

Abandono de empleados

En este trabajo se busca medir el impacto economico de los empleados que abandonan la empresa y mediante un arbol de decision se busca predecir cuales son los empleados con mayor probabilidad de abandonar la empresa. Se ofrece un analisis sobre las variables numericas y cualitativas, transformacion de datos y limpieza del dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
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

```
df.isna().sum().sort_values(ascending = False)
# De esta forma vemos las columnas y la cantidad de nulos, de aca se procede a eliminar columnas
```

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_porc	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
```

Observamos que las variables anos_en_puesto y conciliacion tiene muchos valores nulos por lo que no nos proporcionan informacion. Por otro lado se puede sexo, educacion, satisfaccion_trabajo e implicacion se puede llenar esos nulos con informacion

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

✓ Aca hacemos un EDA de variables categoricas.

Utilizando los graficos podemos observar de manera mas facil como llenar los datos nulos

```
def graficos_eda_categoricos(cat):

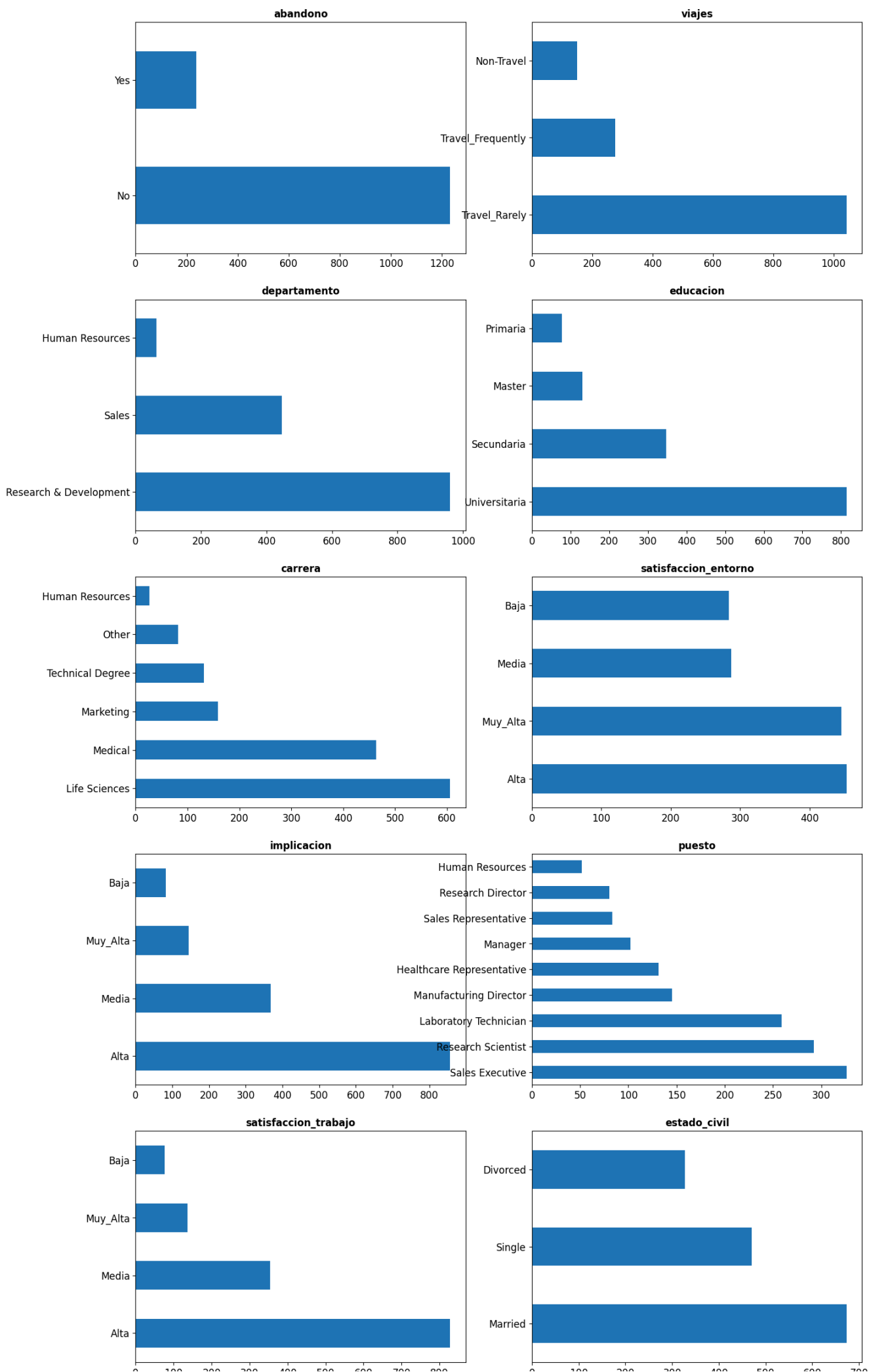
    #Calculamos el número de filas que necesitamos
    from math import ceil
    filas = ceil(cat.shape[1] / 2)

