Actividad | Visualización

Integrantes

Alumno: **Erick de Jesus Hernández Cerecedo**
Matricula: **A01066428**

Información del Curso

Nombre: **Ciencia y analítica de datos**
Profesor: **Jobish Vallikavungal Devassia**

Fechas: **Martes 1 de noviembre de 2022**

---

1. Descarga de datos

Descarga los datosEnlaces a un sitio externo. y carga el dataset en tu libreta. Descripción aquí.

In [325…

```
# Importar la librerias necesarias para la actividad
import pandas as pd
import numpy as np
import requests
import os

# URL de la base de datos
path = 'default of credit card clients.csv'
# url='https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje-/main/defau

# # Peticion de los datos
# r=requests.get(url)

# # Guardamos los datos en un archivo local con el mismo nombre
# path=os.path.join(os.getcwd(),'default of credit card clients.csv')  # Obtenemos la di
# with open(path,'wb') as f:                                               # Crear a
#     f.write(r.content)                                                   # Escribi

# Lectura de los datos obtenios CSV
data = pd.read_csv(path)
```

In [326…

```
# Visualización del dataframe
data.head()
```

Out[326]:

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X17	X18	X19
0	1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	...	0.0	0.0	0.0	0.0	689.0
1	2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	...	3272.0	3455.0	3261.0	0.0	1000.0
2	3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.0	1518.0	1500.0
3	4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.0	2000.0	2019.0
4	5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	...	20940.0	19146.0	19131.0	2000.0	36681.0

5 rows × 25 columns

2. Información del DataFrame

Obten la información del DataFrame con los métodos y propiedades: shape, columns, head(), dtypes, info(), isna()

In [327…

```
# Información de los datos
print("INFROMACIÓN DE DATAFRAME -----------------")

# shape: forma que tienes el dataframe
print("\nShape: ", data.shape)

# columns: Nombres de las columnas que tiene el dataframe
print("\nColumns: \n", data.columns)

# dtypes: Tipos de datos por cada feature
print("\nDtypes: \n", data.dtypes)

# info: Informacion del dataframe
print("\nInfo: ")
print(data.info())

# isna: Validacion si existen Null or NAN en el dataframe
print("\nIsna: ")
print(data.isna())

print("\n----------------- FIN PARTE 2 -----------------")
```

```
INFROMACIÓN DE DATAFRAME -----------------

Shape:  (30000, 25)

Columns:
 Index(['ID', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9', 'X10',
       'X11', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19', 'X20',
       'X21', 'X22', 'X23', 'Y'],
      dtype='object')

Dtypes:
 ID       int64
X1       int64
X2     float64
X3     float64
X4     float64
X5     float64
X6     float64
X7     float64
X8     float64
X9     float64
X10    float64
X11    float64
X12    float64
X13    float64
X14    float64
X15    float64
X16    float64
X17    float64
X18    float64
X19    float64
X20    float64
X21    float64
X22    float64
X23    float64
Y      float64
dtype: object
```

```
Info:
<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      29999 non-null  float64
 3   X3      29998 non-null  float64
 4   X4      29998 non-null  float64
 5   X5      29995 non-null  float64
 6   X6      29997 non-null  float64
 7   X7      29995 non-null  float64
 8   X8      29993 non-null  float64
 9   X9      29991 non-null  float64
 10  X10     29984 non-null  float64
 11  X11     29986 non-null  float64
 12  X12     29989 non-null  float64
 13  X13     29989 non-null  float64
 14  X14     29987 non-null  float64
 15  X15     29985 non-null  float64
 16  X16     29983 non-null  float64
 17  X17     29990 non-null  float64
 18  X18     29992 non-null  float64
 19  X19     29991 non-null  float64
 20  X20     29992 non-null  float64
 21  X21     29989 non-null  float64
 22  X22     29989 non-null  float64
 23  X23     29995 non-null  float64
 24  Y       29997 non-null  float64
dtypes: float64(23), int64(2)
memory usage: 5.7 MB
None

Isna:
          ID     X1     X2     X3     X4     X5     X6     X7     X8     X9  \
0      False  False  False  False  False  False  False  False  False  False
1      False  False  False  False  False  False  False  False  False  False
2      False  False  False  False  False  False  False  False  False  False
3      False  False  False  False  False  False  False  False  False  False
4      False  False  False  False  False  False  False  False  False  False
...      ...    ...    ...    ...    ...    ...    ...    ...    ...    ...
29995  False  False  False  False  False  False  False  False  False  False
29996  False  False  False  False  False  False  False  False  False  False
29997  False  False  False  False  False  False  False  False  False  False
29998  False  False  False  False  False  False  False  False  False  False
29999  False  False  False  False  False  False  False  False  False  False

       ...    X15    X16    X17    X18    X19    X20    X21    X22    X23  \
0      ...  False  False  False  False  False  False  False  False  False
1      ...  False  False  False  False  False  False  False  False  False
2      ...  False  False  False  False  False  False  False  False  False
3      ...  False  False  False  False  False  False  False  False  False
4      ...  False  False  False  False  False  False  False  False  False
...    ...    ...    ...    ...    ...    ...    ...    ...    ...    ...
29995  ...  False  False  False  False  False  False  False  False  False
29996  ...  False  False  False  False  False  False  False  False  False
29997  ...  False  False  False  False  False  False  False  False  False
29998  ...  False  False  False  False  False  False  False  False  False
29999  ...  False  False  False  False  False  False  False  False  False

           Y
0      False
1      False
```

```
2      False
3      False
4      False
...      ...
29995  False
29996  False
29997  False
29998  False
29999  False

[30000 rows x 25 columns]

----------------- FIN PARTE 2 -----------------
```

3. Limpieza de los datos

Limpia los datos eliminando los registros nulos o rellena con la media de la columna

In [328…

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from pandas import isnull

# Validación si hay valores Null o NaN
print("LIMPIEZA DE DATOS ------------------\n")
print("Validacion de datos NaN y Null")
print("Null values: ", data.isnull().values.any())
print("NaN  values: ", data.isna().values.any())

# Eliminado de la columna ID
data.drop(columns=['ID'], inplace=True)

# Eliminando filas con Na en Y
data = data.dropna(subset=['Y'])

# Lista de variables Numericas
numericas = ['X1','X5','X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19', 'X20', 'X

