

## 1.Importar librerías requeridas

```
import pandas as pd
import numpy as np
import sklearn as skl
```

## 2. Lee el archivo CSV llamado empleadosRETO.csv y coloca los datos en un frame de Pandas llamado EmpleadosAttrition.

```
EmpleadosAttrition=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/ingeniería de características/empleadosRETO.csv')
EmpleadosAttrition
```

	Age	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	50	Travel_Rarely	Research & Development	1 km	2	Medical	1	997
1	36	Travel_Rarely	Research & Development	6 km	2	Medical	1	178
2	21	Travel_Rarely	Sales	7 km	1	Marketing	1	1780
3	52	Travel_Rarely	Research & Development	7 km	4	Life Sciences	1	1118
4	33	Travel_Rarely	Research & Development	15 km	1	Medical	1	582
...	...	...	...	...	...	...	...	...
395	33	Travel_Rarely	Research & Development	14 km	3	Medical	1	325
396	31	Travel_Rarely	Sales	20 km	3	Life Sciences	1	175
397	37	Travel_Frequently	Research & Development	11 km	3	Other	1	306
398	38	Travel_Rarely	Research & Development	4 km	2	Medical	1	1687
399	33	Travel_Rarely	Research & Development	14 km	3	Medical	1	252

400 rows × 30 columns

## 3. Elimina las columnas que, con alta probabilidad (estimada por ti), no tienen relación alguna con la salida.

Hay algunas columnas que contienen información que no ayuda a definir el desgaste de un empleado, tal es caso de las siguientes:

- EmployeeCount: número de empleados, todos tienen un 1
- EmployeeNumber: ID del empleado, el cual es único para cada empleado
- Over18: mayores de edad, todos dicen "Y"
- StandardHours: horas de trabajo, todos tienen "80"

 Generar
 

Cerrar

```
EmpleadosAttrition= EmpleadosAttrition.drop(columns=['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'])
EmpleadosAttrition
```



	Age	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender
0	50	Travel_Rarely	Research & Development	1 km	2	Medical	4	Male
1	36	Travel_Rarely	Research & Development	6 km	2	Medical	2	Male
2	21	Travel_Rarely	Sales	7 km	1	Marketing	2	Male
3	52	Travel_Rarely	Research & Development	7 km	4	Life Sciences	2	Male
4	33	Travel_Rarely	Research & Development	15 km	1	Medical	2	Male
...	...	...	...	...	...	...	...	...
395	33	Travel_Rarely	Research & Development	14 km	3	Medical	3	Male
396	31	Travel_Rarely	Sales	20 km	3	Life Sciences	2	Female
397	37	Travel_Frequently	Research & Development	11 km	3	Other	2	Male
398	38	Travel_Rarely	Research & Development	4 km	2	Medical	4	Female

4. Analizando la información proporcionada, detectaste que no se cuenta con los años que el empleado lleva en la compañía y parece ser un buen dato. Dicha cantidad se puede calcular con la fecha de contratación 'HiringDate':

a. Crea una columna llamada Year y obtén el año de contratación del empleado a partir de su fecha 'HiringDate'. No se te olvide que debe ser un entero.

b. Crea una columna llamada YearsAtCompany que contenga los años que el empleado lleva en la compañía hasta el año 2018. Para su cálculo, usa la variable Year que acabas de crear.

```
EmpleadosAttrition['year'] = EmpleadosAttrition['HiringDate'].str.split(pat='/').str[2].astype(int)
EmpleadosAttrition['YearsAtCompany'] = 2018 - EmpleadosAttrition['year']
EmpleadosAttrition
```