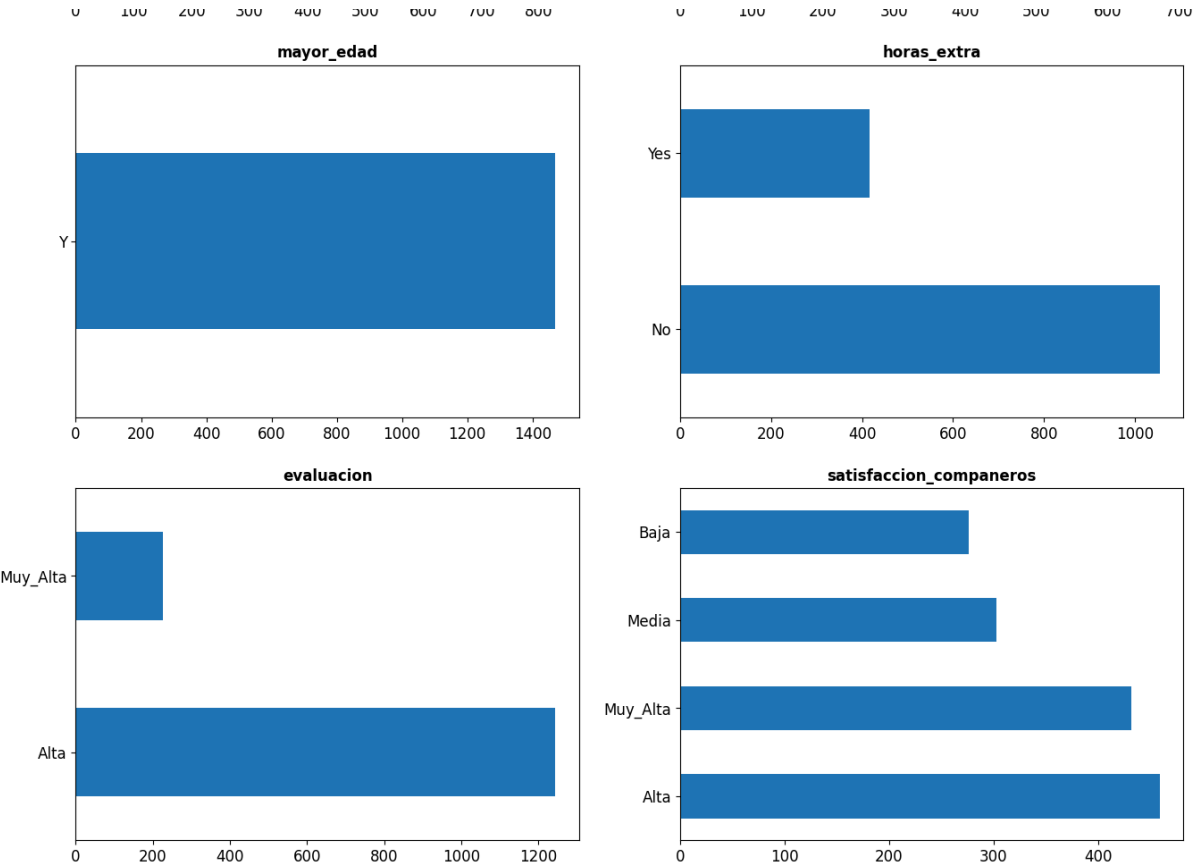
    #Definimos el gráfico
    f, ax = plt.subplots(nrows = filas, ncols = 2, figsize = (16, filas * 6))

