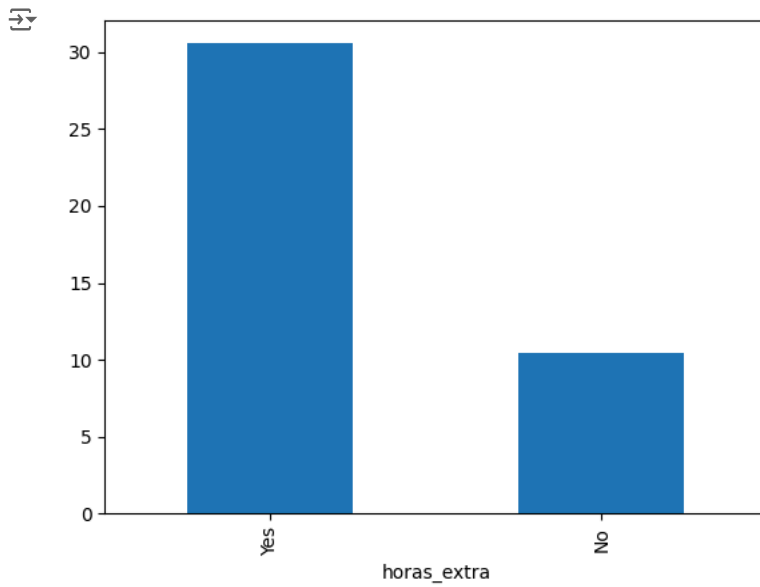


```
# Análisis por estado civil
temp = df.groupby('estado_civil').abandono.mean().sort_values(ascending = False) * 100
temp.plot.bar();
```

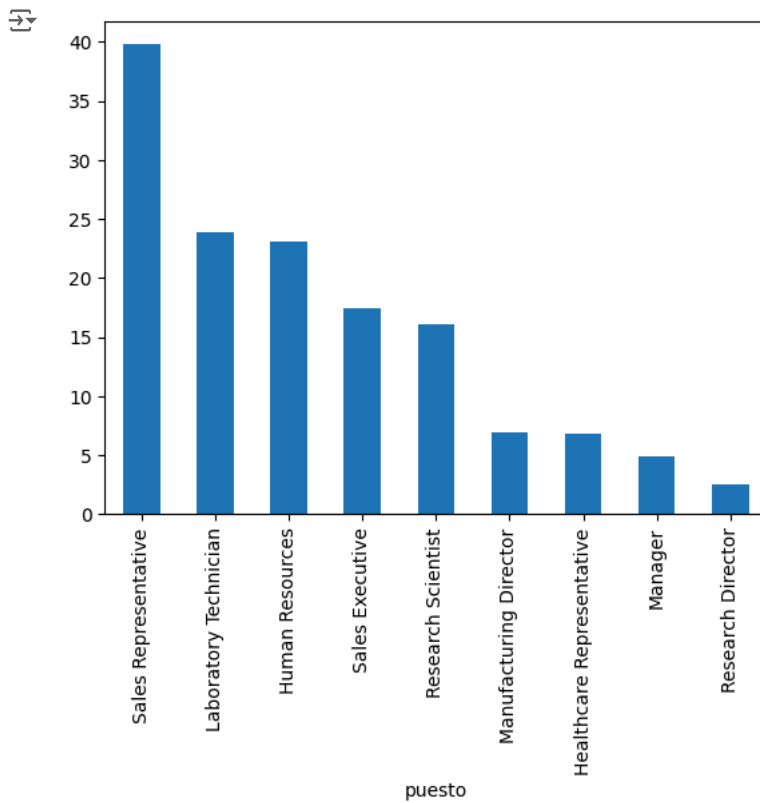


```
# Análisis por horas extras
temp = df.groupby('horas_extra').abandono.mean().sort_values(ascending = False) * 100
temp.plot.bar();
```

