```
# %% [markdown]
# 1.Importa las librerías requeridas.
# 2.Lee el archivo CSV llamado empleadosRETO.csv y coloca los datos en un frame de Pandas llamado EmpleadosAttrition.
#

# %%
import pandas as pd
import numpy as np
import datetime

EmpleadosAttrition= pd.read_csv('EmpleadosRETO.csv')

# Mostrar el contenido del DataFrame
print(EmpleadosAttrition)

# %% [markdown]
# 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:
# a.EmployeeCount: número de empleados, todos tienen un 1
# b.EmployeeNumber: ID del empleado, el cual es único para cada empleado
# c.Over18: mayores de edad, todos dicen “Y”
# d.StandardHours: horas de trabajo, todos tienen “80”

# %%
columns_drop = ['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours']
EmpleadosAttrition = EmpleadosAttrition.drop(columns=columns_drop, axis=1)

# Muestra el DataFrame después de eliminar las columnas
print(EmpleadosAttrition)

# %% [markdown]
# 4.Analiza la información proporcionada, si detectaste que no se cuenta con los años que el empelado lleva en la compañía y parece ser un buen dato. Dicha cantidad se puede calcular con la fecha de contratación ‘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 ser un entero.

# %%
EmpleadosAttrition['Year']=EmpleadosAttrition['HiringDate'].str[-4:].astype(int)
print(EmpleadosAttrition['Year'])

# %% [markdown]
```

```
# 6.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['YearatCompany']= 2018-EmpleadosAttrition['Year']
print(EmpleadosAttrition['YearatCompany'])


# %% [markdown]
# 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=EmpleadosAttrition.rename(columns={'DistanceFromHome':'DistanceFromHome_Km'})
EmpleadosAttrition['DistanceFromHome_Km']

# %% [markdown]
# 9.Crea una nueva variable DistanceFromHome que sea entera, es decir, solo con números.
#

# %%
EmpleadosAttrition['DistanceFromHome']=EmpleadosAttrition['DistanceFromHome_Km'].str[:2].astype(int)
print(EmpleadosAttrition['DistanceFromHome'])

# %% [markdown]
# 10.Borra las columnas Year, HiringDate y DistanceFromHome_km debido a que ya no son útiles.

# %%
EmpleadosAttrition.drop(['Year','HiringDate','DistanceFromHome_Km'], axis=1, inplace=True)
print(EmpleadosAttrition.columns)

# %% [markdown]
# 11.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.
```

```
# %%
SueldoPromDepto=EmpleadosAttrition[['MonthlyIncome','Department']]

SueldoPromedio=SueldoPromDepto.groupby('Department').mean()
print(SueldoPromedio)


# %% [markdown]
# 12. 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.

# %%
#print(EmpleadosAttrition['MonthlyIncome'])
EmpleadosAttrition['MonthlyIncome_Norm']=EmpleadosAttrition['MonthlyIncome']/EmpleadosAttrition['MonthlyIncome'].abs().max()

print(EmpleadosAttrition['MonthlyIncome_Norm'])

# %% [markdown]
# 13.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

# %%
EmpleadosAttrition_transform = EmpleadosAttrition.loc[:,['BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','Attrition','OverTime']]

from sklearn.preprocessing import LabelBinarizer
col1 = LabelBinarizer()
```

```
col2 = LabelBinarizer()
col3 = LabelBinarizer()
col4 = LabelBinarizer()
col5 = LabelBinarizer()
col6 = LabelBinarizer()
col7 = LabelBinarizer()

EmpleadosAttrition['BusinessTravel'] =
col1.fit_transform(EmpleadosAttrition_transform['BusinessTravel'].astype(str))
EmpleadosAttrition['Department'] =
col2.fit_transform(EmpleadosAttrition_transform['Department'].astype(str))
EmpleadosAttrition['EducationField'] =
col3.fit_transform(EmpleadosAttrition_transform['EducationField'].astype(str))
EmpleadosAttrition['Gender'] =
col4.fit_transform(EmpleadosAttrition_transform['Gender'].astype(str))
EmpleadosAttrition['JobRole'] =
col5.fit_transform(EmpleadosAttrition_transform['JobRole'].astype(str))
EmpleadosAttrition['MaritalStatus'] =
col6.fit_transform(EmpleadosAttrition_transform['MaritalStatus'].astype(str))
EmpleadosAttrition['Attrition'] =
col7.fit_transform(EmpleadosAttrition_transform['Attrition'].astype(str))
EmpleadosAttrition['OverTime'] =
col7.fit_transform(EmpleadosAttrition_transform['OverTime'].astype(str))