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

    #Creamos 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'))
```





Aspecto a tener en cuenta: la variable educacion en su mayoría son universitarios y la variable implicacion la mayoría de datos es 'Alta', dicho esto es que procedemos a llenar los datos nulos con aquellos valores que mas se frecuentan. Algo que llamo la atencion es que la variables mayor_edad solo tiene un solo valor por ende vamos a eliminar esta variable

```
df.drop('mayor_edad',axis = 1, inplace = True)

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

✓ Ahora hacemos un analisis eda pero de variables continuas o numericas

```
import statistics
def estadisticos_cont(num):
    #Calculamos describe
    estadisticos = num.describe().T
    #Añadimos la mediana
    estadisticos['median'] = num.median()
    #Reordenamos para que la mediana esté al lado de la media
    estadisticos = estadisticos.iloc[:, [0,1,8,2,3,4,5,6,7]]
    #Lo devolvemos
    return(estadisticos)
```

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



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

Una vez visto el cuadro con las estadísticas, las variables horas_quincena y empleados son constantes por ende las eliminamos porque no proporcionan informacion. La variables sexo no se la entiende bien porque va del 1 al 4 cuando generalmente suele estar representada por dos valores que indica hombre o mujer, donde tambien eliminaremos esta variable

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

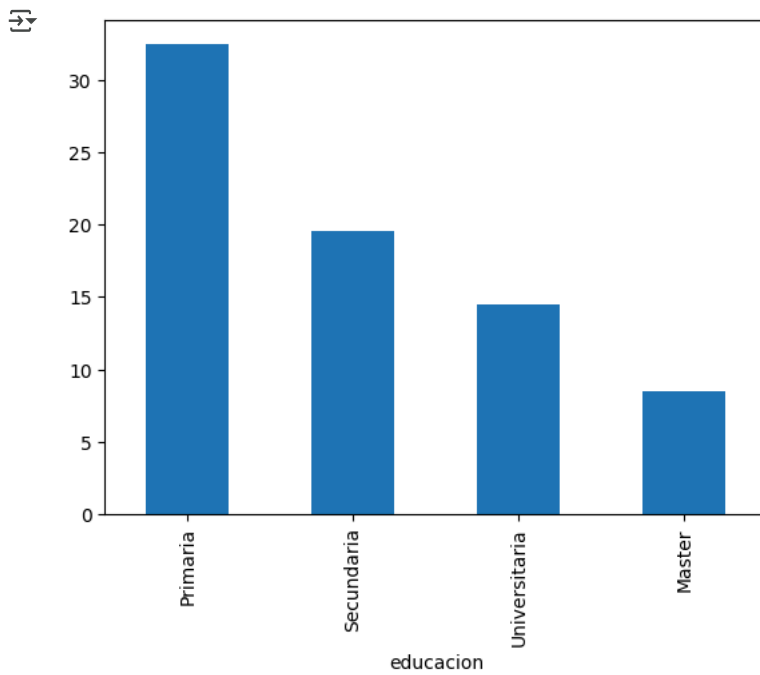
```
# Calculamos la tasa de abandono usando normalize
df['abandono'].value_counts(normalize = True)*100
```



