```
import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje-/main

df.fillna(0,inplace=True)

print(df)

df.describe(include="all")

#Importamos el archivo
```

```
           ID      X1   X2   X3   X4    X5   X6   X7   X8   X9  ...      X15  \
0           1   20000  2.0  2.0  1.0  24.0  2.0  2.0 -1.0 -1.0  ...      0.0
1           2  120000  2.0  2.0  2.0  26.0 -1.0  2.0  0.0  0.0  ...   3272.0
2           3   90000  2.0  2.0  2.0  34.0  0.0  0.0  0.0  0.0  ...  14331.0
3           4   50000  2.0  2.0  1.0  37.0  0.0  0.0  0.0  0.0  ...  28314.0
4           5   50000  1.0  2.0  1.0  57.0 -1.0  0.0 -1.0  0.0  ...  20940.0
...       ...     ...  ...  ...  ...   ...  ...  ...  ...  ...  ...      ...
29995   29996  220000  1.0  3.0  1.0  39.0  0.0  0.0  0.0  0.0  ...  88004.0
29996   29997  150000  1.0  3.0  2.0  43.0 -1.0 -1.0 -1.0 -1.0  ...   8979.0
29997   29998   30000  1.0  2.0  2.0  37.0  4.0  3.0  2.0 -1.0  ...  20878.0
29998   29999   80000  1.0  3.0  1.0  41.0  1.0 -1.0  0.0  0.0  ...  52774.0
29999   30000   50000  1.0  2.0  1.0  46.0  0.0  0.0  0.0  0.0  ...  36535.0

            X16      X17      X18      X19      X20     X21      X22     X23  \
0           0.0      0.0      0.0    689.0      0.0     0.0      0.0     0.0
1        3455.0   3261.0      0.0   1000.0   1000.0  1000.0      0.0  2000.0
2       14948.0  15549.0   1518.0   1500.0   1000.0  1000.0   1000.0  5000.0
3       28959.0  29547.0   2000.0   2019.0   1200.0  1100.0   1069.0  1000.0
4       19146.0  19131.0   2000.0  36681.0  10000.0  9000.0    689.0   679.0
...         ...      ...      ...      ...      ...     ...      ...     ...
29995   31237.0  15980.0   8500.0  20000.0   5003.0  3047.0   5000.0  1000.0
```

```
df.info()
#Se identifica el tipo de dato
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ID      30000 non-null  int64  
 1   X1      30000 non-null  int64  
 2   X2      30000 non-null  float64
 3   X3      30000 non-null  float64
 4   X4      30000 non-null  float64
 5   X5      30000 non-null  float64
 6   X6      30000 non-null  float64
 7   X7      30000 non-null  float64
 8   X8      30000 non-null  float64
 9   X9      30000 non-null  float64
 10  X10     30000 non-null  float64
 11  X11     30000 non-null  float64
 12  X12     30000 non-null  float64
 13  X13     30000 non-null  float64
 14  X14     30000 non-null  float64
 15  X15     30000 non-null  float64
 16  X16     30000 non-null  float64
 17  X17     30000 non-null  float64
 18  X18     30000 non-null  float64
 19  X19     30000 non-null  float64
 20  X20     30000 non-null  float64
 21  X21     30000 non-null  float64
 22  X22     30000 non-null  float64
 23  X23     30000 non-null  float64
 24  Y       30000 non-null  float64
```

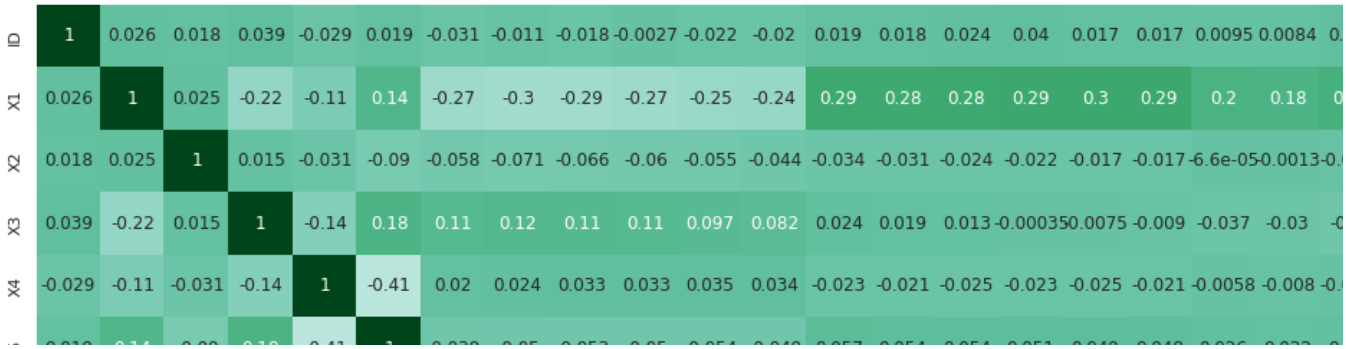
```
dtypes: float64(23), int64(2)
memory usage: 5.7 MB
```

```
df.corr()
```