	Age	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender
0	50	Travel_Rarely	Research & Development	1 km	2	Medical	4	Male
1	36	Travel_Rarely	Research & Development	6 km	2	Medical	2	Male
2	21	Travel_Rarely	Sales	7 km	1	Marketing	2	Male
3	52	Travel_Rarely	Research & Development	7 km	4	Life Sciences	2	Male
4	33	Travel_Rarely	Research & Development	15 km	1	Medical	2	Male
...	...	...	...	...	...	...	...	...
395	33	Travel_Rarely	Research & Development	14 km	3	Medical	3	Male
396	31	Travel_Rarely	Sales	20 km	3	Life Sciences	2	Female
397	37	Travel_Frequently	Research & Development	11 km	3	Other	2	Male
398	38	Travel_Rarely	Research & Development	4 km	2	Medical	4	Female
399	33	Travel_Rarely	Research & Development	14 km	3	Medical	4	Female

400 rows × 28 columns

5. La DistanceFromHome está dada en kilómetros, pero tiene las letras “km” al final y así no puede ser entera:

- a. Renombra la variable DistanceFromHome a DistanceFromHome\_km.
- b. Crea una nueva variable DistanceFromHome que sea entera, es decir, solo con números.


```
EmpleadosAttrition.rename(columns={'DistanceFromHome': 'DistanceFromHome_km'}, inplace =
True)
EmpleadosAttrition['DistanceFromHome'] = EmpleadosAttrition['DistanceFromHome_km'].str.split().str[0].astype(int)
EmpleadosAttrition
```




	Age	BusinessTravel	Department	DistanceFromHome_km	Education	EducationField	EnvironmentSatisfaction	Ge
0	50	Travel_Rarely	Research & Development	1 km	2	Medical	4	
1	36	Travel_Rarely	Research & Development	6 km	2	Medical	2	
2	21	Travel_Rarely	Sales	7 km	1	Marketing	2	
3	52	Travel_Rarely	Research & Development	7 km	4	Life Sciences	2	
4	33	Travel_Rarely	Research & Development	15 km	1	Medical	2	
...	...	...	...	...	...	...	...	
395	33	Travel_Rarely	Research & Development	14 km	3	Medical	3	
396	31	Travel_Rarely	Sales	20 km	3	Life Sciences	2	Fe
397	37	Travel_Frequently	Research & Development	11 km	3	Other	2	
398	38	Travel_Rarely	Research & Development	4 km	2	Medical	4	Fe
399	33	Travel_Rarely	Research & Development	14 km	3	Medical	4	Fe

400 rows × 29 columns

6. Borra las columnas Year, HiringDate y DistanceFromHome\_km debido a que ya no son útiles.

 Generar


10 random numbers using numpy



Cerrar

EmpleadosAttrition= EmpleadosAttrition.drop(columns=['year', 'HiringDate','DistanceFromHome\_km'])


EmpleadosAttrition




	Age	BusinessTravel	Department	Education	EducationField	EnvironmentSatisfaction	Gender	JobInvolvement
0	50	Travel_Rarely	Research & Development	2	Medical	4	Male	3
1	36	Travel_Rarely	Research & Development	2	Medical	2	Male	3
2	21	Travel_Rarely	Sales	1	Marketing	2	Male	3
3	52	Travel_Rarely	Research & Development	4	Life Sciences	2	Male	3
4	33	Travel_Rarely	Research & Development	1	Medical	2	Male	3

7. Aprovechando los ajustes que se están haciendo, la empresa desea saber si todos los departamentos tienen un ingreso promedio similar. Genera una nuevo frame llamado `SueldoPromedioDepto` que contenga el `MonthlyIncome` promedio por departamento de los empleados y colócalo en una variable llamada `SueldoPromedio`. Esta tabla solo es informativa, no la vas a utilizar en el set de datos que estás construyendo.

```
SueldoPromedioDepto=EmpleadosAttrition.groupby('Department')['MonthlyIncome'].mean()
SueldoPromedioDepto=SueldoPromedioDepto.reset_index()
SueldoPromedioDepto.rename(columns={'MonthlyIncome': 'SueldoPromedio'}, inplace =
True)
SueldoPromedioDepto
```



	Department	SueldoPromedio	
0	Human Resources	6239.888889	
1	Research & Development	6804.149813	
2	Sales	7188.250000	

Próximos pasos:

[Generar código con SueldoPromedioDepto](#)[Ver gráficos recomendados](#)[New interactive sheet](#)

8. La variable `MonthlyIncome` tiene un valor numérico muy grande comparada con las otras variables. Escala dicha variable para que tenga un valor entre 0 y 1.

