```
import pandas as pd
import numpy as np
import re
import os
import csv
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, StandardScaler
from sklearn.decomposition import PCA
from sklearn.impute import SimpleImputer
```

```
EmpleadosAttrition = pd.read_csv(r'C:\Users\rahul\OneDrive\Escritorio\Proyectos Data Science\empleadosRETO.csv')
```

```
cols_irrelevantes = ["EmployeeCount", "EmployeeNumber", "Over18", "StandardHours"]
EmpleadosAttrition.drop(columns=cols_irrelevantes, inplace=True, errors="ignore")
```

```
if "HiringDate" in EmpleadosAttrition.columns:
    # Convertir a formato de fecha (los inválidos se vuelven NaT)
    EmpleadosAttrition["HiringDate"] = pd.to_datetime(EmpleadosAttrition["HiringDate"], errors="coerce")
    # Extraer el año de contratación
    EmpleadosAttrition["Year"] = EmpleadosAttrition["HiringDate"].dt.year.astype("Int64")
    # Cálculo de años en la compañía al 2018
    EmpleadosAttrition["YearsAtCompany"] = (2018 - EmpleadosAttrition["Year"]).astype("Int64")
    print("Columnas 'Year' y 'YearsAtCompany' creadas correctamente.")

else:
    print("No existe la columna HiringDate en el dataset.")
```

```
Columnas 'Year' y 'YearsAtCompany' creadas correctamente.
```

```
if "DistanceFromHome" in EmpleadosAttrition.columns:
    # Renombrar columna original
    EmpleadosAttrition.rename(columns={"DistanceFromHome": "DistanceFromHome_km"}, inplace=True)

    # Función para limpiar valores tipo '10 km' a 10
    def convertir_km_a_entero(valor):
        if pd.isna(valor):
            return np.nan
        valor = str(valor).lower().replace("km", "").strip()
        valor = valor.replace(",", ".")
        valor = re.sub(r"[^0-9.]", "", valor)
        try:
            return int(float(valor))
        except:
            return np.nan

    # Crear nueva columna numérica limpia
    EmpleadosAttrition["DistanceFromHome"] = EmpleadosAttrition["DistanceFromHome_km"].apply(convertir_km_a_entero).astype("Int
```

```
EmpleadosAttrition.head()
```

	Age	BusinessTravel	Department	DistanceFromHome_km	Education	EducationField	EnvironmentSatisfaction	Gender	JobInvolvem
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	

5 rows × 29 columns

```
EmpleadosAttrition.drop(columns=["Year", "HiringDate", "DistanceFromHome_km"], inplace=True, errors="ignore")
```

```
EmpleadosAttrition.head()
```

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

5 rows × 26 columns

```
if "MonthlyIncome" in EmpleadosAttrition.columns and "Department" in EmpleadosAttrition.columns:
    EmpleadosAttrition["MonthlyIncome"] = pd.to_numeric(EmpleadosAttrition["MonthlyIncome"], errors="coerce")
    SueldoPromedioDepto = EmpleadosAttrition.groupby("Department")["MonthlyIncome"].mean().reset_index()
    SueldoPromedioDepto.rename(columns={"MonthlyIncome": "SueldoPromedio"}, inplace=True)
    SueldoPromedio = SueldoPromedioDepto.copy()
else:
    SueldoPromedioDepto = pd.DataFrame()
    SueldoPromedio = pd.DataFrame()
```

```
if "MonthlyIncome" in EmpleadosAttrition.columns:
    scaler = MinMaxScaler()
    imputer = SimpleImputer(strategy="median")  # rellenar NaN para escalar
    ingresos = imputer.fit_transform(EmpleadosAttrition[["MonthlyIncome"]])
    EmpleadosAttrition["MonthlyIncome_scaled"] = scaler.fit_transform(ingresos)
```

```
for col in ["BusinessTravel", "Department", "EducationField", "Gender", "JobRole", "MaritalStatus"]:
    if col in EmpleadosAttrition.columns:
        EmpleadosAttrition[col] = LabelEncoder().fit_transform(EmpleadosAttrition[col])

# Convertir Attrition a 0/1 directamente
EmpleadosAttrition["Attrition"] = EmpleadosAttrition["Attrition"].map({"Yes": 1, "No": 0})
```

```
numeric_df = EmpleadosAttrition.select_dtypes(include=[np.number]).fillna(0)
correlaciones = numeric_df.corr()["Attrition"].drop("Attrition")
tabla_corr = correlaciones.abs().sort_values(ascending=False).reset_index()
tabla_corr.columns = ["Variable", "Correlacion_Abs"]
```

```
seleccionadas = tabla_corr[tabla_corr["Correlacion_Abs"] >= 0.1]["Variable"].tolist()
EmpleadosAttritionFinal = EmpleadosAttrition[["Attrition"] + seleccionadas]
```

```
X = EmpleadosAttritionFinal.drop(columns=["Attrition"], errors="ignore")

if X.shape[1] == 0:
    EmpleadosAttritionPCA = np.array([])
    n_componentes = 0
    print("No hay features numéricas para PCA.")
else:
    # Imputar y estandarizar (necesario antes de PCA)
    X_imp = SimpleImputer(strategy="median").fit_transform(X)
    X_scaled = StandardScaler().fit_transform(X_imp)

    # 3) Ejecutar PCA
    pca = PCA(n_components=min(X_scaled.shape[0], X_scaled.shape[1]))
    X_pca = pca.fit_transform(X_scaled)
    EmpleadosAttritionPCA = X_pca

    # 4) Número mínimo de componentes para >= 80% de la varianza
    cum_var = np.cumsum(pca.explained_variance_ratio_)
    n_componentes = int(np.searchsorted(cum_var, 0.80) + 1)
    n_componentes = min(n_componentes, X_pca.shape[1])

    # 5) Asegurar de trabajar sobre una copia independiente para evitar vistas
    EmpleadosAttritionFinal = EmpleadosAttritionFinal.copy()
```

```
    # 6) Asignar las componentes con .loc (una línea por componente)
    for i in range(n_componentes):
        EmpleadosAttritionFinal.loc[:, f"C{i}"] = EmpleadosAttritionPCA[:, i]

    print(f"PCA: añadidos {n_componentes} componentes (C0..C{n_componentes-1}).")
```

```
PCA: añadidos 5 componentes (C0..C4).
```

```
csv_utf8 = "EmpleadosAttritionFinal.csv"
EmpleadosAttritionFinal.to_csv(csv_utf8, index=False, encoding="utf-8-sig")
print(f"Guardado CSV (utf-8-sig): {os.path.abspath(csv_utf8)}")
```

```
Guardado CSV (utf-8-sig): C:\Users\rahul\EmpleadosAttritionFinal.csv
```