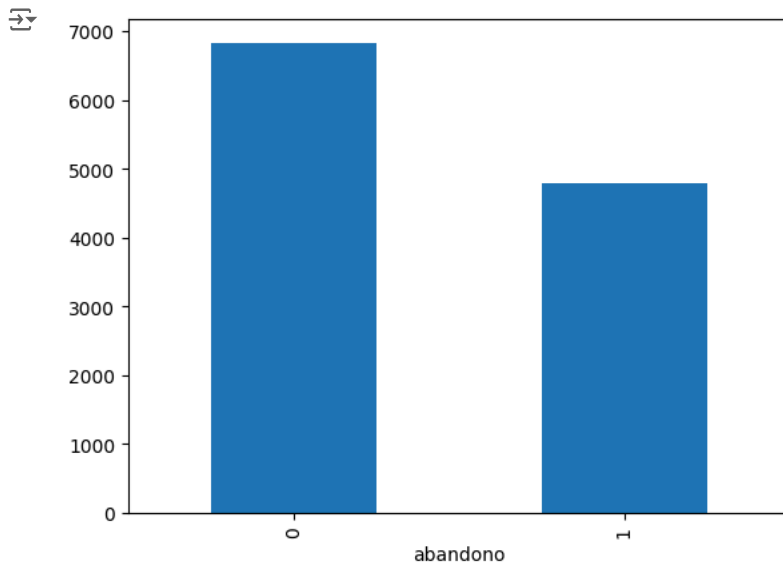




```
# Análisis por puesto
temp = df.groupby('puesto').abandono.mean().sort_values(ascending = False) * 100
temp.plot.bar();
```



```
temp = df.groupby('abandono').salario_mes.mean()
temp.plot.bar();
```



## ✓ Conclusiones:

El perfil medio del empleado que deja la empresa es:

- Bajo nivel educativo
- Soltero
- Trabaja en ventas
- Bajo salario
- Alta carga de horas extras

---


Empieza a programar o a [crear código](#) con IA.

¿Cuál es el impacto económico de este problema?


## ✓ Según el estudio "Cost of Turnover" del Center for American Progress:

- El coste de la fuga de los empleados que ganan menos de 30000 es del 16,1% de su salario
- El coste de la fuga de los empleados que ganan entre 30000-50000 es del 19,7% de su salario
- El coste de la fuga de los empleados que ganan entre 50000-75000 es del 20,4% de su salario
- El coste de la fuga de los empleados que ganan más de 75000 es del 21% de su salario

```
# Genero una nueva variable llamada "salario_ano" del empleado.  
df['salario_ano'] = df.salario_mes.transform(lambda x: x*12)  
df[['salario_mes', 'salario_ano']]
```



	salario_mes	salario_ano
--	-------------	-------------



id		
1	5993	71916
2	5130	61560
4	2090	25080
5	2909	34908
7	3468	41616
...	...	...
2061	2571	30852
2062	9991	119892
2064	6142	73704
2065	5390	64680
2068	4404	52848

1470 rows × 2 columns

```
# Calculo el impacto económico de cada empleado si deja la empresa.
```

```
#Lista de condiciones.
```

```
condiciones = [(df['salario_ano'] <= 30000),
                (df['salario_ano'] > 30000) & (df['salario_ano'] <= 50000),
                (df['salario_ano'] > 50000) & (df['salario_ano'] <= 75000),
                (df['salario_ano'] > 75000)]
```


```
#Lista de resultados.
```

```
resultados = [df.salario_ano * 0.161, df.salario_ano * 0.197, df.salario_ano * 0.204, df.salario_ano * 0.21]
```


```
#Aplico el método "select".
```

```
df['impacto_abandono'] = np.select(condiciones,resultados, default = -999)
```

```
df
```



	edad	abandono	viajes	departamento	distancia_casa	educacion	carrera	satisfaccion_entorno	implicacion	nive
--	------	----------	--------	--------------	----------------	-----------	---------	----------------------	-------------	------



id										
1	41	1	Travel_Rarely	Sales	1	Universitaria	Life Sciences	Media	Alta	
2	49	0	Travel_Frequently	Research & Development	8	Secundaria	Life Sciences	Alta	Media	
4	37	1	Travel_Rarely	Research & Development	2	Secundaria	Other	Muy_Alta	Media	
5	33	0	Travel_Frequently	Research & Development	3	Universitaria	Life Sciences	Muy_Alta	Alta	
7	27	0	Travel_Rarely	Research & Development	2	Universitaria	Medical	Baja	Alta	
...	...	...	...	...	...	...	...	...	...	...
2061	36	0	Travel_Frequently	Research & Development	23	Master	Medical	Alta	Muy_Alta	
2062	39	0	Travel_Rarely	Research & Development	6	Secundaria	Medical	Muy_Alta	Media	
2064	27	0	Travel_Rarely	Research & Development	4	Master	Life Sciences	Media	Muy_Alta	
2065	49	0	Travel_Frequently	Sales	2	Secundaria	Medical	Muy_Alta	Media	
2068	34	0	Travel_Rarely	Research & Development	8	Universitaria	Medical	Media	Muy_Alta	