 Generar

create a dataframe with 2 columns and 10 rows



Cerrar

```
EmpleadosAttrition['MonthlyIncome'] = (EmpleadosAttrition['MonthlyIncome']-
EmpleadosAttrition['MonthlyIncome'].min())/(EmpleadosAttrition['MonthlyIncome'].max()-
EmpleadosAttrition['MonthlyIncome'].min())
EmpleadosAttrition['MonthlyIncome']
```



MonthlyIncome

0 0.864269

9. Todo parece indicar que las variables categóricas que quedan sí son importantes para obtener la variable de salida.  
Convierte todas las variables categóricas que quedan a numéricas:

a. BusinessTravel

b. Department

c. EducationField

d. Gender

e. JobRole

f. MaritalStatus

g. Attrition

adicional: OverTime

`dtype: float64`

```
from sklearn.preprocessing import OneHotEncoder
# Aplicar OneHotEncoder a la columna 'BusinessTravel'
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore') # por si hay salidas NaN
BusinessTravel_encoded = encoder.fit_transform(EmpleadosAttrition[['BusinessTravel']])

# Para obtener los nombres del encoder
feature_names = encoder.get_feature_names_out(['BusinessTravel'])

# Crear un DataFrame en donde los nombres de la columnas sean los del encoder
BusinessTravel = pd.DataFrame(BusinessTravel_encoded, columns=feature_names)
BusinessTravel
```



	BusinessTravel_Non-Travel	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	BusinessTravel_nan
0	0.0	0.0	1.0	0.0
1	0.0	0.0	1.0	0.0
2	0.0	0.0	1.0	0.0
3	0.0	0.0	1.0	0.0
4	0.0	0.0	1.0	0.0
...	...	...	...	...
395	0.0	0.0	1.0	0.0
396	0.0	0.0	1.0	0.0
397	0.0	1.0	0.0	0.0
398	0.0	0.0	1.0	0.0
399	0.0	0.0	1.0	0.0



Próximos pasos:

[Generar código con BusinessTravel](#)[Ver gráficos recomendados](#)[New interactive sheet](#)

```
# Aplicar OneHotEncoder a la columna 'Department'
Department_encoded = encoder.fit_transform(EmpleadosAttrition[['Department']])

# Para obtener los nombres del encoder
feature_names = encoder.get_feature_names_out(['Department'])
```

```
# Crear un DataFrame en donde los nombres de la columnas sean los del encoder
Department = pd.DataFrame(Department_encoded, columns=feature_names)
Department
```

	Department_Human Resources	Department_Research & Development	Department_Sales
0	0.0	1.0	0.0
1	0.0	1.0	0.0
2	0.0	0.0	1.0
3	0.0	1.0	0.0
4	0.0	1.0	0.0
...	...	...	...
395	0.0	1.0	0.0
396	0.0	0.0	1.0
397	0.0	1.0	0.0
398	0.0	1.0	0.0
399	0.0	1.0	0.0

400 rows × 3 columns

Próximos pasos:

Generar código con Department

Ver gráficos recomendados

New interactive sheet

```
# Aplicar OneHotEncoder a la columna 'EducationField'
EducationField = encoder.fit_transform(EmpleadosAttrition[['EducationField']])

# Para obtener los nombres del encoder
feature_names = encoder.get_feature_names_out(['EducationField'])

# Crear un DataFrame en donde los nombres de la columnas sean los del encoder
EducationField = pd.DataFrame(EducationField, columns=feature_names)
EducationField
```

