1. Importa las librerías requeridas.

```
import pandas as pd
from sklearn import preprocessing
from sklearn.decomposition import PCA
```

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

```
url = 'https://drive.google.com/file/d/1ONo7sxRXXP7u2ycvZYbxR0WyUGrg_4US/view?usp=sharing'
path = 'https://drive.google.com/uc?export=download&confirm=1&id='+url.split('/')[-2]
EmpleadosAttrition = pd.read_csv(path)
```

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”

```
EmpleadosAttrition.drop(['EmployeeCount','EmployeeNumber','Over18','StandardHours'],axis=1,inplace=True)
```

4. Analizando la información proporcionada, 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’:

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']
```

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.rstrip(' km').astype(int)
```

6. 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)
```

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 = pd.DataFrame()
SueldoPromedioDepto['SueldoPromedio'] = EmpleadosAttrition.groupby(['Department'])['MonthlyIncome'].mean()
SueldoPromedioDepto.reset_index()
```

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

2 Sales 7188.250000

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.

```
escalador = preprocessing.MinMaxScaler()
frame_escalado = escalador.fit_transform(EmpleadosAttrition[['Age','MonthlyIncome']])
EmpleadosAttrition['MonthlyIncome'] = pd.Series(frame_escalado[:,1])
```

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

```
df1 = pd.get_dummies(EmpleadosAttrition.BusinessTravel, prefix = 'BusinessTravel',dtype=int)
df2 = pd.get_dummies(EmpleadosAttrition.Department, prefix = 'Department',dtype=int)
df3 = pd.get_dummies(EmpleadosAttrition.EducationField, prefix = 'EducationField',dtype=int)
df4 = pd.get_dummies(EmpleadosAttrition.Gender, prefix = 'Gender',dtype=int)
df5 = pd.get_dummies(EmpleadosAttrition.JobRole, prefix = 'JobRole',dtype=int)
df6 = pd.get_dummies(EmpleadosAttrition.MaritalStatus, prefix = 'MaritalStatus',dtype=int)
df6 = pd.get_dummies(EmpleadosAttrition.OverTime, prefix = 'OverTime',dtype=int)
df7 = pd.get_dummies(EmpleadosAttrition.Attrition, prefix = 'Attrition',dtype=int)
EmpleadosAttrition = pd.concat([EmpleadosAttrition,df1,df2,df3,df4,df5,df6,df7], axis=1,sort=False)
EmpleadosAttrition.drop(['BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','OverTime','Attrition'],axis=1
EmpleadosAttrition.rename(columns={'Attrition_Yes':'Attrition'}, inplace=True)
```

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.

```
corrmatrix = EmpleadosAttrition.corr()['Attrition']
corrmatrix
```

```
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
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
OverTime_No                         -0.324777
OverTime_Yes                         0.324777
Attrition_No                        -1.000000
Attrition                            1.000000
Name: Attrition, 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.

```
filterlist = pd.DataFrame()
filterlist = corrmatrix.abs() >= 0.1
dict = {'Result': filterlist}
result = pd.DataFrame(dict)
print(result.loc[result['Result']==False])
```

```
                                    Result
Education                            False
NumCompaniesWorked                   False
PercentSalaryHike                    False
PerformanceRating                    False
RelationshipSatisfaction             False
TrainingTimesLastYear                False
WorkLifeBalance                      False
YearsSinceLastPromotion              False
DistanceFromHome                     False
BusinessTravel_Travel_Frequently     False
BusinessTravel_Travel_Rarely         False
Department_Human Resources           False
Department_Research & Development    False
Department_Sales                     False
EducationField_Human Resources       False
EducationField_Life Sciences         False
EducationField_Marketing             False
EducationField_Medical               False
EducationField_Other                 False
Gender_Female                        False
Gender_Male                          False
JobRole_Human Resources              False
JobRole_Manager                      False
JobRole_Manufacturing Director       False
JobRole_Research Scientist           False
JobRole_Sales Executive              False
```

```
EmpleadosAttritionFinal = EmpleadosAttrition.drop(['Education','NumCompaniesWorked','PercentSalaryHike','PerformanceRating','RelationshipS
                                                   'WorkLifeBalance','YearsSinceLastPromotion','DistanceFromHome','BusinessTravel_Travel_
                                                   'Department_Human Resources','Department_Research & Development','Department_Sales','E
                                                   'EducationField_Life Sciences','EducationField_Marketing','EducationField_Medical','Ed
                                                   'Gender_Male','JobRole_Human Resources','JobRole_Manager','JobRole_Manufacturing Direct
                                                   'JobRole_Sales Executive','OverTime_No','Attrition_No'],axis=1)
```

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]).

```
pca = PCA()
pca.fit(EmpleadosAttritionFinal)
EmpleadosAttritionPCA = pca.transform(EmpleadosAttritionFinal)
print(pca.explained_variance_ratio_)
```

```
[6.35326227e-01 2.41881046e-01 7.89623741e-02 2.10510826e-02
 6.52483876e-03 6.45034960e-03 2.99502469e-03 2.81329247e-03
 1.15391549e-03 6.78943720e-04 5.24543086e-04 4.48499527e-04
 4.04151769e-04 2.84159423e-04 2.55499588e-04 2.20170594e-04
 2.58823978e-05]
```

13. Agrega el mínimo número de Componentes Principales en columnas del frame EmpleadosAttritionPCA que logren explicar el 80% de la varianza al frame EmpleadosAttritionFinal.

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.

```
# 0=63.53% y 1=24.18% explican el 87% de la varianza
new_data = {'C0': EmpleadosAttritionPCA[:,0], 'C1': EmpleadosAttritionPCA[:,1]}
EmpleadosAttritionFinal = EmpleadosAttritionFinal.assign(**new_data)
```

14. 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)
```