ID	X1	X2	X3	X4	X5	X6	X7

```
import seaborn as sns
corrs = df.corr()
sns.set(rc = {'figure.figsize':(25,20)})
sns.heatmap(corrs, vmin = -1, vmax = 1, cmap = "BuGn", annot= True)
#Heatmap de correlacion
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff434c58d90>
```




```
t = df.var().sum()
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled = scaler.fit_transform(df)
scaled[:5]
#Trasnformamos los datos
```

```
array([[-1.73199307, -1.13672015,  0.81010432,  0.18606028, -1.05689531,
        -1.24379131,  1.79455033,  1.78231019, -0.69708132, -0.66663018,
        -1.5302844 , -1.4862636 , -0.64240886, -0.64728421, -0.66784304,
        -0.67231837, -0.66286347, -0.65264751, -0.34183413, -0.22706528,
        -0.29673029, -0.30802478, -0.31408022, -0.29337529,  1.87655989],
       [-1.7318776 , -0.3659805 ,  0.81010432,  0.18606028,  0.85846408,
        -1.02707083, -0.8750728 ,  1.78231019,  0.13911523,  0.18882817,
         0.23495311,  1.99241071, -0.659126  , -0.66663079, -0.63910542,
        -0.62146007, -0.60603717, -0.59789105, -0.34183413, -0.21356735,
        -0.23993143, -0.24419183, -0.31408022, -0.18087162,  1.87655989],
       [-1.73176213, -0.59720239,  0.81010432,  0.18606028,  0.85846408,
        -0.16018894,  0.01480158,  0.11165543,  0.13911523,  0.18882817,
         0.23495311,  0.25307356, -0.29847861, -0.49379012, -0.48226689,
        -0.4495646 , -0.41700543, -0.39155946, -0.25017819, -0.19186649,
        -0.23993143, -0.24419183, -0.24862689, -0.01211612, -0.53289   ],
       [-1.73164666, -0.90549825,  0.81010432,  0.18606028, -1.05689531,
         0.16489177,  0.01480158,  0.11165543,  0.13911523,  0.18882817,
         0.23495311,  0.25307356, -0.0574178 , -0.01320257,  0.03296287,
        -0.23221998, -0.18655874, -0.15651473, -0.22107532, -0.169341  ,
        -0.22857166, -0.23780854, -0.24411061, -0.23712346, -0.53289   ],
       [-1.73153119, -0.90549825, -1.23406505,  0.18606028, -1.05689531,
         2.33209651, -0.8750728 ,  0.11165543, -0.69708132,  0.18882817,
         0.23495311,  0.25307356, -0.57852795, -0.61120432, -0.16106298,
        -0.34683768, -0.34795859, -0.33141299, -0.22107532,  1.33504924,
         0.27125829,  0.26647176, -0.26898288, -0.25518029, -0.53289   ]])
```

```
scaled_df = pd.DataFrame(scaled, columns=df.columns)
scaled_df.head()
```

	ID	X1	X2	X3	X4	X5	X6	X7	
0	-1.731993	-1.136720	0.810104	0.18606	-1.056895	-1.243791	1.794550	1.782310	-0.697
1	-1.731878	-0.365981	0.810104	0.18606	0.858464	-1.027071	-0.875073	1.782310	0.139
2	-1.731762	-0.597202	0.810104	0.18606	0.858464	-0.160189	0.014802	0.111655	0.139
3	-1.731647	-0.905498	0.810104	0.18606	-1.056895	0.164892	0.014802	0.111655	0.139
4	-1.731531	-0.905498	-1.234065	0.18606	-1.056895	2.332097	-0.875073	0.111655	-0.697

