In [43]:
```
# 1. Importar las librerías requeridas.
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
import warnings
```

In [44]:
```
# 2. Leer el archivo CSV y cargarlo en un DataFrame.
EmpleadosAttrition = pd.read_csv('empleadosRETO.csv')
print("Primeras 5 filas del DataFrame original:")
EmpleadosAttrition.head()
```

```
Primeras 5 filas del DataFrame original:
```

Out[44]:

	Age	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCount
0	50	Travel_Rarely	Research & Development	1 km	2	Medical	1
1	36	Travel_Rarely	Research & Development	6 km	2	Medical	1
2	21	Travel_Rarely	Sales	7 km	1	Marketing	1
3	52	Travel_Rarely	Research & Development	7 km	4	Life Sciences	1
4	33	Travel_Rarely	Research & Development	15 km	1	Medical	1

5 rows × 30 columns

In [45]:
```
# 3. Eliminar columnas que no aportan información.
# Estas columnas tienen el mismo valor para todos los empleados o son identificadores únicos.
columns_to_drop = ['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours']
EmpleadosAttrition.drop(columns=columns_to_drop, inplace=True)
print("DataFrame después de eliminar columnas no relevantes:")
EmpleadosAttrition.head()
```

```
DataFrame después de eliminar columnas no relevantes:
```

Out[45]:

	Age	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EnvironmentSatis
0	50	Travel_Rarely	Research & Development	1 km	2	Medical	
1	36	Travel_Rarely	Research & Development	6 km	2	Medical	
2	21	Travel_Rarely	Sales	7 km	1	Marketing	
3	52	Travel_Rarely	Research & Development	7 km	4	Life Sciences	
4	33	Travel_Rarely	Research & Development	15 km	1	Medical	

5 rows × 26 columns

In [46]:
```
# 4. Calcular los años que el empleado lleva en la compañía.
# a. Crear la columna 'Year' a partir de 'HiringDate'
EmpleadosAttrition.loc[229, 'HiringDate']='2/28/2012'  # Corregir valor erróneo
EmpleadosAttrition['HiringDate'] = pd.to_datetime(EmpleadosAttrition['HiringDate'], format='%r
EmpleadosAttrition['Year'] = EmpleadosAttrition['HiringDate'].dt.year
```

```
# b. Crear la columna 'YearsAtCompany' hasta el año 2018
EmpleadosAttrition['YearsAtCompany'] = 2018 - EmpleadosAttrition['Year']
print("Columnas 'HiringDate', 'Year' y la nueva 'YearsAtCompany':")
EmpleadosAttrition[['HiringDate', 'Year', 'YearsAtCompany']].head()
```

```
Columnas 'HiringDate', 'Year' y la nueva 'YearsAtCompany':
```

Out[46]:

	HiringDate	Year	YearsAtCompany
0	2013-06-06	2013	5
1	2015-12-25	2015	3
2	2017-02-14	2017	1
3	2010-07-29	2010	8
4	2011-10-07	2011	7

In [47]:

```
# 5. Limpiar y convertir la columna de distancia desde casa.
# a. Renombrar la variable DistanceFromHome a DistanceFromHome_km.
EmpleadosAttrition.rename(columns={'DistanceFromHome': 'DistanceFromHome_km'}, inplace=True)

# b. Crear una nueva variable DistanceFromHome que sea entera.
EmpleadosAttrition['DistanceFromHome'] = EmpleadosAttrition['DistanceFromHome_km'].str.replace
print("Columnas de distancia procesadas:")
EmpleadosAttrition[['DistanceFromHome_km', 'DistanceFromHome']].head()
```

```
Columnas de distancia procesadas:
```

Out[47]:

	DistanceFromHome_km	DistanceFromHome
0	1 km	1
1	6 km	6
2	7 km	7
3	7 km	7
4	15 km	15

In [48]:

```
# 6. Borrar columnas auxiliares que ya no son útiles.
EmpleadosAttrition.drop(columns=['Year', 'HiringDate', 'DistanceFromHome_km'], inplace=True)
print("DataFrame después de borrar columnas auxiliares:")
EmpleadosAttrition.head()
```

```
DataFrame después de borrar columnas auxiliares:
```

Out[48]:

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

5 rows × 26 columns

In [49]:
```
# 7. Generar tabla informativa de sueldo promedio por departamento.
SueldoPromedioDepto = EmpleadosAttrition.groupby('Department')['MonthlyIncome'].mean().reset_i
SueldoPromedioDepto.rename(columns={'MonthlyIncome': 'SueldoPromedio'}, inplace=True)
print("--- Sueldo Promedio por Departamento ---")
print(SueldoPromedioDepto)
```

```
--- Sueldo Promedio por Departamento ---
              Department  SueldoPromedio
0         Human Resources     6239.888889
1  Research & Development     6804.149813
2                   Sales     7188.250000
```

