

```
# 1. Importa las librerías requeridas.
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder

# 2. Lee el archivo CSV llamado empleadosRETO.csv y coloca los datos en un frame de Pandas llamado EmpleadosAttrition.
EmpleadosAttrition = pd.read_csv("empleadosRETO.csv")

# 3. Elimina las columnas que, con alta probabilidad (estimada por ti), no tienen relación alguna con la salida. Hay algunas columnas que cc
# 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"
EmpleadosAttrition.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis = 1, inplace=True)

# 4. Analiza la información proporcionada, si detectaste que no se cuenta con los años que el empleado lleva en la compañía y parece ser un
# Se convierte la columna 'HiringDate' a formato datetime
EmpleadosAttrition['HiringDate'] = pd.to_datetime(EmpleadosAttrition['HiringDate'], format='%m/%d/%Y', errors='coerce')

# Verificar si hay fechas no válidas
invalid_dates = EmpleadosAttrition[EmpleadosAttrition['HiringDate'].isna()]
if not invalid_dates.empty:
    # El registro 36,Travel_Rarely,Sales,11 km,4,Marketing,1,2045,2,Female,2,2,Sales Executive,4,Married,6652,4,2/30/2012,Y,No,13,3,1,80,8,2
    # que genera un error al convertirlo a fecha
    #print("Hay fechas no válidas que eliminam:" + str(invalid_dates))
    EmpleadosAttrition = EmpleadosAttrition.dropna(subset=['HiringDate'])

# 5. 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 s
# Se crea la nueva columna 'Year' con el año de contratación
EmpleadosAttrition['Year'] = EmpleadosAttrition['HiringDate'].dt.year

# 6. Crea una columna llamada YearsAtCompany que contenga los años que el empleado lleva en la compañía hasta el año 2018. Para su cálculo,
# Se crea la columna 'YearsAtCompany' calculando los años hasta 2018
EmpleadosAttrition['YearsAtCompany'] = 2018 - EmpleadosAttrition['Year']

# 7. La DistanceFromHome está dada en kilómetros, pero tiene las letras "km" al final y así no puede ser entera
# 8. Renombra la variable DistanceFromHome a DistanceFromHome_km
EmpleadosAttrition.rename(columns={'DistanceFromHome': 'DistanceFromHome_km'}, inplace=True)

# 9. Crea una nueva variable DistanceFromHome que sea entera, es decir, solo con números
EmpleadosAttrition['DistanceFromHome'] = EmpleadosAttrition['DistanceFromHome_km'].str.extract(r'(\d+)').astype(int)

# 10. Borra las columnas Year, HiringDate y DistanceFromHome_km debido a que ya no son útiles
EmpleadosAttrition.drop(columns=['Year', 'HiringDate', 'DistanceFromHome_km'], inplace=True)

# Mostrar los primeros registros del DataFrame
EmpleadosAttrition.head()
```

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

5 rows × 26 columns

```
# 11. Aprovechando los ajustes que se están haciendo, la empresa desea saber si todos los departamentos tienen un ingreso promedio similar. G
# Se crea un nuevo DataFrame con el ingreso mensual promedio por departamento
SueldoPromedioDepto = EmpleadosAttrition.groupby('Department')['MonthlyIncome'].mean().reset_index()

# Se renombra la columna de ingreso promedio
SueldoPromedioDepto.rename(columns={'MonthlyIncome': 'SueldoPromedio'}, inplace=True)

# Imprime el nuevo dataframe
```

```
SueldoPromedioDepto.head()
```



	Department	SueldoPromedio
0	Human Resources	6239.888889
1	Research & Development	6804.149813
2	Sales	7192.609756




Próximos pasos:

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

```
# 12. La variable MonthlyIncome tiene un valor numérico muy grande comparada con las otras variables. Escala dicha variable para que tenga un
# Se crea una instancia de MinMaxScaler
scaler = MinMaxScaler()
```


```
# Se ajusta y transformar la columna 'MonthlyIncome'
EmpleadosAttrition['MonthlyIncome'] = scaler.fit_transform(EmpleadosAttrition[['MonthlyIncome']])
```

```
# 13. Todo parece indicar que las variables categóricas que quedan sí son importantes para obtener la variable de salida. Convierte todas las
#a) BusinessTravel
#b) Department
#c) EducationField
#d) Gender
#e) JobRole
#f) MaritalStatus
#g) Attrition
```

```
# Columnas categóricas
categorical_columns = ['BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', 'Attrition']
```

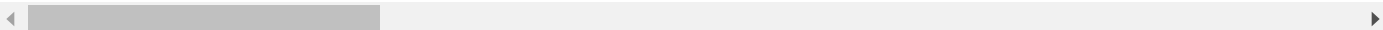
```
# Se aplica LabelEncoder a cada columna categórica
label_encoders = {}
for col in categorical_columns:
    le = LabelEncoder()
    EmpleadosAttrition[col] = le.fit_transform(EmpleadosAttrition[col])
```

```
EmpleadosAttrition.head()
```



	Age	BusinessTravel	Department	Education	EducationField	EnvironmentSatisfaction	Gender	JobInvolvement	JobLevel	JobRole	...	P
0	50	2	1	2	3	4	1	3	4	5	...	
1	36	2	1	2	3	2	1	3	2	4	...	
2	21	2	2	1	2	2	1	3	1	8	...	
3	52	2	1	4	1	2	1	3	3	0	...	
4	33	2	1	1	3	2	1	3	3	3	...	