# Lista de variables Categoricas
categoricas = ['X2', 'X3', 'X4', 'X6', 'X7', 'X8','X9','X10','X11', 'Y']

# Reemplazo de datos NaN por media de la columna usando el Simple Imputer
# pipeline_numericas = Pipeline(steps=[('imputer', SimpleImputer(missing_values=np.nan,
# pipeline_categoricas = Pipeline(steps=[('imputer', SimpleImputer(missing_values=np.nan
pipeline_numericas = SimpleImputer(missing_values=np.nan, strategy='mean')
pipeline_categoricas = SimpleImputer(missing_values=np.nan, strategy='most_frequent')

# transformation = ColumnTransformer(
#     transformers=[
#         ('pipeline_numericas', pipeline_numericas, numericas),
#         ('pipeline_categoricas', pipeline_categoricas, categoricas),
#     ],
#     remainder='passthrough'
# )

# Transformaciones con fit y transform
data[numericas] = pipeline_numericas.fit_transform(data[numericas])
data[categoricas] = pipeline_categoricas.fit_transform(data[categoricas])

# Validación si hay valores Null o NaN
print("\nValidacion de limpieza")
print("Null values: ", data.isnull().values.any())
print("NaN  values: ", data.isna().values.any())
```

```
print("\n----------------- FIN PARTE 3 -----------------")
```

```
LIMPIEZA DE DATOS -----------------

Validacion de datos NaN y Null
Null values:  True
NaN  values:  True

Validacion de limpieza
Null values:  False
NaN  values:  False

----------------- FIN PARTE 3 -----------------
```

In [329…

```
data.head()
```

Out[329]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X15	X16	X17	X18	X1
0	20000.0	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	-2.0	...	0.0	0.0	0.0	0.0	689.
1	120000.0	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	0.0	...	3272.0	3455.0	3261.0	0.0	1000.
2	90000.0	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.0	1518.0	1500.
3	50000.0	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.0	2000.0	2019.
4	50000.0	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	0.0	...	20940.0	19146.0	19131.0	2000.0	36681.

5 rows × 24 columns

4. Estadistica descriptiva, Tendencia Central y Dispersión

Calcula la estadística descriptiva con describe() y explica las medidas de tendencia central y dispersión

In [330…

```
print("ESTADISTICA DESCRIPTIVA, TENDENCIA CENTRAL Y DISPERSIÓN  -----------------")

# Obtenemos la estadistica descriptiva
data.describe()
```

```
ESTADISTICA DESCRIPTIVA, TENDENCIA CENTRAL Y DISPERSIÓN  -----------------
```

Out[330]:

	X1	X2	X3	X4	X5	X6	
count	29997.000000	29997.000000	29997.000000	29997.000000	29997.000000	29997.000000	29997.0
mean	167496.072274	1.603794	1.853085	1.551955	35.483862	-0.016768	-0.
std	129748.803871	0.489116	0.790317	0.521963	9.217346	1.123708	1.
min	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000	-2.0
25%	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000	-1.0
50%	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.0
75%	240000.000000	2.000000	2.000000	2.000000	41.000000	0.000000	0.0
max	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000	8.0

8 rows × 24 columns

data.columns

In [331…

```
print("\n----------------- FIN PARTE 4 -----------------")
```

```
------------------ FIN PARTE 4 ------------------
```

5. Conteo de variables categoricas

Realiza el conteo de las variables categóricas

In [332…

```python
print("CONTEO DE VARIABLES CATEGÓRICAS  -----------------")
# Conteo de las variables categoricas

for column in categoricas:
    print("\n", pd.value_counts(data[column]))
    # print(pd.DataFrame(data[column].value_counts()))

print("\n----------------- FIN PARTE 5 -----------------")
```

```
CONTEO DE VARIABLES CATEGÓRICAS  -----------------

 2.0    18112
1.0    11885
Name: X2, dtype: int64

 2.0    14030
1.0    10584
3.0     4915
5.0      280
4.0      123
6.0       51
0.0       14
Name: X3, dtype: int64

 2.0    15965
1.0    13655
3.0      323
0.0       54
Name: X4, dtype: int64

  0.0    14738
-1.0     5684
 1.0     3688
-2.0     2759
 2.0     2665
 3.0      322
 4.0       76
 5.0       26
 8.0       19
 6.0       11
 7.0        9
Name: X6, dtype: int64

  0.0    15732
-1.0     6047
 2.0     3925
-2.0     3782
 3.0      326
 4.0       99
 1.0       28
 5.0       25
 7.0       20
 6.0       12
 8.0        1
Name: X7, dtype: int64

  0.0    15767
-1.0     5935
```

```
-2.0      4085
 2.0      3817
 3.0       240
 4.0        76
 7.0        27
 6.0        23
 5.0        21
 1.0         4
 8.0         2
Name: X8, dtype: int64

  0.0     16457
-1.0      5685
-2.0      4348
 2.0      3156
 3.0       180
 4.0        69
 7.0        58
 5.0        35
 6.0         5
 1.0         2
 8.0         2
Name: X9, dtype: int64

  0.0     16951
-1.0      5535
-2.0      4546
 2.0      2623
 3.0       178
 4.0        84
 7.0        58
 5.0        17
 6.0         4
 8.0         1
Name: X10, dtype: int64

  0.0     16290
-1.0      5735
-2.0      4895
 2.0      2764
 3.0       184
 4.0        49
 7.0        46
 6.0        19
 5.0        13
 8.0         2
Name: X11, dtype: int64

 0.0    23362
1.0     6635
Name: Y, dtype: int64

----------------- FIN PARTE 5 -----------------
```

In [333…

```
X = data.drop(columns=['Y'])
y = data['Y']
```

6. Escalado de Datos

Escala los datos, si consideras necesario

In [334…

```
print("ESCALADO DE DATOS  -----------------")