	EducationField_Human Resources	EducationField_Life Sciences	EducationField_Marketing	EducationField_Medical	EducationField_0
0	0.0	0.0	0.0	1.0	
1	0.0	0.0	0.0	1.0	
2	0.0	0.0	1.0	0.0	
3	0.0	1.0	0.0	0.0	
4	0.0	0.0	0.0	1.0	
...	...	...	...	...	...
395	0.0	0.0	0.0	1.0	
396	0.0	1.0	0.0	0.0	
397	0.0	0.0	0.0	0.0	
398	0.0	0.0	0.0	1.0	
399	0.0	0.0	0.0	1.0	

400 rows × 6 columns

Próximos pasos:

Generar código con EducationField

Ver gráficos recomendados

New interactive sheet

```
# Aplicar OneHotEncoder a la columna 'Gender'

Gender_encoded = encoder.fit_transform(EmpleadosAttrition[['Gender']])
```

```
Gender_encoded = encoder.fit_transform(EmpleadosAttrition[['Gender']])

# Para obtener los nombres del encoder
feature_names = encoder.get_feature_names_out(['Gender'])

# Crear un DataFrame en donde los nombres de la columnas sean los del encoder
Gender = pd.DataFrame(Gender_encoded, columns=feature_names)
Gender
```

	Gender_Female	Gender_Male
0	0.0	1.0
1	0.0	1.0
2	0.0	1.0
3	0.0	1.0
4	0.0	1.0
...	...	...
395	0.0	1.0
396	1.0	0.0
397	0.0	1.0
398	1.0	0.0
399	1.0	0.0

400 rows × 2 columns

Próximos pasos:

Generar código con Gender

Ver gráficos recomendados

New interactive sheet

```
# Aplicar OneHotEncoder a la columna 'JobRole'

JobRole_encoded = encoder.fit_transform(EmpleadosAttrition[['JobRole']])

# Para obtener los nombres del encoder
feature_names = encoder.get_feature_names_out(['JobRole'])

# Crear un DataFrame en donde los nombres de la columnas sean los del encoder
JobRole = pd.DataFrame(JobRole_encoded, columns=feature_names)
JobRole
```

	JobRole_Healthcare Representative	JobRole_Human Resources	JobRole_Laboratory Technician	JobRole_Manager	JobRole_Manufacturing Director	JobRole_Research Director
0	0.0	0.0	0.0	0.0	0.0	1.0
1	0.0	0.0	0.0	0.0	1.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...
395	0.0	0.0	1.0	0.0	0.0	0.0
396	0.0	0.0	0.0	0.0	0.0	0.0
397	0.0	0.0	0.0	0.0	0.0	1.0
398	0.0	0.0	1.0	0.0	0.0	0.0
399	0.0	0.0	0.0	0.0	0.0	0.0

400 rows × 9 columns

Próximos pasos:

Generar código con JobRole

Ver gráficos recomendados

New interactive sheet

```
# Aplicar OneHotEncoder a la columna 'MaritalStatus'
MaritalStatus_encoded = encoder.fit_transform(EmpleadosAttrition[['MaritalStatus']])

# Para obtener los nombres del encoder
feature_names = encoder.get_feature_names_out(['MaritalStatus'])

# Crear un DataFrame en donde los nombres de la columnas sean los del encoder
MaritalStatus = pd.DataFrame(MaritalStatus_encoded, columns=feature_names)
MaritalStatus
```

	MaritalStatus_Divorced	MaritalStatus_Married	MaritalStatus_Single	MaritalStatus_nan
0	1.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0
3	0.0	0.0	1.0	0.0
4	0.0	1.0	0.0	0.0
...	...	...	...	...
395	0.0	1.0	0.0	0.0
396	0.0	1.0	0.0	0.0
397	1.0	0.0	0.0	0.0
398	0.0	1.0	0.0	0.0
399	0.0	1.0	0.0	0.0

400 rows × 4 columns

Próximos pasos: [Generar código con MaritalStatus](#)