5 rows × 25 columns

```
from sklearn.decomposition import PCA
pcs = PCA()

pcs_t = pcs.fit_transform(scaled_df)

pcs_t[:5]

#generamos el PCA
```

```
array([[-1.81185539e+00, -1.30768482e+00, -4.82160675e-01,
        -4.98785796e-01, -9.47242929e-01, -1.76252911e+00,
        -8.59602312e-01,  2.80762710e+00,  4.97623198e-01,
        -4.07399065e-01, -2.74572897e-01,  3.18141399e-02,
        -6.23595345e-02,  2.27974382e+00,  9.05524549e-01,
        -3.61280892e-01, -9.17814746e-02,  3.90950605e-01,
        -2.64357856e-01, -9.06274940e-01,  6.15013843e-01,
        -7.28620872e-02,  7.81329718e-03, -6.56720848e-04,
         1.48270459e-02],
       [-6.82792725e-01, -2.44421448e+00,  1.14010685e+00,
        -4.25132058e-01, -5.17370293e-01, -1.63315444e+00,
        -1.68236580e-01,  8.98486638e-01,  4.93278902e-01,
        -1.06120187e-02, -2.13794785e-01,  3.06937995e-01,
         1.81098041e-01, -1.40777389e+00,  1.38992120e-01,
        -3.39120647e-01,  6.99904741e-01,  2.10392817e+00,
         2.58866368e-02, -2.76579484e-01,  8.53581852e-01,
        -1.51451166e-01, -4.87764558e-03, -1.32407875e-02,
         1.08135660e-02],
       [-8.69221109e-01, -1.00806528e+00,  5.74075796e-01,
        -5.55650660e-01, -8.36637489e-03, -1.61497649e+00,
         1.03437369e+00, -4.56867088e-02, -8.23738449e-02,
         8.02886148e-02, -2.10719837e-01,  3.20003969e-01,
         1.04670984e-01, -9.64973409e-02, -5.29365807e-01,
         2.83270900e-01,  3.71452885e-03,  2.51930494e-02,
         9.38258222e-03, -2.22067968e-02,  2.72248601e-02,
         8.42547034e-02, -8.59500239e-02, -5.68174472e-02,
        -5.20451663e-02],
       [-2.20597249e-01, -7.44805772e-01, -6.83970159e-01,
         1.06685798e-01, -2.24402965e-03, -1.90584942e+00,
         8.93795210e-01, -2.93791405e-01,  2.56111321e-01,
```

```
       1.25403881e-01,  5.57076424e-02,  1.73259842e-01,
       5.92690945e-02,  3.69859359e-01,  6.54260587e-01,
       4.53677298e-01,  3.63568070e-02, -9.06810244e-03,
       1.82717719e-01,  6.61372774e-03,  1.87474163e-02,
      -2.03496587e-02,  1.33136324e-01,  1.98439822e-03,
      -4.99395186e-02],
     [-8.70527823e-01, -3.56920659e-02, -9.63729693e-01,
       2.12803739e+00, -1.33225541e+00, -4.93278431e-01,
       2.00305351e+00, -4.59893901e-01,  6.78407756e-01,
       3.54926360e-01,  5.41063074e-01, -6.30082780e-02,
       1.12057382e+00, -2.22356991e-01, -7.83410598e-02,
       1.27097716e+00,  2.03081527e-01,  2.84511253e-01,
       8.74895928e-02, -5.97122666e-01,  2.60236935e-01,
      -6.21044156e-02,  7.22608049e-02, -4.38038331e-02,
      -4.68186959e-03]])


pcsSummary_df = pd.DataFrame({
                           '% Varianza Explicada': (pcs.explained_variance_ratio_.round(4))
                           '% Varianza Acumulada': (pcs.explained_variance_ratio_.cumsum())
                           })
pcsSummary_df
```

	% Varianza Explicada	% Varianza Acumulada
0	26.21	26.205319
1	16.81	43.017167
2	6.22	49.236413
3	5.90	55.139881
4	4.24	59.383079
5	3.94	63.325192
6	3.88	67.204733
7	3.66	70.866344

```
pcs_labels = [f'PC{i + 1}' for i in range(len(scaled_df.columns))]
pcsSummary_df.index = pcs_labels
pcsSummary_df
#Hacemos la varianza de los componentes y el acumulado de la varianza
```

	% Varianza Explicada	% Varianza Acumulada
PC1	26.21	26.205319
PC2	16.81	43.017167
PC3	6.22	49.236413
PC4	5.90	55.139881
PC5	4.24	59.383079
PC6	3.94	63.325192
PC7	3.88	67.204733

```
pcs_labels = [f'PC{i + 1}' for i in range(len(scaled_df.columns))]
pcsSummary_df.index = pcs_labels
pcsSummary_df
#Se etiqueta la grafica anterior de acuerdo al componente
```