# Importacion de la libreria StandarScaler para el escalado de datos
```

```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X[numericas] = scaler.fit_transform(X[numericas])

print("\n----------------- FIN PARTE 6 -----------------")
X.head()
```

```
ESCALADO DE DATOS  -----------------

----------------- FIN PARTE 6 -----------------
```

Out[334]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X14	X15	X16	
0	-1.136801	2.0	2.0	1.0	-1.245918	2.0	2.0	-1.0	-1.0	-2.0	...	-0.668170	-0.672697	-0.663289	-
1	-0.366068	2.0	2.0	2.0	-1.028932	-1.0	2.0	0.0	0.0	0.0	...	-0.639431	-0.621836	-0.606458	-
2	-0.597288	2.0	2.0	2.0	-0.160989	0.0	0.0	0.0	0.0	0.0	...	-0.482585	-0.449929	-0.417412	-
3	-0.905581	2.0	2.0	1.0	0.164490	0.0	0.0	0.0	0.0	0.0	...	0.032671	-0.232571	-0.186949	-
4	-0.905581	1.0	2.0	1.0	2.334348	-1.0	0.0	-1.0	0.0	0.0	...	-0.161365	-0.347196	-0.348360	-

5 rows × 23 columns

7. Reducción de dimensiones

Reduce las dimensiones con PCA, si consideras necesario.

1. Indica la varianza de los datos explicada por cada componente seleccionado. Para actividades de exploración de los datos la varianza > 70%
2. Indica la importancia de las variables en cada componente

In [335…

```python
summary = {
    "Varianzas: ": X.var().round(2),
    "Valor Min: ": X.min().round(2),
    "valor Max: ": X.max().round(2)
}

pd.DataFrame(summary).transpose()
```

Out[335]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X14	X15	X16	X17
Varianzas:	1.00	0.24	0.62	0.27	1.00	1.26	1.43	1.43	1.37	1.28	...	1.00	1.00	1.00	1.00
Valor Min:	-1.21	1.00	0.00	0.00	-1.57	-2.00	-2.00	-2.00	-2.00	-2.00	...	-2.95	-3.32	-2.00	-6.36
valor Max:	6.42	2.00	6.00	3.00	4.72	8.00	8.00	8.00	8.00	8.00	...	23.32	13.19	14.59	15.50

3 rows × 23 columns

In [336…

```python
# Varianza total de todas las variables
varianzaTotal = X.var().sum()

# Varianza de cada variable
varianzas = list()
for i, col in enumerate(X.columns):
    varianzas.append({
        'Nombre': col,
        'Varianza': X[col].var()
    })
```

```python
#Porcentaje que representa la varianza de cada variable respecto al total.
print('Varianza Total: ', varianzaTotal.round(2) )

sumaPorcentajes = 0
for varianza in varianzas:
    sumaPorcentajes += ((varianza['Varianza']/varianzaTotal)*100).round(2)
    print(varianza['Nombre'] + ': \t' ,  ((varianza['Varianza']/varianzaTotal)*100).roun

print('Porcentaje de Varianza de Variables: ', sumaPorcentajes)
```

```
Varianza Total:  23.23
X1:      4.3 %
X2:      1.03 %
X3:      2.69 %
X4:      1.17 %
X5:      4.3 %
X6:      5.43 %
X7:      6.17 %
X8:      6.15 %
X9:      5.88 %
X10:     5.52 %
X11:     5.69 %
X12:     4.3 %
X13:     4.3 %
X14:     4.3 %
X15:     4.3 %
X16:     4.3 %
X17:     4.3 %
X18:     4.3 %
X19:     4.3 %
X20:     4.3 %
X21:     4.3 %
X22:     4.3 %
X23:     4.3 %
Porcentaje de Varianza de Variables:  99.92999999999996
```

In [337…

```python
from sklearn.decomposition import PCA

reduccion = PCA()

X_reducida = reduccion.fit_transform(X)

summary = pd.DataFrame({
    'Varianza Explicada': np.round(reduccion.explained_variance_ratio_,2) * 100, # Obten
    'Varianza Acumulada': np.cumsum(reduccion.explained_variance_ratio_) * 100 # Obtenem
})

filas = [f'PC{i + 1}' for i in range(len(X.columns))]
summary.index = filas

summary.transpose()
```

Out[337]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	
Varianza Explicada	31.000000	21.000000	7.000000	5.000000	4.000000	4.00000	4.00000	4.000000	3.00
Varianza Acumulada	31.440887	52.118888	58.765027	63.522144	67.636616	71.47465	75.26266	79.028949	82.4

2 rows × 23 columns

In [338…

```python
X_componentes = pd.DataFrame(X_reducida, columns = filas)
```

```
print("Varianza total variables originales: ", X.var().sum().round(5))
print("Varianza total de los componentes: ", X_componentes.var().sum().round(5))
```