# %% [markdown]
# 14. 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.

# %%
daset_corr=EmpleadosAttrition.corr()

daset_corr=daset_corr['Attrition']

daset_corr




# %% [markdown]
# Testing Manual de la correlacion entre las variables de lagunas

# %%
```

```
print('\nAge\n',EmpleadosAttrition['Age'].corr(EmpleadosAttrition['Attrition']))
print('\nBusiness
Travel\n',EmpleadosAttrition['BusinessTravel'].corr(EmpleadosAttrition['Attrition']))
print('\nDepartment\n',EmpleadosAttrition['Department'].corr(EmpleadosAttrition['Att
rition']))
print('\nEducationField\n',EmpleadosAttrition['EducationField'].corr(EmpleadosAttriti
on['Attrition']))
print('\nGender\n',EmpleadosAttrition['Gender'].corr(EmpleadosAttrition['Attrition']))
print('\nJobRole\n',EmpleadosAttrition['JobRole'].corr(EmpleadosAttrition['Attrition'])
)
print('\nMaritalStatus\n',EmpleadosAttrition['MaritalStatus'].corr(EmpleadosAttrition['
Attrition']))

# %% [markdown]
# 15. 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 salidaAttrition; esto es equivalente a borrar las que no
cumplen con el límite.

# %%
EmpleadosAttritionFinal = EmpleadosAttrition[['Age', 'JobInvolvement', 'JobLevel',
'JobRole', 'JobSatisfaction', 'MaritalStatus', 'MonthlyIncome', 'OverTime',
'TotalWorkingYears', 'YearsInCurrentRole', 'Attrition','YearatCompany']]

EmpleadosAttritionFinal.corr()

EmpleadosAttritionFinal

# %% [markdown]
# 16.   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]).

# %%
EmpleadosAttritionPCA = EmpleadosAttritionFinal

EmpleadosAttritionPCA

# %% [markdown]
# 17. 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 unaC0,
C1, etc., hasta las que vayas a agregar.

# %%
from sklearn.decomposition import PCA

# Creamos una instancia del modelo PCA y ajustamos los datos
pca = PCA(n_components=.8)
pca.fit(EmpleadosAttritionPCA)

# Obtenemos las componentes principales y su varianza explicada
components = pca.components_
variance = pca.explained_variance_ratio_

# Imprimimos los resultados
print(f"Las componentes principales son: {components}")
print(f"La varianza explicada por cada componente es: {variance}")

# %%
PCA_transform=pca.transform(EmpleadosAttritionPCA)

PCA_Columnas = pd.DataFrame(PCA_transform.reshape(-1,
PCA_transform.shape[-1]), columns=[f'PC{i+1}' for i in
range(PCA_transform.shape[-1])])
PCA_Columnas

# %%
PCA_union =
pd.concat([EmpleadosAttritionPCA,PCA_Columnas],ignore_index=True, axis=1)
EmpleadosAttritionFinal=PCA_union
EmpleadosAttritionFinal

# %% [markdown]
# 18. Guarda el set de datos que has formado y que tienes en
EmpleadosAttritionFinal en un archivo CSV llamado EmpleadosAttritionFinal.csv.
Las últimas columnas que colocaste quedarán después de la variable Attrition, lo
cual no importa, pero si gustas lo puedes arreglar antes de escribir el archivo.

# %%
EmpleadosAttritionFinal.to_csv('EmpleadosAttritionFinal.csv', index=False)
```