In [ ]:
```
# 8. Escalar la variable 'MonthlyIncome' a un rango entre 0 y 1.
scaler = MinMaxScaler()
EmpleadosAttrition['MonthlyIncome'] = scaler.fit_transform(EmpleadosAttrition[['MonthlyIncome'
print("Columna 'MonthlyIncome' escalada:")
EmpleadosAttrition[['MonthlyIncome']].head()
```

```
Columna 'MonthlyIncome' escalada:
```

Out[ ]:

	MonthlyIncome
0	0.864269
1	0.207340
2	0.088062
3	0.497574
4	0.664470

In [ ]:
```
# 9. Convertir todas las variables categóricas restantes a numéricas.
# Se convierten las variables binarias usando map
EmpleadosAttrition['Attrition'] = EmpleadosAttrition['Attrition'].map({'Yes': 1, 'No': 0})
EmpleadosAttrition['Gender'] = EmpleadosAttrition['Gender'].map({'Male': 1, 'Female': 0})
EmpleadosAttrition['OverTime'] = EmpleadosAttrition['OverTime'].map({'Yes': 1, 'No': 0})

# Se convierten las variables con múltiples categorías usando one-hot encoding
categorical_cols_to_encode = ['BusinessTravel', 'Department', 'EducationField', 'JobRole', 'Ma
EmpleadosAttrition = pd.get_dummies(EmpleadosAttrition, columns=categorical_cols_to_encode, dr

print("DataFrame con variables categóricas convertidas a numéricas:")
EmpleadosAttrition.head()
```

```
DataFrame con variables categóricas convertidas a numéricas:
```

Out[ ]:

	Age	Education	EnvironmentSatisfaction	Gender	JobInvolvement	JobLevel	JobSatisfaction	Mont
0	50	2	4	1	3	4	4	
1	36	2	2	1	3	2	2	
2	21	1	2	1	3	1	2	
3	52	4	2	1	3	3	2	
4	33	1	2	1	3	3	3	

5 rows × 40 columns

In [ ]:
```
# 10. Calcular la correlación lineal de cada variable con 'Attrition'
correlation_matrix = EmpleadosAttrition.corr()
# Se usa .abs() para considerar tanto correlaciones positivas como negativas.
correlation_with_attrition = correlation_matrix['Attrition'].abs().sort_values(ascending=False
print("--- Correlación de las variables con Attrition (en valor absoluto) ---")
```

```
print(correlation_with_attrition)
```

```
--- Correlación de las variables con Attrition (en valor absoluto) ---
Attrition                              1.000000
OverTime                               0.324777
JobLevel                               0.214266
TotalWorkingYears                      0.213329
Age                                    0.212121
MaritalStatus_Single                   0.205849
YearsInCurrentRole                     0.203918
MonthlyIncome                          0.194936
JobRole_Sales Representative           0.191294
YearsAtCompany                         0.176001
JobInvolvement                         0.166785
JobSatisfaction                        0.164957
EducationField_Technical Degree        0.129104
JobRole_Laboratory Technician          0.125264
EnvironmentSatisfaction                0.124327
JobRole_Research Director              0.116263
MaritalStatus_Married                  0.094734
JobRole_Manager                        0.089885
Department_Research & Development      0.072269
TrainingTimesLastYear                  0.070884
YearsSinceLastPromotion                0.069000
Department_Sales                       0.066116
PercentSalaryHike                      0.060880
Education                              0.055531
EducationField_Medical                 0.054144
DistanceFromHome                       0.052732
BusinessTravel_Travel_Rarely           0.042755
JobRole_Manufacturing Director         0.042404
BusinessTravel_Travel_Frequently       0.035387
JobRole_Human Resources                0.032714
RelationshipSatisfaction               0.030945
Gender                                 0.028839
EducationField_Life Sciences           0.027457
WorkLifeBalance                        0.021723
EducationField_Marketing               0.016768
NumCompaniesWorked                     0.009082
JobRole_Research Scientist             0.007977
PerformanceRating                      0.006471
EducationField_Other                   0.004275
JobRole_Sales Executive                0.003115
Name: Attrition, dtype: float64
```