```
Varianza total variables originales:  23.23486
Varianza total de los componentes:  23.23486
```

In [339…

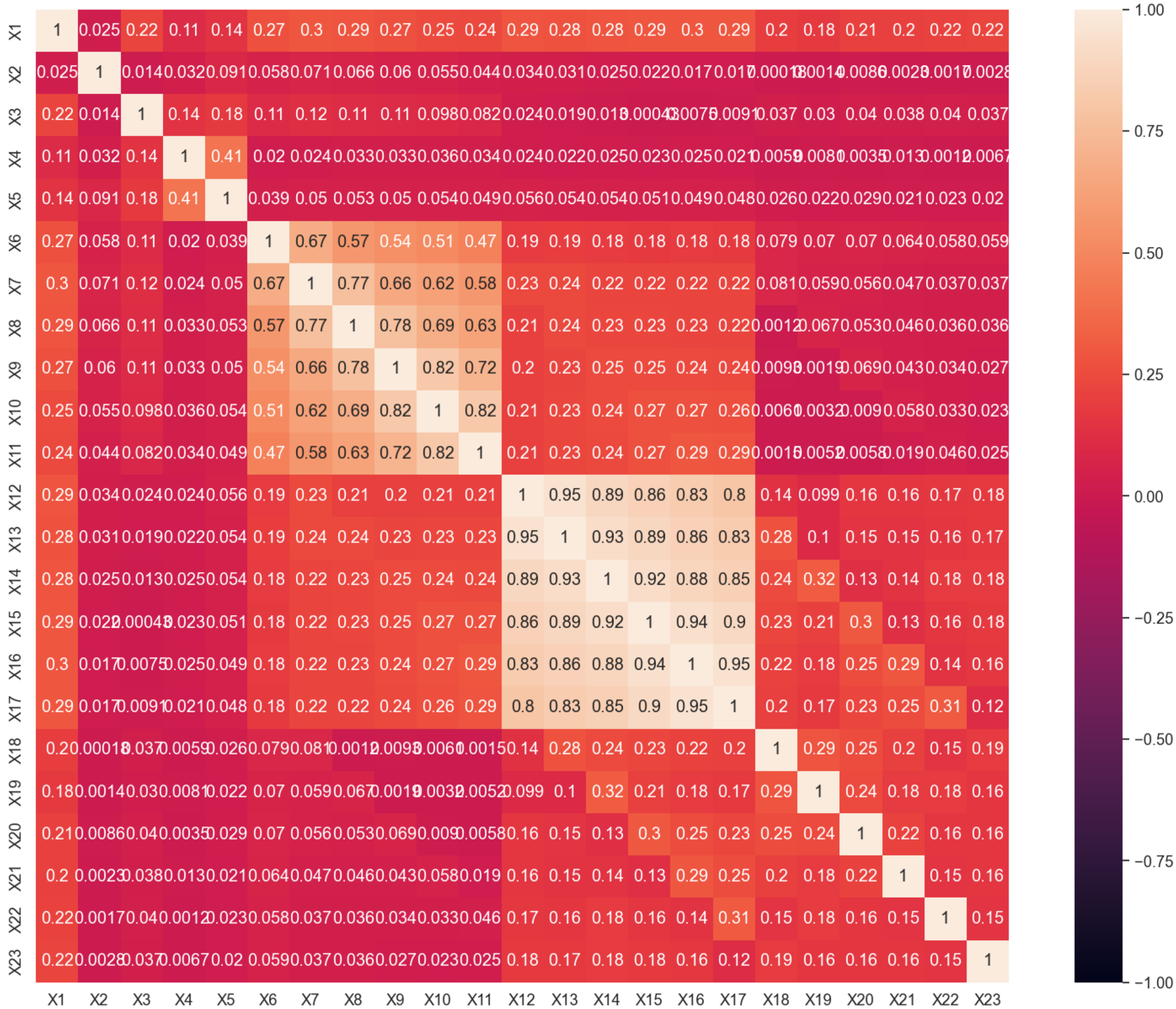
```
summaryaux = {
    'Desviacion Estandar': np.sqrt(reduccion.explained_variance_),
    'Proporcion de Varianza': reduccion.explained_variance_ratio_,
    'Proporcion acumulativa': np.cumsum(reduccion.explained_variance_ratio_)
}
pcsSummary = pd.DataFrame(summaryaux)[0:8].transpose()
pcsSummary = pcsSummary.round(2)
pcsSummary.columns = X_componentes.columns[0:8]
pcsSummary
```

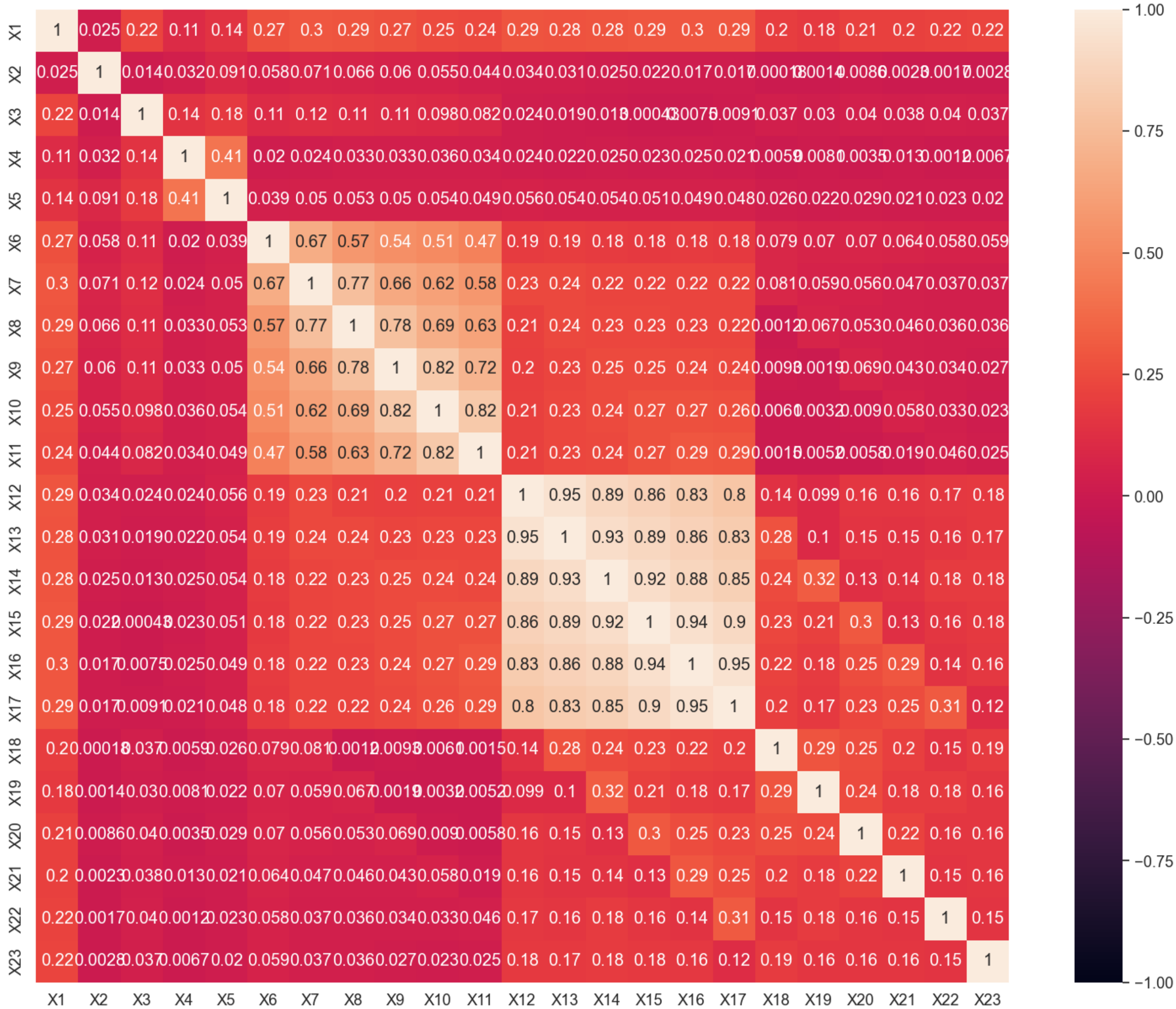
Out[339]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Desviacion Estandar	2.70	2.19	1.24	1.05	0.98	0.94	0.94	0.94
Proporcion de Varianza	0.31	0.21	0.07	0.05	0.04	0.04	0.04	0.04
Proporcion acumulativa	0.31	0.52	0.59	0.64	0.68	0.71	0.75	0.79

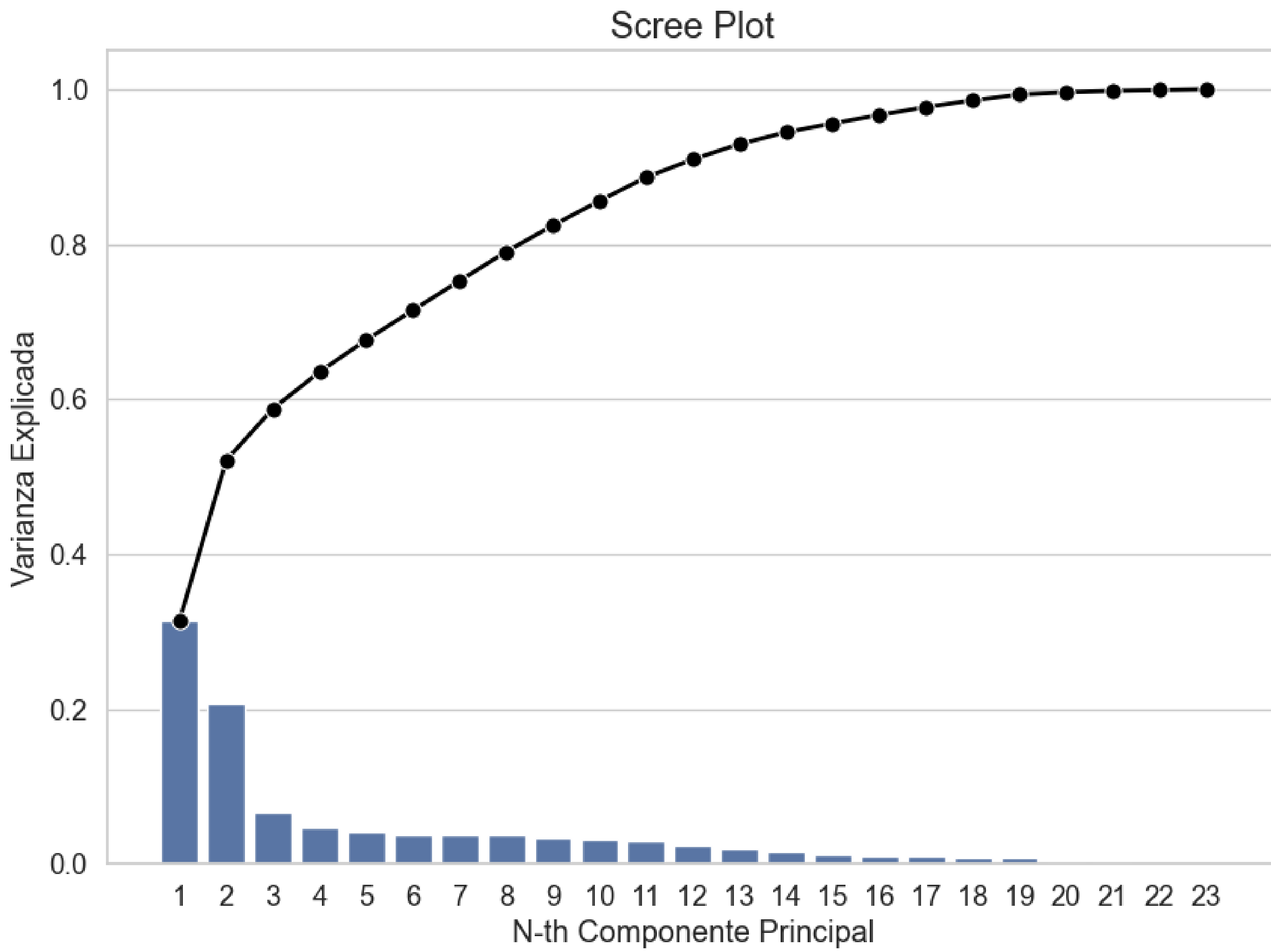
In [340…