[Ver gráficos recomendados](#)

[New interactive sheet](#)

```
# Aplicar OneHotEncoder a la columna 'Attrition'
Attrition_encoded = encoder.fit_transform(EmpleadosAttrition[['Attrition']])

# Para obtener los nombres del encoder
feature_names = encoder.get_feature_names_out(['Attrition'])

# Crear un DataFrame en donde los nombres de la columnas sean los del encoder
Attrition = pd.DataFrame(Attrition_encoded, columns=feature_names)
Attrition
```

	Attrition_No	Attrition_Yes
0	1.0	0.0
1	1.0	0.0
2	0.0	1.0
3	1.0	0.0
4	0.0	1.0
...	...	...
395	0.0	1.0
396	0.0	1.0
397	1.0	0.0
398	1.0	0.0
399	1.0	0.0

400 rows × 2 columns

Próximos pasos: [Generar código con Attrition](#)

[Ver gráficos recomendados](#)

[New interactive sheet](#)

```
# Aplicar OneHotEncoder a la columna 'OverTime'
```



```
OverTime_encoded = encoder.fit_transform(EmpleadosAttrition[['OverTime']])
```


```
# Para obtener los nombres del encoder
```

```
feature_names = encoder.get_feature_names_out(['OverTime'])
```

```
# Crear un DataFrame en donde los nombres de la columnas sean los del encoder
```

```
OverTime = pd.DataFrame(OverTime_encoded, columns=feature_names)
```

```
OverTime
```



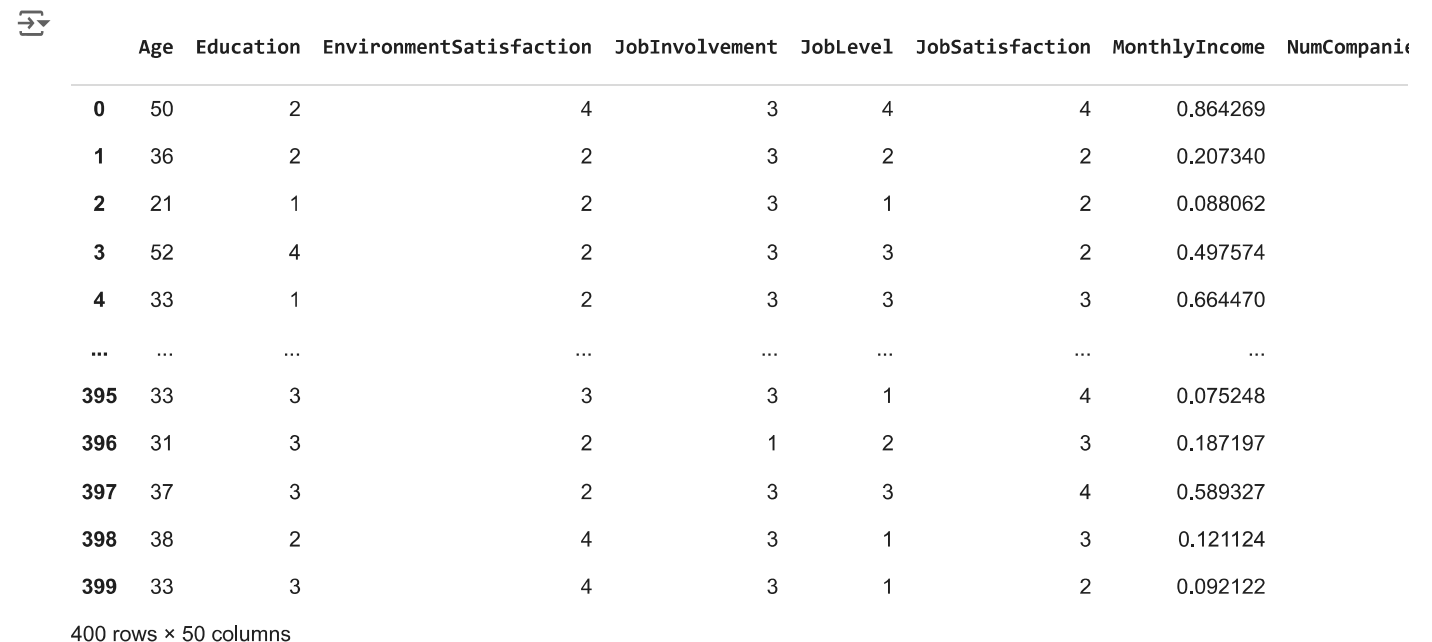
	OverTime_No	OverTime_Yes
0	1.0	0.0
1	1.0	0.0
2	1.0	0.0
3	1.0	0.0
4	0.0	1.0
...	...	...
395	0.0	1.0
396	0.0	1.0
397	0.0	1.0
398	1.0	0.0
399	1.0	0.0