	% Varianza Explicada	% Varianza Acumulada
PC1	26.21	26.205319
PC2	16.81	43.017167
PC3	6.22	49.236413
PC4	5.90	55.139881
PC5	4.24	59.383079
PC6	3.94	63.325192

```
pcs_df = pd.DataFrame(pcs_t, columns =pcs_labels)
print("Varianza total variables originales: ", scaled_df.var().sum())
print("Varianza total de los componentes: ", pcs_df.var().sum())
```

```
Varianza total variables originales:  25.000833361112036
Varianza total de los componentes:  25.00083336111202
```

```
total_var =scaled_df.var().sum()
pd.DataFrame({
"Porcentaje Varianza": (scaled_df.var()/ total_var) * 100,
"Porcentaje Varianza Acumulado": (scaled_df.var().cumsum() / total_var) * 100
})
```

	Porcentaje Varianza	Porcentaje Varianza Acumulado
ID	4.0	4.0
X1	4.0	8.0
X2	4.0	12.0
X3	4.0	16.0
X4	4.0	20.0
X5	4.0	24.0
X6	4.0	28.0
X7	4.0	32.0
X8	4.0	36.0
X9	4.0	40.0

¿Cuál es el número de componentes mínimo y por qué?

No hay numero minimo de componentes, lo que se busca es minimizar o simplificar la cantidad de datos/informacion. Por lo tanto se toman los datos con as variacion para no descartar informacion importante o bien tomar los valor relevantes.

¿Cuál es la variación de los datos que representan esos componentes?

La variacion es respecto a la media para disminuir la dimencionalidad de los datos multivariados o que tanta informacion hay en esos datos

X18	4.0	76.0

¿Cuál es la pérdida de información después de realizar PCA?

Lo importante es entender que el PCA nno es precisamente la informacion pura de las variable si no una trasnformacion de los datos para poder disminur la cantidad de datos a analizar Por lo tanto una ve que se obtiene la cantidad minima de datos se descartan otro que no precisamente son importantes sin embargo pueden ser no relevantes para el modelo. *

De las variables originales, ¿Cuál tiene mayor y cuál tiene menor importancia en los componentes principales?

Las variables estan dentro de los componentes en el ejemplo los valores de x6 a x17 son los de mayor importancia dada la correlacion que tienen los datos. los datos de menor importancia son los menor variacion donde la curva ya no tiende a moverse.

¿Cuándo se recomienda realizar un PCA y qué beneficios ofrece para Machine Learning?