```
# Generamos un arreglo con el numero de los componentes
pc_components = np.arange(reduccion.n_components_) + 1

# El Acumulado del radio de la varianza en pcs
cusm = np.cumsum(reduccion.explained_variance_ratio_)

# La variancia por cada componente principal
vartio = reduccion.explained_variance_ratio_

# Immportacion de librerias para graficar
import seaborn as sns
import matplotlib.pyplot as plt

scree = sns.set(style = 'whitegrid', font_scale=1.2)
fig, ax = plt.subplots(figsize = (10, 7))
scree = sns.barplot(x = pc_components, y = vartio, color='b')
scree = sns.lineplot(x = pc_components - 1,
                     y = cusm,
                     color = 'black',
                     linestyle = '-',
                     linewidth = 2,
                     marker = 'o',
                     markersize = 8)
scree.set_title('Scree Plot', fontsize = 17)
scree.set(xlabel='N-th Componente Principal', ylabel='Varianza Explicada')
plt.show()
```


X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X13
X14
X15
X16
X17
X18
X19
X20
X21
X22
X23
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X13
X14
X15
X16
X17
X18
X19
X20
X21
X22
X23
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



In [341…

```
# Matriz de covarianza
covarianza_matiz = np.cov(X.T)

#Calculamos los autovalores y autovectores de la matriz y los mostramos
eigenvalue, eigenvector = np.linalg.eig(covarianza_matiz)

print('Eigenvectors \n%s' %eigenvector)
print('\nEigenvalues \n%s' %eigenvalue)
```