1470 rows × 27 columns

▼ ¿Cuánto ha costado este problema a la empresa, en el último año?

```
coste_total = df.loc[df.abandono == 1].impacto_abandono.sum()
coste_total
```

↗ 2719005.912

---

✓ ¿Cuánto cuesta a la empresa, que los empleados no estén motivados? (pérdidas en implicación == Baja)

```
df.loc[(df.abandono == 1) & (df.implicacion == 'Baja')].impacto_abandono.sum()
```

↗ 368672.688

---

✓ ¿Cuánto dinero se podría ahorrar fidelizando mejor a nuestros empleados?

```
print(f"Reducir un 10% la fuga de empleados nos ahorraría {int(coste_total * 0.1)}$ cada año.")
print(f"Reducir un 20% la fuga de empleados nos ahorraría {int(coste_total * 0.2)}$ cada año.")
print(f"Reducir un 30% la fuga de empleados nos ahorraría {int(coste_total * 0.3)}$ cada año.")
```

↗ Reducir un 10% la fuga de empleados nos ahorraría 271900\$ cada año.  
Reducir un 20% la fuga de empleados nos ahorraría 543801\$ cada año.  
Reducir un 30% la fuga de empleados nos ahorraría 815701\$ cada año.

---

✓ Puedo seguir trazando estrategias asociadas a los insights de abandono:

Se ha visto que los "representantes de ventas" son el puesto que más se van.

¿Tendría sentido hacer un plan específico para ellos?

¿Cuál sería el coste ahorrado si se disminuye la fuga en un 30%?

Para ello, primero habrá que calcular el % de representantes de ventas que se fueron el año pasado.

```
total_repre_pasado = len(df.loc[df.puesto == 'Sales Representative'])
abandonos_repre_pasado = len(df.loc[(df.puesto == 'Sales Representative') & (df.abandono == 1)])
porc_pasado = abandonos_repre_pasado / total_repre_pasado
```

```
porc_pasado
```

↗ 0.39759036144578314

Ahora voy a estimar cuántos se nos irán este año

```
total_repre_actual = len(df.loc[(df.puesto == 'Sales Representative') & (df.abandono == 0)])
se_iran = int(total_repre_actual * porc_pasado)
```

```
se_iran
```

↗ 19

Sobre ellos, cuántos podemos retener (hipótesis 30%) y cuánto dinero puede suponer.

```
retenemos = int(se_iran * 0.3)
```

```
ahorramos = df.loc[(df.puesto == 'Sales Representative') & (df.abandono == 0)].impacto_abandono.sum() * porc_pasado * 0.3
```

```
print(f'Podemos retener {retenemos} representantes de ventas y ello supondría ahorrar {ahorramos}$.'
```

Podemos retener 5 representantes de ventas y ello supondría ahorrar 37447.22424578312\$.

Este dato también es muy interesante, porque permite determinar el presupuesto para acciones de retención por departamento o perfil.

Ahora ya sabemos, que podemos gastarnos hasta 37.000\$ sólo en acciones específicas para retener a "representantes de ventas" y se estarían pagando sólo con la pérdida evitada.