Es utilizado para identificar patrones en conjuntos de datos con un número dimensiones considerable. Uno de los beneficios que tien un PCA es que permite disminuir multiples datos a solo utilizar para el modelo los mas relevantes con un riesgo bajo de no tomar dato de alto impacto.

```
comps_df = pd.DataFrame(
pcs.components_.round(4), columns = pcs_df.columns, index = scaled_df.columns)
comps_df.iloc[:,:25]
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	.
ID	0.0061	0.0657	-0.0220	0.0199	-0.0056	0.0141	0.1644	0.1972	0.2031	0.2098	
X1	0.0227	0.3115	0.0310	-0.0881	-0.0397	0.0626	-0.2968	-0.3280	-0.3349	-0.3344	
X2	-0.0601	0.0102	0.0263	-0.3231	0.4715	-0.4815	-0.0203	0.0176	0.0622	0.0894	
X3	0.0539	0.0757	-0.0761	0.2248	-0.4209	0.4377	0.0189	0.0484	0.0810	0.1061	
X4	0.4988	-0.1863	0.6097	0.4067	0.0222	-0.1599	-0.1052	-0.0408	-0.0027	0.0373	
X5	0.7123	-0.0560	-0.6478	0.0068	0.1752	0.0071	-0.0145	0.0022	-0.0130	-0.0171	

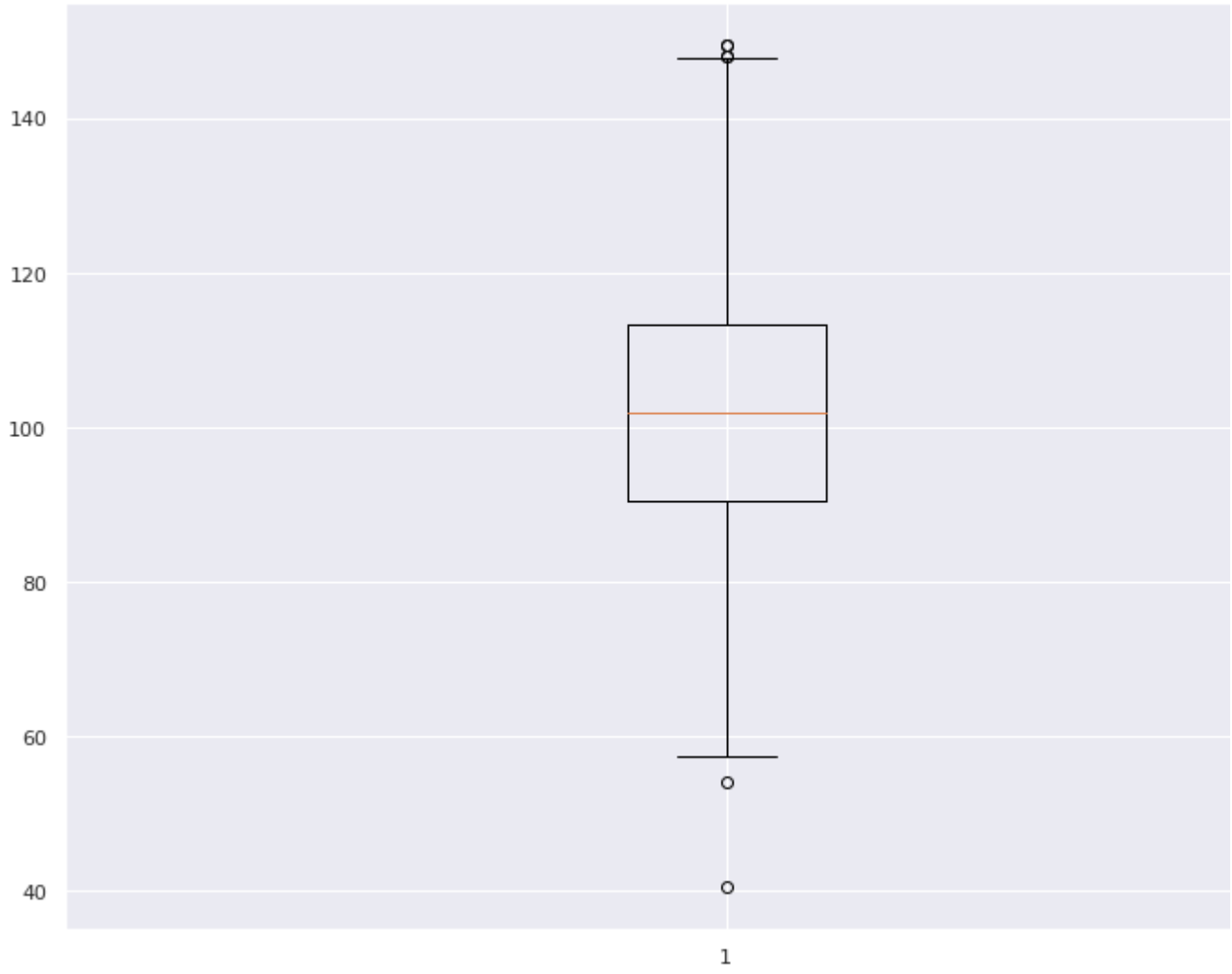
```
comps_df.iloc[:,:25].abs().idxmax()
#Se ponderisa los componentes de acuerdo a la variables originales
```

```
PC1      X5
PC2     X15
PC3      X5
PC4      X7
PC5     X14
PC6     X14
PC7     X16
PC8     X17
PC9     X19
PC10    X17
PC11    X20
PC12    X17
PC13    X18
PC14      Y
PC15      Y
PC16    X21
PC17    X23
PC18    X21
PC19    X12
PC20    X12
PC21    X11
PC22     X9
PC23     X8
PC24    X10
PC25     X7
dtype: object
```

Y	-0.0024	0.0028	0.0009	0.0018	-0.0013	0.0003	0.0005	0.0001	-0.0049	-0.0013

```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(5)

fig = plt.figure(figsize =(10, 7))
```

```
ax = fig.add_axes([0, 0, 1, 1])

bp = ax.boxplot(df)

plt.show()
```



Productos pagados de Colab - Cancela los contratos aquí

✓ 0 s se ejecutó 09:25