```
Eigenvectors
[[-2.49713060e-03 -2.92983950e-01 -8.65089853e-02  1.03621536e-01
   9.21927698e-02  5.17594054e-02 -3.02519246e-01  6.39830918e-02
  -7.78940770e-02  6.57549473e-01  3.18267979e-03 -7.03727150e-02
   5.76991334e-01 -9.86096899e-03 -1.45303793e-02 -2.77532417e-03
  -5.85885347e-04  7.46261396e-02 -8.80106499e-03  3.82835610e-02
  -6.36767831e-02 -3.60297846e-02  6.40150407e-03]
 [-1.12279889e-02 -7.52134278e-03  2.57216841e-03 -4.48848547e-02
  -9.83066687e-03  7.07673979e-04 -8.56757390e-03 -2.35676839e-03
   6.85746906e-03 -2.04156532e-02 -2.07803411e-03  8.08707831e-03
   9.74349462e-02  4.78408042e-03  1.58385174e-03 -1.87546526e-03
  -1.76902893e-03  1.88709006e-02  2.45313260e-02  4.66566779e-01
   6.97712859e-01  4.66382103e-01  2.54170976e-01]
 [ 2.55534623e-02  5.04732898e-02  5.39377939e-02  2.82273156e-01
  -3.35696887e-02 -4.30713542e-02  1.73479001e-01 -3.00145082e-02
   6.32933956e-02 -5.77132884e-01 -1.43355821e-02 -6.71362625e-02
   7.21776610e-01 -3.70068774e-03 -8.01308653e-04 -2.20029934e-03
   1.83156700e-03  4.36984851e-02 -4.94776300e-03  3.97354906e-02
  -1.11044905e-01 -3.28152162e-02  2.29820995e-02]
 [ 1.41751976e-03  1.67897096e-02 -1.62324958e-02 -2.57742640e-01
   1.12631500e-02 -8.20293656e-03  7.43600841e-03 -2.22682406e-03
  -6.30716639e-03 -2.29395678e-02  4.56928536e-03 -1.38818474e-04
  -2.75704380e-02 -6.24694659e-03  4.26872567e-04  2.45167358e-03
```

```
  1.42005393e-03 -2.15543483e-02  1.93438985e-02  8.46621125e-01
 -3.55165986e-01 -2.74558249e-01 -1.10641106e-01]
[-7.13479307e-04 -5.96790687e-02  4.11187461e-02  9.10573854e-01
 -5.14585125e-02  2.53324287e-02 -3.68980546e-02 -1.06949174e-02
 -9.66179528e-04  7.59945751e-02  6.13126331e-03  3.59817907e-02
 -3.01480987e-01 -2.98321222e-04  1.61189817e-03 -2.32804892e-04
  1.22827185e-05 -1.86530057e-02  1.31480083e-02  2.44292110e-01
 -3.28735446e-02 -3.55390757e-02 -1.70819476e-02]
[ 2.37573725e-01  2.32793587e-01  6.62487817e-02  4.28863566e-02
  3.48412250e-01 -3.55827685e-02  4.31211834e-01  3.79912826e-02
 -8.93964016e-02  2.29128929e-01 -7.89708168e-02  6.29878505e-01
  1.26150239e-01 -5.22215342e-03  1.60992954e-04 -1.99075794e-04
 -2.41846534e-03 -2.70036328e-01  2.74600028e-02  4.20735498e-04
  2.15657573e-02 -8.22678437e-02  1.12979724e-01]
[ 3.01932493e-01  2.77414597e-01  4.41330861e-03  4.14315620e-02
  3.15338050e-01 -2.44159161e-02  2.73960510e-01  3.67697852e-02
 -5.41010395e-02  1.37097917e-01 -1.89732555e-02 -2.86305994e-01
 -3.80972231e-02 -2.28424955e-02 -9.59446655e-03  7.17584525e-05
  2.50577371e-03  5.73917374e-01 -1.07359602e-01  3.74054626e-02
 -9.21945690e-02  3.05650017e-01 -3.21477367e-01]
[ 3.11713194e-01  2.85218419e-01 -7.76270273e-02  3.28866229e-02
  1.16869258e-01 -4.48309168e-02  4.47285484e-02  2.55386433e-02
  3.25762706e-02  9.42117635e-02  1.75224793e-01 -5.30247078e-01
 -2.89038774e-02  1.81025136e-02  2.18912089e-02  3.91808704e-03
 -6.77980910e-03 -1.44063426e-01  2.49859930e-01 -2.00569198e-02
  6.12652681e-02 -3.27617160e-01  5.27140723e-01]
[ 3.11440838e-01  2.75336484e-01 -1.33379861e-01  2.53507747e-02
 -1.04259567e-01 -3.37884130e-02 -2.14207763e-01 -3.61816969e-02
  9.03518763e-02  3.25075773e-02  8.53549471e-03 -1.57773909e-01
  6.03793418e-02  1.74422016e-02 -1.24222086e-03  1.24982922e-03
  1.14466269e-02 -5.63312868e-01 -5.36426336e-01  2.47417307e-02
 -1.44685729e-02  1.90240723e-01 -2.63056007e-01]
[ 3.01759324e-01  2.50784303e-01 -1.57004267e-01  5.77300641e-03
 -2.32122567e-01  3.11423101e-02 -3.04292898e-01 -3.71723816e-02
 -5.80045674e-02 -3.23026430e-02 -3.36195294e-02  1.65643241e-01
  6.12519385e-02  1.97711335e-02 -1.94501599e-02  4.92687343e-04
 -6.86318716e-03 -4.32229437e-02  7.12787664e-01 -1.35068486e-02
 -1.43509266e-02  1.57053987e-01 -3.16235795e-01]
[ 2.94918080e-01  2.29757700e-01 -1.61441440e-01 -5.01057437e-03
 -2.78212114e-01  1.51932906e-01 -3.11420321e-01  1.86856720e-02
  7.01267476e-03 -6.47899742e-02 -6.31784852e-02  3.44384763e-01
 -4.46046645e-04 -3.36016467e-02  1.01849196e-02 -3.90456924e-03
  7.35458209e-05  4.78064361e-01 -3.44606209e-01 -3.66035322e-03
  9.54437045e-02 -2.90915883e-01  2.59205403e-01]
[ 2.67553738e-01 -2.44673175e-01  2.25898324e-01 -2.01696270e-02
  5.56009267e-02 -2.55269244e-02  2.80723475e-03  4.89087123e-02
 -4.18775829e-03 -5.71899732e-02 -8.24980361e-03 -4.19384897e-02
  1.06971888e-03  4.15251986e-01  4.32508893e-01  3.17584371e-01
 -1.83987023e-01  1.04080037e-02 -7.74568271e-03 -4.98120253e-03
  3.30181418e-01 -3.26699268e-01 -3.09353100e-01]
[ 2.78023452e-01 -2.48627532e-01  1.95728489e-01 -2.57933531e-02
 -3.07184465e-03 -8.45887485e-02  2.76543284e-02  4.85226773e-02
  4.48334586e-02 -4.47497733e-02  1.30850805e-01 -1.10719483e-02
 -1.66991323e-02  3.98603073e-02 -3.43582033e-01 -6.46623450e-01
  3.29053207e-01  5.67820048e-03 -8.38043856e-04 -4.56328361e-03
  2.28052715e-01 -2.24878496e-01 -2.03531325e-01]
[ 2.80866479e-01 -2.54414525e-01  1.33457881e-01 -2.71743732e-02
 -5.08150949e-02 -1.53370299e-01  3.69771877e-02 -6.42424954e-02
  1.15073647e-01 -1.77418954e-02 -1.03728491e-01 -2.48065410e-02
 -3.34076385e-02 -4.83724492e-01 -4.97318404e-01  5.26350652e-01
 -8.48336007e-02 -1.13566577e-02  1.37693968e-02 -5.65953086e-03
  7.94294222e-02 -7.92348731e-02 -4.11886552e-02]
[ 2.84920824e-01 -2.54930828e-01  1.19707830e-01 -3.32383022e-02
 -9.37438410e-02 -4.09699416e-02  2.34145323e-02 -9.45603367e-03
 -1.22703266e-01 -4.23160166e-02 -2.74311235e-02 -5.94905947e-03
 -4.07082986e-02 -5.22051665e-01  4.88067474e-01 -3.45426380e-01
```