## ✓ APARTADO MODELO DE MACHINE LEARNING

```
df_ml = df.copy()
```

```
df_ml.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1470 entries, 1 to 2068
Data columns (total 27 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   edad                                 1470 non-null   int64
1   abandono                           1470 non-null   int64
2   viajes                             1470 non-null   object
3   departamento                       1470 non-null   object
4   distancia_casa                     1470 non-null   int64
5   educacion                          1470 non-null   object
6   carrera                            1470 non-null   object
7   satisfaccion_entorno               1470 non-null   object
8   implicacion                        1470 non-null   object
9   nivel_laboral                     1470 non-null   int64
10  puesto                             1470 non-null   object
11  satisfaccion_trabajo                1470 non-null   object
12  estado_civil                       1470 non-null   object
13  salario_mes                        1470 non-null   int64
14  num_empresas_anteriores             1470 non-null   int64
15  horas_extra                        1470 non-null   object
16  incremento_salario_porcentaje      1470 non-null   int64
17  evaluacion                         1470 non-null   object
18  satisfaccion_companeros             1470 non-null   object
19  nivel_acciones                     1470 non-null   int64
20  anos_experiencia                   1470 non-null   int64
21  num_formaciones_ult_ano             1470 non-null   int64
22  anos_compania                      1470 non-null   int64
23  anos_desde_ult_promocion            1470 non-null   int64
24  anos_con_manager_actual              1470 non-null   int64
25  salario_ano                        1470 non-null   int64
26  impacto_abandono                   1470 non-null   float64
dtypes: float64(1), int64(14), object(12)
memory usage: 321.6+ KB
```

## ✓ PREPARACIÓN DE LOS DATOS PARA LA MODELIZACIÓN

### ✓ Transformar todas las variables categóricas a numéricas

```
from sklearn.preprocessing import OneHotEncoder
```

```
#Categóricas
```

```
cat = df_ml.select_dtypes('O')
```

```
#Instanciando
```

```
ohe = OneHotEncoder(sparse = False)
```

```
#Entrenando
```

```
ohe.fit(cat)
```

```
#Aplico
```


```
cat_ohe = ohe.transform(cat)
```

```
#Pongo los nombres
```

```
cat_ohe = pd.DataFrame(cat_ohe, columns = ohe.get_feature_names_out(input_features = cat.columns)).reset_index(drop = True)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_`
warnings.warn(
```

```
cat_ohe
```



	viajes_Non-Travel	viajes_Travel_Frequently	viajes_Travel_Rarely	departamento_Human Resources	departamento_Research & Development	departamento_Sale
0	0.0	0.0	1.0	0.0	0.0	1.
1	0.0	1.0	0.0	0.0	1.0	0.
2	0.0	0.0	1.0	0.0	1.0	0.
3	0.0	1.0	0.0	0.0	1.0	0.
4	0.0	0.0	1.0	0.0	1.0	0.
...	...	...	...	...	...	.
1465	0.0	1.0	0.0	0.0	1.0	0.
1466	0.0	0.0	1.0	0.0	1.0	0.
1467	0.0	0.0	1.0	0.0	1.0	0.
1468	0.0	1.0	0.0	0.0	0.0	1.
1469	0.0	0.0	1.0	0.0	1.0	0.

1470 rows × 48 columns


## ▼ Dataframe final

Selecciono las variables numéricas para poder juntarlas a las cat\_hoe.

```
num = df.select_dtypes('number').reset_index(drop = True)
```

Las junto todas en el dataframe final.

```
df_ml = pd.concat([cat_ohe,num], axis = 1)
df_ml
```



	viajes_Non-Travel	viajes_Travel_Frequently	viajes_Travel_Rarely	departamento_Human Resources	departamento_Research & Development	departamento_Sale
0	0.0	0.0	1.0	0.0	0.0	1.
1	0.0	1.0	0.0	0.0	1.0	0.
2	0.0	0.0	1.0	0.0	1.0	0.
3	0.0	1.0	0.0	0.0	1.0	0.
4	0.0	0.0	1.0	0.0	1.0	0.
...	...	...	...	...	...	.
1465	0.0	1.0	0.0	0.0	1.0	0.
1466	0.0	0.0	1.0	0.0	1.0	0.
1467	0.0	0.0	1.0	0.0	1.0	0.
1468	0.0	1.0	0.0	0.0	0.0	1.
1469	0.0	0.0	1.0	0.0	1.0	0.