In [54]:
```
# 11. Seleccionar variables con correlación >= 0.1.
highly_correlated_vars = correlation_with_attrition[correlation_with_attrition >= 0.1]
final_columns = highly_correlated_vars.index.tolist()
EmpleadosAttritionFinal = EmpleadosAttrition[final_columns]

print(f"Se seleccionaron {len(EmpleadosAttritionFinal.columns)} variables de un total de {len(
print("\nVariables finales seleccionadas:")
print(EmpleadosAttritionFinal.columns)
EmpleadosAttritionFinal.head()
```

```
Se seleccionaron 16 variables de un total de 40.

Variables finales seleccionadas:
Index(['Attrition', 'OverTime', 'JobLevel', 'TotalWorkingYears', 'Age',
       'MaritalStatus_Single', 'YearsInCurrentRole', 'MonthlyIncome',
       'JobRole_Sales Representative', 'YearsAtCompany', 'JobInvolvement',
       'JobSatisfaction', 'EducationField_Technical Degree',
       'JobRole_Laboratory Technician', 'EnvironmentSatisfaction',
       'JobRole_Research Director'],
      dtype='object')
```

Out[54]:

	Attrition	OverTime	JobLevel	TotalWorkingYears	Age	MaritalStatus_Single	YearsInCurrentRole	M
0	0	0	4	32	50	False	4	
1	0	0	2	7	36	False	2	
2	1	0	1	1	21	True	0	
3	0	0	3	18	52	True	6	
4	1	1	3	15	33	False	6	

In [55]:
```
# 12. Crear una variable con los Componentes Principales (PCA).
# Se excluye la variable objetivo 'Attrition' del análisis PCA
X_final = EmpleadosAttritionFinal.drop('Attrition', axis=1)

pca = PCA()
EmpleadosAttritionPCA = pca.fit_transform(X_final)

print(f"Dimensiones del array resultante de PCA: {EmpleadosAttritionPCA.shape}")
```

```
Dimensiones del array resultante de PCA: (400, 15)
```

In [ ]:
```
# 13. Agregar los Componentes Principales que explican el 80% de la varianza.
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)

# Encontrar el número mínimo de componentes para alcanzar el 80%
n_components_80 = np.where(cumulative_variance >= 0.80)[0][0] + 1
print(f"Número de componentes para explicar al menos el 80% de la varianza: {n_components_80}"
print("Varianza acumulada por componente:", cumulative_variance)

# Agregar los componentes como nuevas columnas C0, C1...
for i in range(n_components_80):
    col_name = f'C{i}'
    EmpleadosAttritionFinal = EmpleadosAttritionFinal.assign(**{col_name: EmpleadosAttritionPC

print("\nDataFrame final con los componentes principales agregados:")
EmpleadosAttritionFinal.head()
```

```
Número de componentes para explicar al menos el 80% de la varianza: 2
Varianza acumulada por componente: [0.63567076 0.87769527 0.95671221 0.97776996 0.98429511 0.9
9071934
 0.99370518 0.9964977  0.99755473 0.99853648 0.99916131 0.99947423
 0.99975114 0.99997374 1.        ]

DataFrame final con los componentes principales agregados:
```

Out[ ]:

	Attrition	OverTime	JobLevel	TotalWorkingYears	Age	MaritalStatus_Single	YearsInCurrentRole	M
0	0	0	4	32	50	False	4	
1	0	0	2	7	36	False	2	
2	1	0	1	1	21	True	0	
3	0	0	3	18	52	True	6	
4	1	1	3	15	33	False	6	

In [68]:
```
# Mover la columna 'Attrition' al final del DataFrame
columnas = [col for col in EmpleadosAttritionFinal.columns if col != 'Attrition'] + ['Attritic
EmpleadosAttritionFinal = EmpleadosAttritionFinal[columnas]

print("\nDataFrame con la columna 'Attrition' reordenada al final:")
```

```
EmpleadosAttritionFinal.head()
```

DataFrame con la columna 'Attrition' reordenada al final:

Out[68]:

	OverTime	JobLevel	TotalWorkingYears	Age	MaritalStatus_Single	YearsInCurrentRole	MonthlyInco
0	0	4	32	50	False	4	0.864
1	0	2	7	36	False	2	0.207
2	0	1	1	21	True	0	0.088
3	0	3	18	52	True	6	0.497
4	1	3	15	33	False	6	0.664

In [69]:

```
# 14. Guardar el set de datos final en un archivo CSV.
EmpleadosAttritionFinal.to_csv('EmpleadosAttritionFinal.csv', index=False)
print("Archivo 'EmpleadosAttritionFinal.csv' guardado exitosamente.")
```

```
Archivo 'EmpleadosAttritionFinal.csv' guardado exitosamente.
```