```
 -3.64178252e-01 -3.85005420e-02  3.52478053e-03 -2.69339048e-03
 -1.21237436e-01  1.36088289e-01  1.02889723e-01]
[ 2.83673885e-01 -2.52238214e-01  1.01150783e-01 -3.67753608e-02
 -7.92212093e-02  1.24408770e-01  2.56163899e-02  5.97002427e-02
  1.19865585e-02 -3.90041217e-02 -4.63017724e-02  2.36695321e-02
 -4.91358340e-02  6.79249654e-02  2.50849140e-01  2.25850132e-01
  7.18474598e-01 -2.09578042e-02  1.68441984e-02  9.21892540e-04
 -2.47726964e-01  2.56563822e-01  2.21118157e-01]
[ 2.77808236e-01 -2.47680469e-01  9.15051597e-02 -3.64802582e-02
  1.21499285e-02  1.50240387e-01 -2.98107669e-02 -1.19383116e-01
 -7.04169233e-04 -6.61461998e-02  4.51311248e-03  4.66539010e-02
 -5.89783842e-02  5.14410426e-01 -3.38730399e-01 -7.11074849e-02
 -4.26115382e-01 -2.89609455e-03 -1.76065561e-02  3.70568929e-03
 -2.90127563e-01  2.85391765e-01  2.68177937e-01]
[ 5.67565892e-02 -1.56876186e-01 -3.98644401e-01  3.81720966e-03
 -2.07157827e-01 -3.16560504e-01  2.34351075e-01  4.39065033e-02
  2.32938102e-01  6.34039156e-02  7.05645334e-01  1.61336435e-01
  7.27031027e-03  4.39300664e-02  6.74598595e-02  8.46266215e-02
 -4.46200777e-02  8.04953993e-02 -1.74472911e-02  5.09233445e-05
 -4.15922560e-02  7.64937788e-02 -3.10999581e-02]
[ 4.47821280e-02 -1.43249888e-01 -4.31069549e-01  8.25782644e-03
 -1.83804704e-01 -3.23833011e-01  2.60843693e-01 -3.30221510e-01
  2.15867037e-01  1.13784719e-01 -5.99453437e-01 -7.82267174e-02
 -1.41076298e-02  1.46596399e-01  7.07094181e-02 -1.24484753e-01
  3.79294999e-02  3.09799582e-02  2.55140289e-02 -8.13073057e-04
  3.18600990e-02 -6.16759227e-02  2.79823615e-02]
[ 4.91398236e-02 -1.56332518e-01 -3.94126586e-01  9.85075903e-03
 -1.47745503e-01  2.18829322e-01  2.47476812e-01  4.86821403e-02
 -7.85413704e-01 -9.18200084e-02  2.29541803e-02 -1.02593901e-01
 -1.29981758e-02  1.36184024e-03 -1.23577312e-01  6.30739417e-02
  2.65522635e-02 -7.82999903e-02 -6.29104844e-02 -3.59802987e-03
  9.03388688e-02 -7.37416512e-02 -6.47987356e-02]
[ 4.34804391e-02 -1.44514375e-01 -3.35276225e-01  1.02558843e-03
  1.18645796e-01  6.72604554e-01  1.67276721e-01  3.16421444e-01
  4.61708370e-01 -5.18498603e-02 -8.29056686e-02 -3.40284038e-02
 -1.92931355e-02 -1.12381360e-01 -2.60204865e-03 -4.19924741e-02
 -8.12635176e-02 -6.81067311e-02  6.06847456e-02  9.36407314e-05
  6.71964649e-02 -4.31370548e-02 -1.01334887e-01]
[ 4.19431385e-02 -1.40328200e-01 -2.63044867e-01  1.24137852e-02
  5.65010467e-01  1.05060603e-01 -2.65042021e-01 -6.34721980e-01
 -1.46627368e-02 -1.97172228e-01  1.72608545e-01  7.70663066e-02
 -4.72071804e-02 -1.00421611e-01  6.95454936e-02  8.13552282e-03
  9.48173930e-02  8.00860384e-03 -3.69389126e-03 -9.61239537e-03
  4.77912883e-02 -4.17927326e-02 -2.40103060e-02]
[ 3.90433848e-02 -1.31172544e-01 -2.94865527e-01  2.68664953e-02
  3.97650647e-01 -4.22175640e-01 -3.02935796e-01  5.87600879e-01
 -6.84753641e-02 -2.47224385e-01 -1.78568827e-01  5.76939091e-02
 -8.60759185e-02  3.47500767e-02 -2.74063906e-02 -8.29234572e-03
 -1.72152553e-02 -9.50206998e-03 -6.07449239e-04 -3.13135887e-03
 -4.53971051e-02  4.87866865e-02  7.00009672e-02]]