1470 rows × 63 columns

## ▼ DISEÑO DE LA MODELIZACIÓN

### ▼ Separación de variables "predictoras" y "target".

```
x = df_ml.drop(columns='abandono')
y = df_ml['abandono']
```

### ▼ Separación train y test.

```
from sklearn.model_selection import train_test_split


train_x, test_x, train_y, test_y = train_test_split(x, y, test_size = 0.3)
```

## ✓ ENTRENAMIENTO DEL MODELO SOBRE TRAIN

```
from sklearn.tree import DecisionTreeClassifier


#Instanciando
ac = DecisionTreeClassifier(max_depth=4)

#Entrenando
ac.fit(train_x,train_y)
```

 `DecisionTreeClassifier`  
`DecisionTreeClassifier(max_depth=4)`


## ✓ PREDICCIÓN Y VALIDACIÓN SOBRE TEST

```
# Predicción
pred = ac.predict_proba(test_x)[: , 1]
pred[:20]
```

 `array([0.0777439 , 0.0777439 , 0.0777439 , 0.36956522, 0.07857143,`  
`0.0777439 , 0.0777439 , 0.36956522, 0.0777439 , 0.0777439 ,`  
`0.0777439 , 0.0777439 , 0.0777439 , 0.0777439 , 0.07857143,`  
`0.0777439 , 0.07857143, 0.0777439 , 0.0777439 , 0.36956522])`

```
# Evaluación
from sklearn.metrics import roc_auc_score
```

```
roc_auc_score(test_y,pred)
```