5 rows × 26 columns



```
# 14. Ahora debes hacer la evaluación de las variables para quedarte con las mejores. Calcula la correlación lineal de cada una de las varia
# Se calcula la correlación de todas las variables numéricas con respecto a 'Attrition'
# Se seleccionan solo las columnas numéricas para el cálculo de la correlación
numeric_columns = EmpleadosAttrition.select_dtypes(include=['number']).columns
```

```
# Se calcula la correlación solo para las columnas numéricas
correlations = EmpleadosAttrition[numeric_columns].corr()
```

```
# 15. Selecciona solo aquellas variables que tengan una correlación mayor o igual a 0.1, dejándolas en otro frame llamado EmpleadosAttrition
# Filtrar las variables con correlación >= 0.1 con Attrition
selected_columns = correlations.index[correlations['Attrition'].abs() >= 0.1].tolist()
```

```
# Crear el nuevo DataFrame con las variables seleccionadas
EmpleadosAttritionFinal = EmpleadosAttrition[selected_columns]
```

```
EmpleadosAttritionFinal.head()
```

	Age	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	MaritalStatus	MonthlyIncome	TotalWorkingYears	YearsInCurrentCompany
0	50	4	3	4	4	0	0.864269	32	
1	36	2	3	2	2	0	0.207340	7	
2	21	2	3	1	2	2	0.088062	1	
3	52	2	3	3	2	2	0.497574	18	
4	33	2	2	2	2	1	0.664470	15	

Próximos pasos:

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

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

# 16. Crea una nueva variable llamada EmpleadosAttritionPCA formada por los componentes principales del frame EmpleadosAttritionFinal. Recuerda que se separan las características y la variable objetivo

```
X = EmpleadosAttritionFinal.drop(columns=['Attrition']) # Variables predictoras
y = EmpleadosAttritionFinal['Attrition'] # Variable objetivo
```

```
# Se escalan las características
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Se aplica PCA
pca = PCA()
EmpleadosAttritionPCA = pca.fit_transform(X_scaled)
```

```
# Se verifican las primeras filas del resultado
print("Componentes principales (primeras filas):")
EmpleadosAttritionPCA[:,0]
```

```
2.30547382e+00, 3.66425348e+00, 2.15102504e+00, -7.59914013e-01,
2.56705853e+00, 1.81339216e+00, -2.59125803e+00, -1.52066392e+00,
-1.28213218e+00, 5.01262194e+00, -1.79780918e+00, 1.14100702e+00,
-1.79639554e+00, 4.42507665e+00, -1.07912372e+00, -2.15526210e+00,
```

```
-1.16582497e+00,  2.85090370e+00,  1.90354244e+00, -1.85513489e+00,
 1.71840404e+00,  6.56523270e-01, -2.04141491e+00, -1.98554494e+00,
 2.05918181e+00, -2.30386784e+00,  6.02755892e-01,  3.64788679e+00,
-5.85761590e-01,  4.20830240e+00, -1.42145105e+00, -1.26854867e+00,
-1.09199652e+00, -6.26944000e-01, -7.78184340e-01, -1.98177317e+00,
-2.13523093e+00,  2.73548161e+00, -2.31566320e+00, -1.24204000e+00,
-9.83553488e-01, -1.23790585e+00,  1.02209940e+00, -1.55002016e+00,
 6.58188848e-03,  7.20247512e-03, -4.29751703e-01, -1.01013047e+00,
 1.52384656e-01, -3.33322828e-01, -1.76421314e+00, -5.20060881e-01,
 3.06341400e+00, -1.66941331e+00, -1.50593215e+00, -1.52347721e+00,
 1.24925599e+00, -2.02768798e+00, -1.06891866e+00]]
```

# 17. Agrega el mínimo número de Componentes Principales en columnas del frame EmpleadosAttritionPCA que logren explicar el 80% de la varianz

# Se determina el número mínimo de componentes que explican al menos el 80% de la varianza

```
varianza_acumulada = pca.explained_variance_ratio_.cumsum()
n_componentes = (varianza_acumulada >= 0.8).argmax() + 1
```

# Se crea un DataFrame con los componentes principales seleccionados

```
componentes_seleccionados = pd.DataFrame(
    EmpleadosAttritionPCA[:, :n_componentes],
    columns=[f"C{i}" for i in range(n_componentes)]
)
```

# Se Agregan los componentes principales al DataFrame original

```
EmpleadosAttritionFinal = EmpleadosAttritionFinal.assign(**componentes_seleccionados)
```

# 18. Guarda el set de datos que has formado y que tienes en EmpleadosAttritionFinal en un archivo CSV llamado EmpleadosAttritionFinal.csv. L

# Guardar el DataFrame en un archivo CSV

```
EmpleadosAttritionFinal.to_csv("EmpleadosAttritionFinal.csv", index=False)
```

```
print("El archivo 'EmpleadosAttritionFinal.csv' ha sido guardado con éxito.")
```

```
→ El archivo 'EmpleadosAttritionFinal.csv' ha sido guardado con éxito.
```