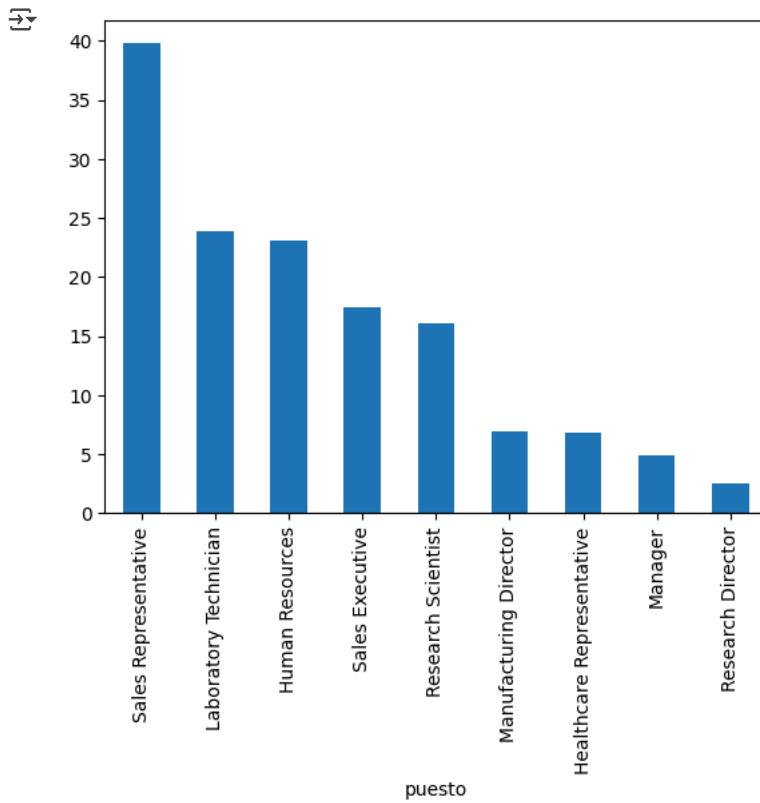
```
No      83.877551
Yes      16.122449
Name: abandono, dtype: float64
```

```
df['abandono'] = df.abandono.map({'No':0, 'Yes': 1})
```

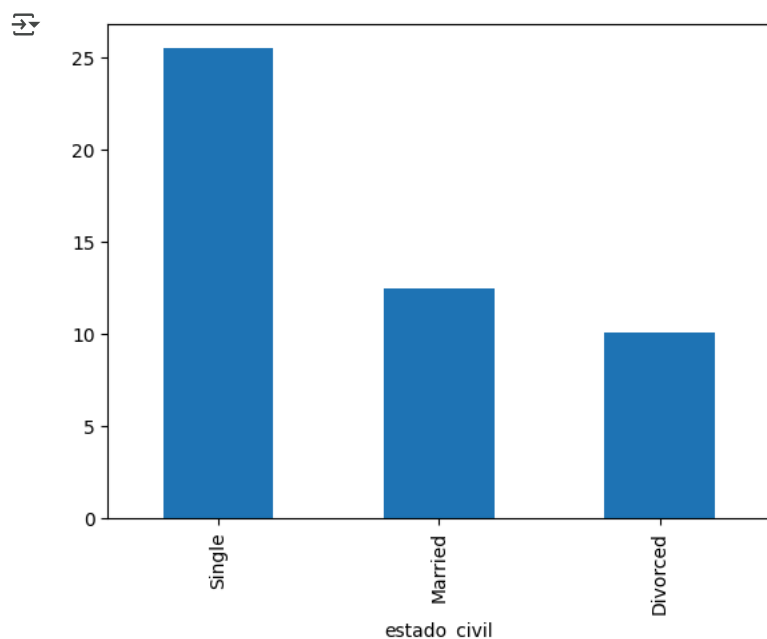
```
temp = df.groupby('educacion').abandono.mean().sort_values(ascending =False) * 100
temp.plot.bar();
```