 `0.6930304197150559`

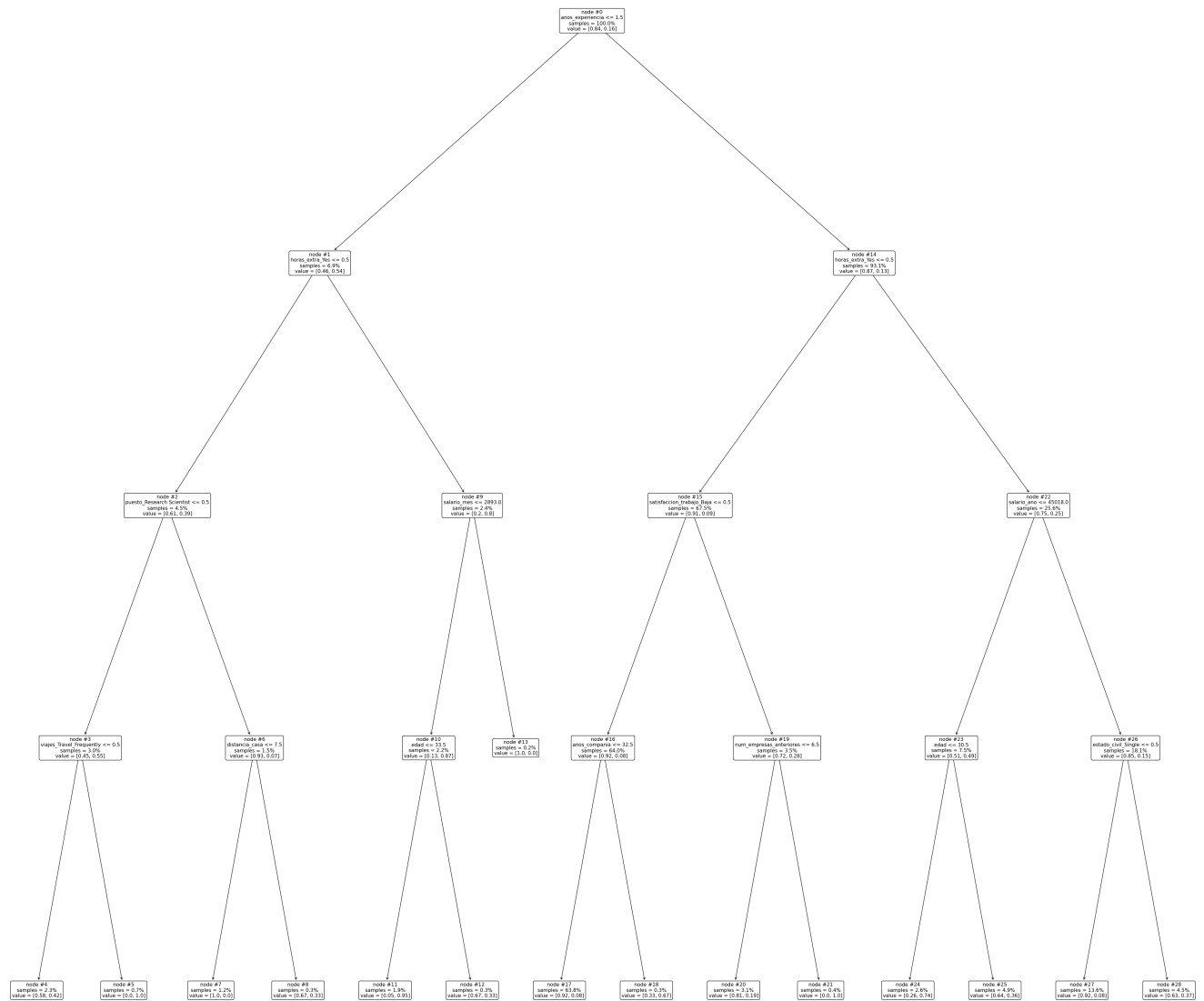
## ✓ INTERPRETACIÓN

### ✓ Diagrama del árbol

```
from sklearn.tree import plot_tree

plt.figure(figsize = (50,50))

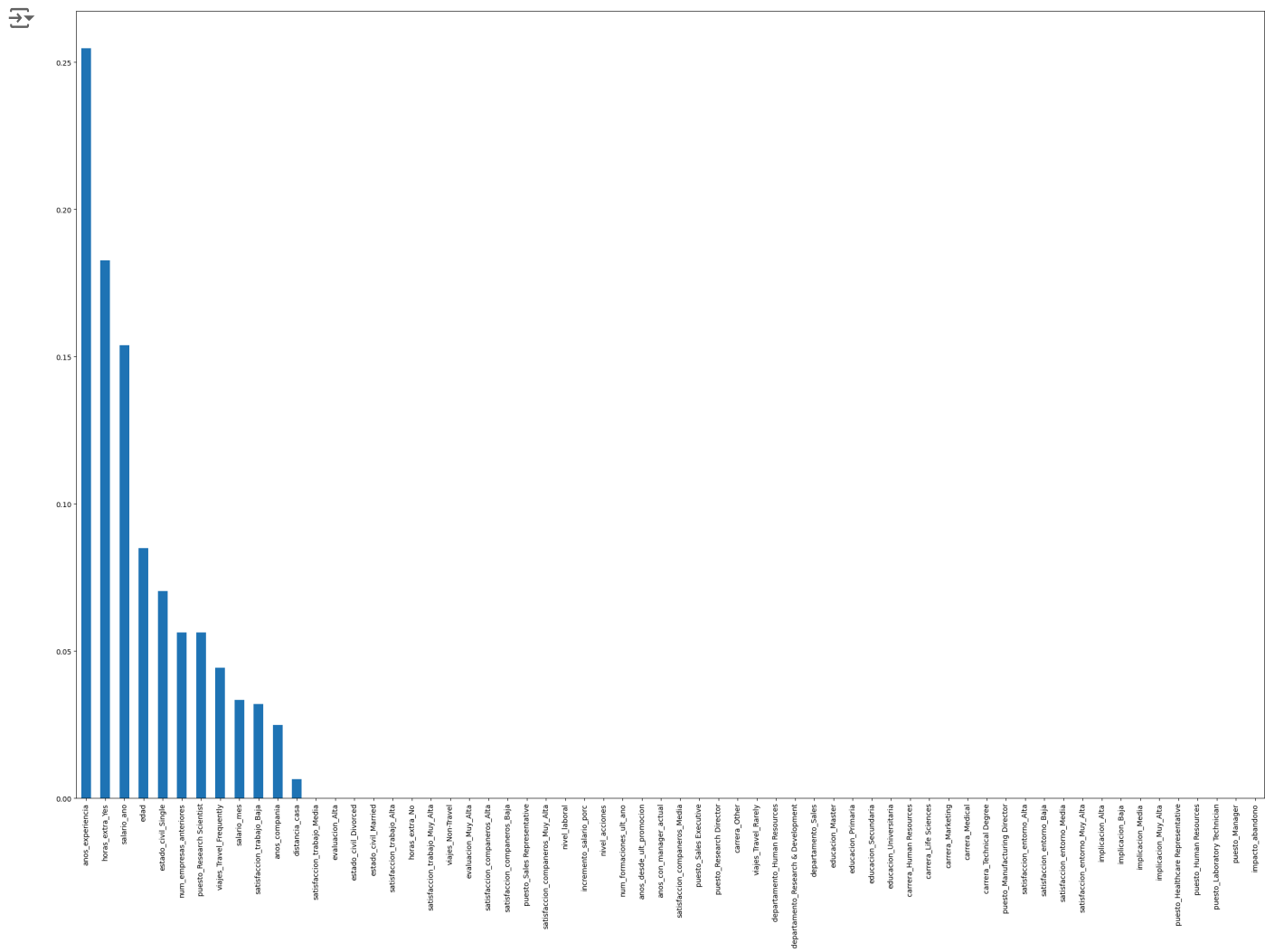
plot_tree(ac,
          feature_names= test_x.columns,
          impurity = False,
          node_ids = True,
          proportion = True,
          rounded = True,
          precision = 2);
```