400 rows × 2 columns

Próximos pasos:

[Generar código con OverTime](#)
[Ver gráficos recomendados](#)
[New interactive sheet](#)

```
EmpleadosAttrition=pd.concat([EmpleadosAttrition, BusinessTravel, Department, EducationField, Gender, JobRole, MaritalS
EmpleadosAttrition=EmpleadosAttrition.drop(columns=['BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole
EmpleadosAttrition
```



	Age	Education	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	NumCompanies
0	50	2	4	3	4	4	0.864269	
1	36	2	2	3	2	2	0.207340	
2	21	1	2	3	1	2	0.088062	
3	52	4	2	3	3	2	0.497574	
4	33	1	2	3	3	3	0.664470	
...	...	...	...	...	...	...	...	...
395	33	3	3	3	1	4	0.075248	
396	31	3	2	1	2	3	0.187197	
397	37	3	2	3	3	4	0.589327	
398	38	2	4	3	1	3	0.121124	
399	33	3	4	3	1	2	0.092122	

400 rows × 50 columns

10. Ahora debes hacer la evaluación de las variables para quedarte con las mejores. Calcula la correlación lineal de cada una de las variables con respecto al Attrition.

```
correlation_matrix=EmpleadosAttrition.corr()
Attrition_yes_corr=correlation_matrix['Attrition_Yes']
Attrition_yes_corr
```



	Attrition_Yes
Age	-0.212121
Education	-0.055531
EnvironmentSatisfaction	-0.124327
JobInvolvement	-0.166785
JobLevel	-0.214266
JobSatisfaction	-0.164957
MonthlyIncome	-0.194936
NumCompaniesWorked	-0.009082
PercentSalaryHike	-0.060880
PerformanceRating	-0.006471
RelationshipSatisfaction	-0.030945
TotalWorkingYears	-0.213329
TrainingTimesLastYear	-0.070884
WorkLifeBalance	-0.021723
YearsInCurrentRole	-0.203918
YearsSinceLastPromotion	-0.069000
YearsAtCompany	-0.176001
DistanceFromHome	0.052732
BusinessTravel_Non-Travel	-0.100698
BusinessTravel_Travel_Frequently	0.035387
BusinessTravel_Travel_Rarely	0.042755
BusinessTravel_nan	-0.044677
Department_Human Resources	0.023389
Department_Research & Development	-0.072269
Department_Sales	0.066116
EducationField_Human Resources	0.043404
EducationField_Life Sciences	-0.027457
EducationField_Marketing	0.016768
EducationField_Medical	-0.054144
EducationField_Other	-0.004275
EducationField_Technical Degree	0.129104
Gender_Female	0.028839
Gender_Male	-0.028839
JobRole_Healthcare Representative	-0.103274
JobRole_Human Resources	0.032714
JobRole_Laboratory Technician	0.125264
JobRole_Manager	-0.089885
JobRole_Manufacturing Director	-0.042404
JobRole_Research Director	-0.116263
JobRole_Research Scientist	0.007977
JobRole_Sales Executive	-0.003115
JobRole_Sales Representative	0.191294
MaritalStatus_Divorced	-0.107869