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



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



✓ Analisis

Aca vamos a buscar calcular el costo economico del abandono. 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

```
df['salario_anual'] = df['salario_mes'].transform(lambda x : x * 12)
df[['salario_mes', 'salario_anual']]
```

	salario_mes	salario_anual
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

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

```
resultados = [df['salario_anual'] * 0.161, df['salario_anual'] * 0.197, df['salario_anual'] * 0.204, df['salario_anual'] * 0.21 ]
```

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


df




	edad	abandono	viajes	departamento	distancia_casa	educacion	carrera	satisfaccion_entorno	implicacion	nivel_1
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


```
costo_total = df[df['abandono']== 1 ].impacto_abandono.sum()
print(f'El costo total de los empleados que han abandonado la empresa es de: {costo_total}')
```

 El costo total de los empleados que han abandonado la empresa es de: 2719005.912

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

 368672.688

```
print(f'Si se reduce un 10% el abandono, nos ahorrariamos {int(costo_total * 0.1)} dolares anuales')
print(f'Si se reduce un 20% el abandono, nos ahorrariamos {int(costo_total * 0.2)} dolares anuales')
print(f'Si se reduce un 30% el abandono, nos ahorrariamos {int(costo_total * 0.3)} dolares anuales')
```


 Si se reduce un 10% el abandono, nos ahorrariamos 271900 dolares anuales
Si se reduce un 20% el abandono, nos ahorrariamos 543801 dolares anuales
Si se reduce un 30% el abandono, nos ahorrariamos 815701 dolares anuales

```
### Y podemos seguir trazando estrategias asociadas a los insights de abandono:
```

```
'''Habíamos visto que los representantes de ventas son el puesto que más se van. ¿Tendría sentido hacer un plan específico para ellos?
¿Cual sería el coste ahorrado si disminuimos la fuga un 30%?
Primero vamos a calcular el % de representantes de ventas que se han ido 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 vamos 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 cuantos podemos retener (hipótesis 30%) y cuanto 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}$.')

'''Este dato también es muy interesante porque nos permite determinar el presupuesto para acciones de retención por departamento o perfil.
Ya que sabemos que podemos gastarnos hasta 37.000$ sólo en acciones específicas para retener a representantes de ventas y se estarían pagando
...'''

Podemos retener 5 representantes de ventas y ello supondría ahorrar 37447.22424578312$.
Este dato también es muy interesante porque nos permite determinar el presupuesto para acciones de retención por departamento o perfil.
Ya que sabemos que podemos gastarnos hasta 37.000$ sólo en acciones específicas para retener a representantes de ventas y se estarían pagando
...'''

```

Machine learning

```

'''debemos preparar la base de datos para poder hacer el modelo de machine learning
para esto las variables categoricas deben pasarse a variables numericas
- no pueden haber valores nulos-'''

from sklearn.preprocessing import OneHotEncoder
#importo esta libreria que me permite transformar las categoricas en numericas

df_ml = df.copy()

#Categoricas
cat = df_ml.select_dtypes('O')

#Instanciamos
ohe = OneHotEncoder(sparse = False)

#Entrenamos
ohe.fit(cat)

#Aplicamos
cat_ohe = ohe.transform(cat)

#Ponemos 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:868: FutureWarning: `sparse` was renamed to `sparse_output`
warnings.warn(

cat_ohe


```

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

1470 rows × 48 columns

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

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



	edad	abandono	distancia_casa	nivel_laboral	salario_mes	num_empresas_anteriores	incremento_salario_porcentaje	nivel_acciones
0	41	1	1	2	5993	8	11	0
1	49	0	8	2	5130	1	23	1
2	37	1	2	1	2090	6	15	0
3	33	0	3	1	2909	1	11	0
4	27	0	2	1	3468	9	12	1
...
1465	36	0	23	2	2571	4	17	1
1466	39	0	6	3	9991	4	15	1
1467	27	0	4	2	6142	1	20	1
1468	49	0	2	2	5390	2	14	0
1469	34	0	8	2	4404	2	12	0