Importancia de las variables

```
pd.Series(ac.feature_importances_,index = test_x.columns).sort_values(ascending = False).plot(kind = 'bar', figsize = (30,20));
```



EXPLORACIÓN

Incorporación del scoring al dataframe principal.

```
df['scoring_abandono'] = ac.predict_proba(df_ml.drop(columns = 'abandono'))[:, 1]
df
```

	edad	abandono	viajes	departamento	distancia_casa	educacion	carrera	satisfaccion_entorno	implicacion	nive
id										
1	41	1	Travel_Rarely	Sales	1	Universitaria	Life Sciences	Media	Alta	
2	49	0	Travel_Frequently	Research & Development	8	Secundaria	Life Sciences	Alta	Media	
4	37	1	Travel_Rarely	Research & Development	2	Secundaria	Other	Muy_Alta	Media	
5	33	0	Travel_Frequently	Research & Development	3	Universitaria	Life Sciences	Muy_Alta	Alta	
7	27	0	Travel_Rarely	Research & Development	2	Universitaria	Medical	Baja	Alta	
...	...	...	...	...	...	...	...	...	...	...
2061	36	0	Travel_Frequently	Research & Development	23	Master	Medical	Alta	Muy_Alta	
2062	39	0	Travel_Rarely	Research & Development	6	Secundaria	Medical	Muy_Alta	Media	
2064	27	0	Travel_Rarely	Research & Development	4	Master	Life Sciences	Media	Muy_Alta	
2065	49	0	Travel_Frequently	Sales	2	Secundaria	Medical	Muy_Alta	Media	
2068	34	0	Travel_Rarely	Research & Development	8	Universitaria	Medical	Media	Muy_Alta	

1470 rows × 28 columns

Ejemplo de los 10 empleados con mayor probabilidad de dejar la empresa.

```
df.sort_values(by = 'scoring_abandono', ascending = False)[0:10]
```

	edad	abandono	viajes	departamento	distancia_casa	educacion	carrera	satisfaccion_entorno	implicacion	ni
id										
648	30	1	Travel_Frequently	Sales	12	Secundaria	Life Sciences	Media	Media	
584	33	1	Travel_Rarely	Research & Development	10	Primaria	Medical	Baja	Baja	
1859	29	0	Travel_Rarely	Research & Development	29	Primaria	Life Sciences	Muy_Alta	Baja	
1037	26	1	Non-Travel	Sales	29	Universitaria	Medical	Media	Baja	
1716	47	1	Travel_Frequently	Sales	9	Primaria	Life Sciences	Alta	Baja	
1111	28	1	Travel_Frequently	Research & Development	1	Secundaria	Medical	Baja	Media	
1188	43	1	Travel_Rarely	Sales	9	Primaria	Marketing	Baja	Baja	
1944	27	1	Travel_Frequently	Human Resources	22	Secundaria	Human Resources	Baja	Media	
2023	23	1	Travel_Frequently	Sales	9	Universitaria	Marketing	Muy_Alta	Alta	
1427	31	1	Travel_Frequently	Sales	1	Primaria	Life Sciences	Media	Baja	

10 rows × 28 columns

Ejemplo: riesgo de dejar la empresa por puesto de trabajo.

```
df.boxplot(column='scoring_abandono', by='puesto', figsize = (20,12));
```