MaritalStatus_Married	-0.094734
MaritalStatus_Single	0.205849
MaritalStatus_nan	0.010609
Attrition_No	-1.000000
Attrition_Yes	1.000000
OverTime_No	-0.324777
OverTime_Yes	0.324777

**dtype:** float64

11. Selecciona solo aquellas variables que tengan una correlación mayor o igual a 0.1, dejándolas en otro frame llamado EmpleadosAttritionFinal. No olvides mantener la variable de salida Attrition; esto es equivalente a borrar las que no cumplen con el límite.

```
#Buscando los valores de corr mayores a 0.1. Nota, se usa siempre el valor absoluto
print(Attrition_yes_corr[abs(Attrition_yes_corr)>=0.1])
```

```
#seleccionar la tabla con las columnas. Nótese que como con el encoder se generó
#Attrition_Yes, Attrition_No, OverTime_Yes y OverTime No, solo se seleccionaron las columnas con Yes y se renombraron
EmpleadosAttritionFinal=EmpleadosAttrition[['Age', 'EnvironmentSatisfaction', 'JobInvolvement', 'JobLevel', 'JobSatisfac
EmpleadosAttritionFinal.rename(columns={'Attrition_Yes': 'Attrition'}, inplace =
True)
EmpleadosAttritionFinal.rename(columns={'OverTime_Yes': 'Overtime'}, inplace =
True)
EmpleadosAttritionFinal
```

```

➡ Age -0.212121
EnvironmentSatisfaction -0.124327
JobInvolvement -0.166785
JobLevel -0.214266
JobSatisfaction -0.164957
MonthlyIncome -0.194936
TotalWorkingYears -0.213329
YearsInCurrentRole -0.203918
YearsAtCompany -0.176001
BusinessTravel_Non-Travel -0.100698
EducationField_Technical Degree 0.129104
JobRole_Healthcare Representative -0.103274
JobRole_Laboratory Technician 0.125264
JobRole_Research Director -0.116263
JobRole_Sales Representative 0.191294
MaritalStatus_Divorced -0.107869
MaritalStatus_Single 0.205849
Attrition_No -1.000000
Attrition_Yes 1.000000
OverTime_No -0.324777
OverTime_Yes 0.324777
Name: Attrition_Yes, dtype: float64
<ipython-input-112-de0885161d5e>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
EmpleadosAttritionFinal.rename(columns={'Attrition_Yes': 'Attrition'}, inplace =
<ipython-input-112-de0885161d5e>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
EmpleadosAttritionFinal.rename(columns={'OverTime_Yes': 'Overtime'}, inplace =

```

	Age	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	TotalWorkingYears
0	50	4	3	4	4	0.864269	32
1	36	2	3	2	2	0.207340	7
2	21	2	3	1	2	0.088062	1
3	52	2	3	3	2	0.497574	18
4	33	2	3	3	3	0.664470	15
...	...	...	...	...	...	...	...
395	33	3	3	1	4	0.075248	8
396	31	2	1	2	3	0.187197	4
397	37	2	3	3	4	0.589327	10
398	38	4	3	1	3	0.121124	7
399	33	4	3	1	2	0.092122	8

400 rows × 19 columns

Próximos pasos:

[Generar código con EmpleadosAttritionFinal](#)

[Ver gráficos recomendados](#)

[New interactive sheet](#)

12. Crea una nueva variable llamada EmpleadosAttritionPCA formada por los componentes principales del frame EmpleadosAttritionFinal. Recuerda que el resultado del proceso PCA es un numpy array, por lo que, para hacer referencia a una columna, por ejemplo, la 0, puedes usar la instrucción EmpleadosAttritionPCA[:,0]).

```

#importar y aplicar la transformación
from sklearn.decomposition import PCA
pca = PCA(19) #son 19 porque son 19 columnas
EmpleadosAttritionPCA = pca.fit_transform(EmpleadosAttritionFinal)
EmpleadosAttritionPCA=pd.DataFrame(EmpleadosAttritionPCA)
print(EmpleadosAttritionPCA)