1470 rows × 9 columns

#ahora lo que hacemos es determinar las variables target y predictoras

```
x = df_ml.drop('abandono',axis =1)
y = df_ml['abandono']
```


```
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)
#elegimos un 70% para train y 30% para test
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
#Instanciar
ac = DecisionTreeClassifier(max_depth=4)
# 4 son las ramas que le indico al arbol
```


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



```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=4)
```


PREDICCION Y VALIDACION SOBRE TEST

```
# Predicción
pred = ac.predict_proba(test_x)[:, 1] #con el corchete le indico que me interesa mas los 1 que los 0
pred[:20] #observamos la probabilidad de abandono de los primeros 20 empleados
```



```
array([[0.04822335, 0.04822335, 0.07619048, 0.13855422, 0.07619048,
        0.04822335, 0.04822335, 0.27083333, 0.30612245, 0.56140351,
        0.04822335, 0.21212121, 0.04822335, 0.04822335, 0.06
        ,
        0.21212121, 0.04822335, 0.07619048, 0.04822335, 0.07619048])
```

```
# Evaluación
from sklearn.metrics import roc_auc_score
# esta libreria importada nos indica que tan bueno es el modelo: podemos decir que
# si el numero es menor a 70 no es bueno modelo, si esta entre 70 y 80 es medianamente
# bueno, y si esta entre 0.8 y 1 es un buen modelo
roc_auc_score(test_y,pred)
# le pasamos el targe de quien se ha ido y quien no ; y le pasamos la prediccion
# y compara el test con la prediccion e indica si el modelo es bueno o no
```

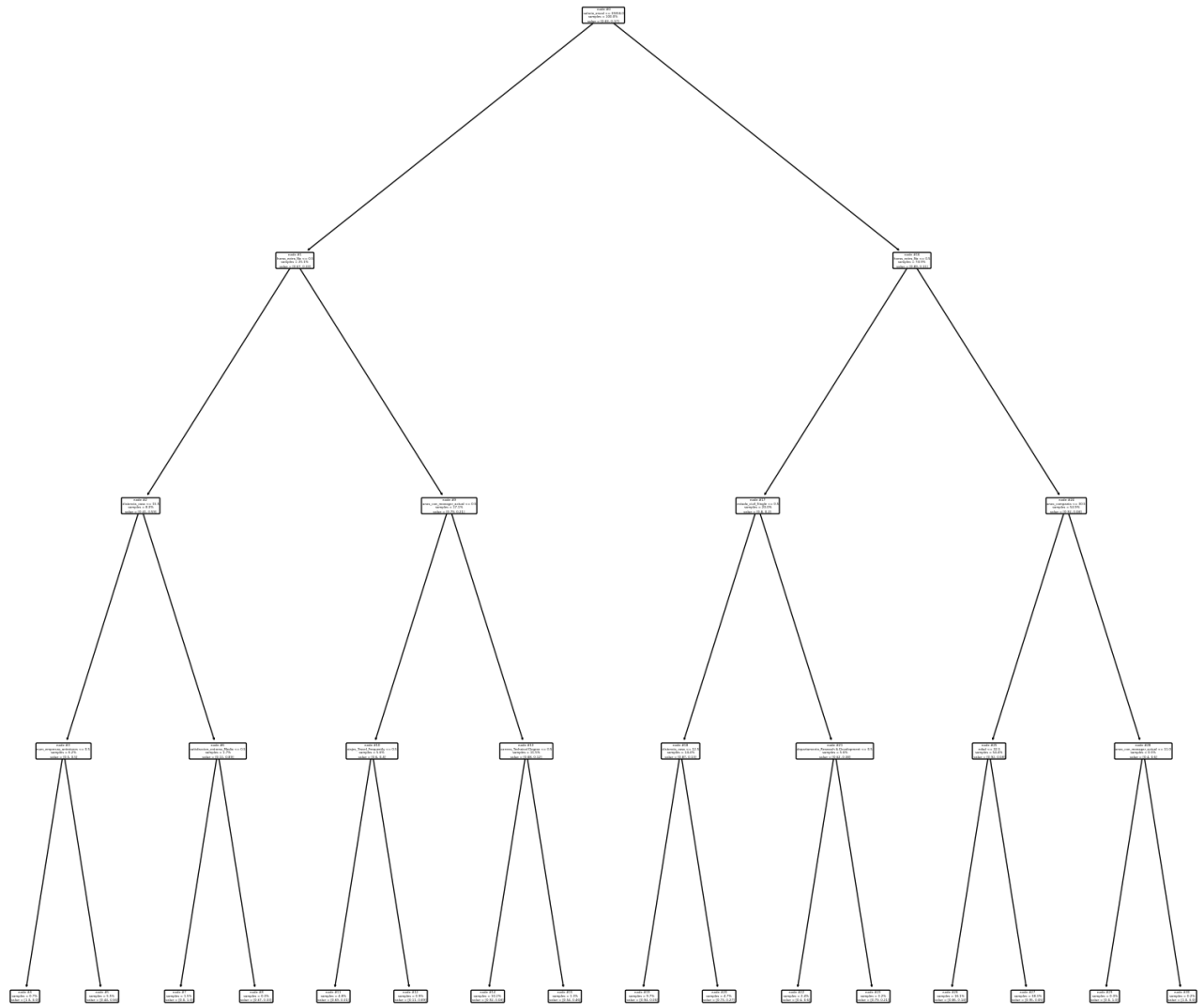


```
0.6862020202020203
```

```
from sklearn.tree import plot_tree
```

```
plt.figure(figsize = (20,20))
```

```
plot_tree(ac,  
          feature_names= test_x.columns,  
          impurity = False,  
          node_ids = True,  
          proportion = True,  
          rounded = True,  
          precision = 2);  
#en caso que no se vea bien la imagen se puede abrir en una pestaña nueva
```