```

```

0  20.252734 -4.207118 10.721005 1.868202 0.322578 2.050336 0.097166
1  -6.382336 -3.690989 -0.156199 -0.215953 -0.069780 -0.917947 0.474261
2  -21.513871 1.744893 3.225836 -0.987670 -0.081283 -0.970779 0.175403
3  13.827285 -5.843714 -2.224650 0.987369 -0.175956 -0.992844 0.290588
4  -1.434535 4.144322 3.924502 1.673850 0.592946 -0.383056 0.542497
..      ...      ...      ...      ...      ...      ...
395 -6.977475 0.384383 0.195720 0.605111 0.767822 0.993200 -0.825433
396 -12.043090 -2.043794 0.205287 0.260124 0.752201 -0.430748 0.325027
397 -0.642749 3.420290 -3.347864 1.449444 1.409517 0.125163 1.085773
398 -6.313785 -7.697475 0.860586 -0.338977 -0.475273 1.312528 -0.452975
399 -5.589425 3.527690 -2.061372 1.531561 -1.438211 0.634778 -0.536651

7  0.443771 -0.695972 0.163452 0.307199 0.409713 0.499497 -0.204595
8  0.228920 -0.837801 0.044091 -0.101315 0.415987 0.139596 -0.061672
9  0.237488 0.662161 -0.318825 -0.122170 0.344002 0.666507 0.067232
10 0.177851 0.553020 -0.501046 -0.289485 0.410621 -0.345118 0.727763
11 0.248725 0.393926 0.974987 0.050542 -0.357246 0.323287 0.171902
..      ...      ...      ...      ...      ...      ...
395 0.472170 0.428193 0.909087 0.642571 -0.553890 0.199659 0.293151
396 -1.808071 0.235075 0.887290 -0.133881 -0.493664 0.189130 0.036277
397 0.146913 -0.345588 0.984529 0.152730 0.361801 0.007553 -0.394461
398 0.422710 -0.256279 -0.273526 0.522307 -0.524359 -0.030094 0.097632
399 0.445293 -0.121545 -0.332054 -0.329148 -0.253162 0.094242 -0.162347

14 -0.082915 0.534587 -0.188034 -0.107405 0.049467
15 -0.124457 -0.189321 0.011874 -0.078071 -0.050903
16 0.346318 0.191278 -0.022676 0.489495 0.018221
17 -0.114979 0.106768 0.001121 0.054099 0.054121
18 0.289076 -0.269640 -0.200723 0.024050 0.186224
..      ...      ...      ...      ...
395 0.106587 0.047221 -0.186131 0.087843 0.009168
396 0.054348 -0.188413 -0.230552 0.003983 -0.082337
397 -0.405937 0.555391 -0.210787 -0.015977 0.025234
398 -0.181287 0.058674 0.068174 0.064508 0.051315
399 0.008860 -0.054734 -0.161082 -0.123484 0.021238

```

[400 rows x 19 columns]

13. Agrega el mínimo número de Componentes Principales en columnas del frame EmpleadosAttritionPCA que logren explicar el 80% de la varianza, al frame EmpleadosAttritionFinal. Puedes usar la instrucción assign, columna por columna, llamando a cada una C0, C1, etc., hasta las que vayas a agregar

Es posible que el código generado esté sujeto a una licencia |

#cálculo de porcentaje

print(pca.explained\_variance\_ratio\_)

#el siguiente código es para visualizar valores de manera más clara sin exponentes

ratio=pd.DataFrame(pca.explained\_variance\_ratio\_)

ratio

#La columna donde con mayor porcentaje de la varianza es la primer columna con 63.4%,

# por lo cual al ser menor que el 80%, no se agrega nada al Data Frame EmpleadosAttritionFinal