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



FUGA DE EMPLEADOS

departamento

Todas

Tasa de abandono

16,1 %

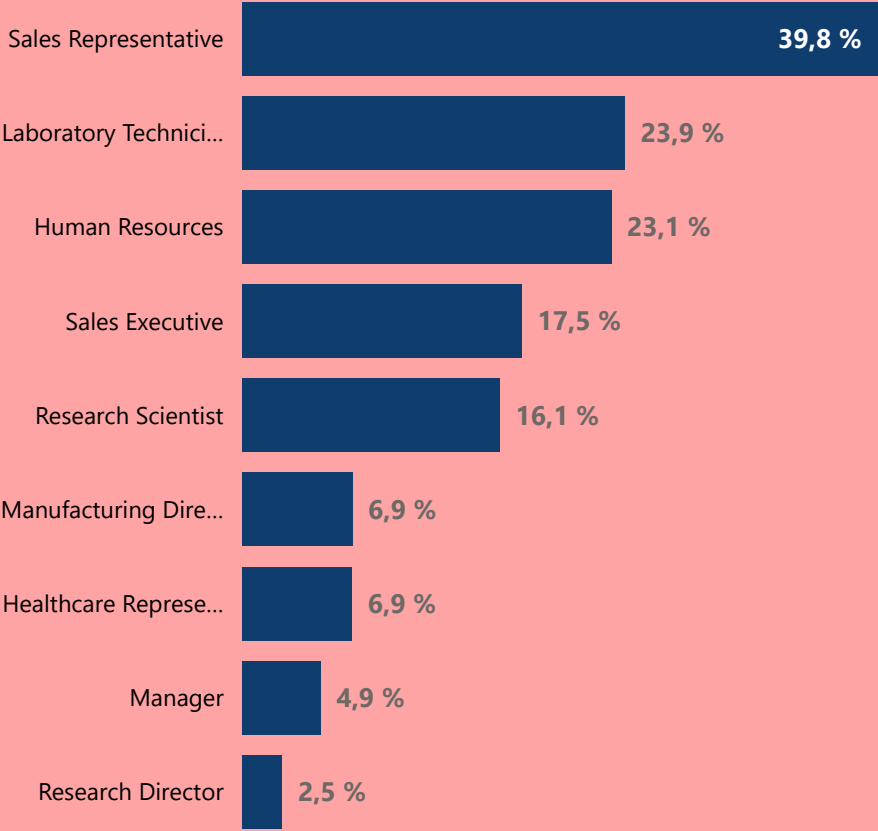
Cantidad en riesgo

99

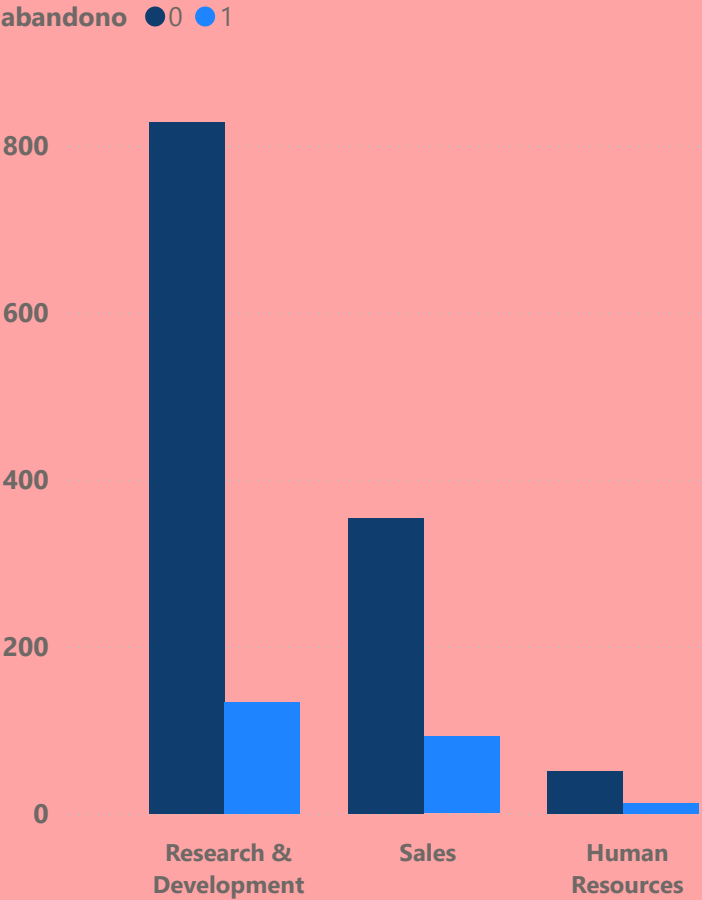
Impacto económico en USD

913.100

Tasa de abandono por puesto



Distinción de abandono por departamento



id	puesto	Impacto economico
4	Laboratory Technician	4.037,88
19	Laboratory Technician	3.918,10
33	Research Scientist	9.264,52
45	Research Scientist	4.430,08
55	Laboratory Technician	4.430,08
73	Research Director	33.914,16
133	Human Resources	4.005,04
167	Sales Representative	3.236,10
235	Sales Representative	4.491,90
259	Manager	50.397,48
284	Research Scientist	4.497,70
315	Research Scientist	4.615,55
325	Laboratory Technician	4.706,35
329	Manager	48.051,36
394	Laboratory Technician	4.584,64
399	Sales Representative	6.593,20
401	Sales Executive	14.266,94
403	Research Scientist	4.493,83
454	Laboratory Technician	4.093,91
478	Sales Representative	4.200,17
483	Laboratory Technician	4.275,52
514	Research Scientist	4.414,62
556	Laboratory Technician	3.288,26
614	Sales Representative	3.628,30
622	Laboratory Technician	4.520,88
644	Manager	45.864,00
648	Sales Representative	3.927,76
Total		913.099,87