Eigenvalues
[7.30524622 4.80450477 1.54422112 1.10530938 0.95599197 0.89176185
 0.8801387  0.87509199 0.78681505 0.7058286  0.73884262 0.53606002
 0.46046176 0.07024428 0.04082093 0.02322078 0.02531726 0.35374531
 0.17444391 0.20088283 0.23594902 0.25170585 0.26825631]
```

In [342…

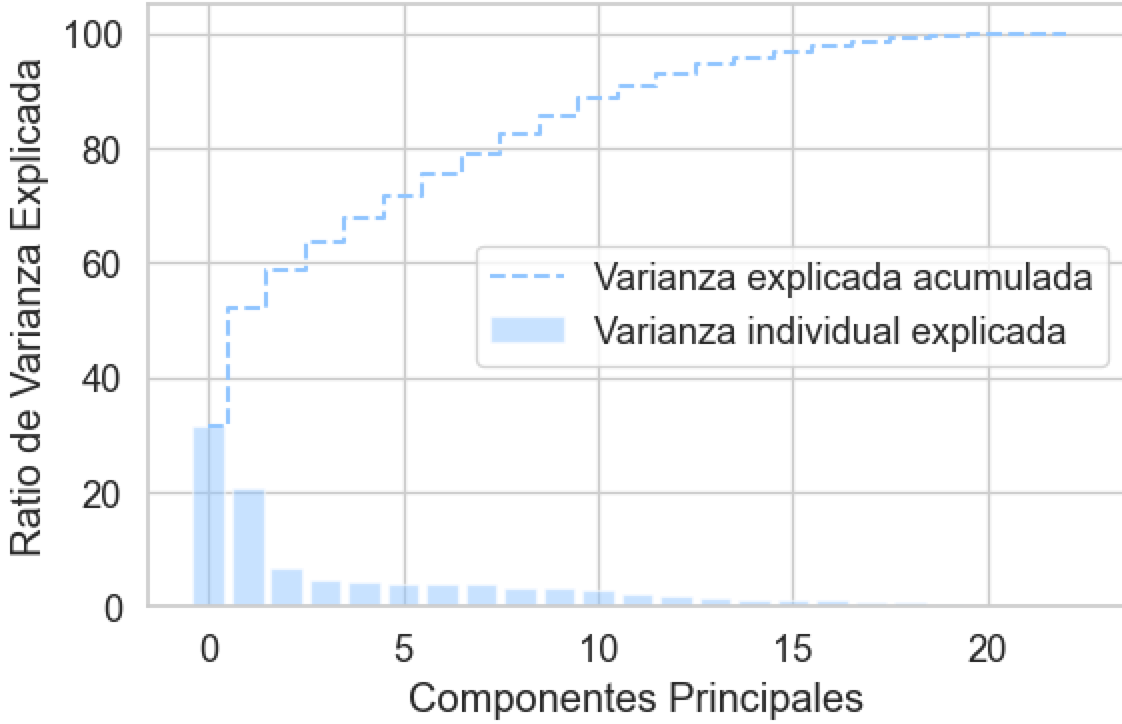
```python
#  Hacemos una lista de parejas (autovector, autovalor)
eigenpairs = [(np.abs(eigenvalue[i]), eigenvector[:,i]) for i in range(len(eigenvalue))]

# Ordenamos estas parejas den orden descendiente con la función sort
eigenpairs.sort(key=lambda x: x[0], reverse=True)

# Visualizamos la lista de autovalores en orden desdenciente
print('Autovalores en orden descendiente:')
```

```
for i in eigenpairs:
    print(i[0])
```

```
Autovalores en orden descendiente:
7.305246219085278
4.8045047709404
1.544221123058335
1.1053093810579915
0.9559919651365707
0.8917618489653881
0.8801386955358247
0.8750919940245098
0.786815050215725 8
0.7388426227842843
0.7058285998168216
0.5360600220037858
0.46046176168475006
0.3537453070580859
0.268256313376149
0.25170584966448745
0.23594901984251596
0.20088282596795748
0.17444391454568212
0.0702442761599057
0.040820931205504785
0.025317256730054766
0.023220777686931923
```

In [343…

```
# A partir de los autovalores, calculamos la varianza explicada
varianza_explicada = [(i / sum(eigenvalue))*100 for i in sorted(eigenvalue, reverse=True
varainza_acumulada = np.cumsum(varianza_explicada)

# Representamos en un diagrama de barras la varianza explicada por cada autovalor, y la
with plt.style.context('seaborn-pastel'):
        plt.figure(figsize=(6, 4))

        plt.bar(range(23), varianza_explicada, alpha=0.5, align='center',
                label='Varianza individual explicada')
        plt.step(range(23), varainza_acumulada, where='mid', linestyle='--', label='Vari
        plt.ylabel('Ratio de Varianza Explicada')
        plt.xlabel('Componentes Principales')
        plt.legend(loc='best')
        plt.tight_layout()
        plt.show()
```



In [344…

```
#Generamos la matríz a partir de los pares autovalor-autovector
matrix_proyeccion = np.hstack((eigenpairs[0][1].reshape(23,1),
                              eigenpairs[1][1].reshape(23,1)))

print('Matriz Proyección:\n', matrix_proyeccion)

Y = X.dot(matrix_proyeccion)
```

```
Matriz Proyección:
 [[-0.00249713 -0.29298395]
 [-0.01122799 -0.00752134]
 [ 0.02555346  0.05047329]
 [ 0.00141752  0.01678971]
 [-0.00071348 -0.05967907]
 [ 0.23757373  0.23279359]
 [ 0.30193249  0.2774146 ]
 [ 0.31171319  0.28521842]
 [ 0.31144084  0.27533648]
 [ 0.30175932  0.2507843 ]
 [ 0.29491808  0.2297577 ]
 [ 0.26755374 -0.24467318]
 [ 0.27802345 -0.24862753]
 [ 0.28086648 -0.25441452]
 [ 0.28492082 -0.25493083]
 [ 0.28367388 -0.25223821]
 [ 0.27780824 -0.24768047]
 [ 0.05675659 -0.15687619]
 [ 0.04478213 -0.14324989]
 [ 0.04913982 -0.15633252]
 [ 0.04348044 -0.14451438]
 [ 0.04194314 -0.1403282 ]
 [ 0.03904338 -0.13117254]]
```

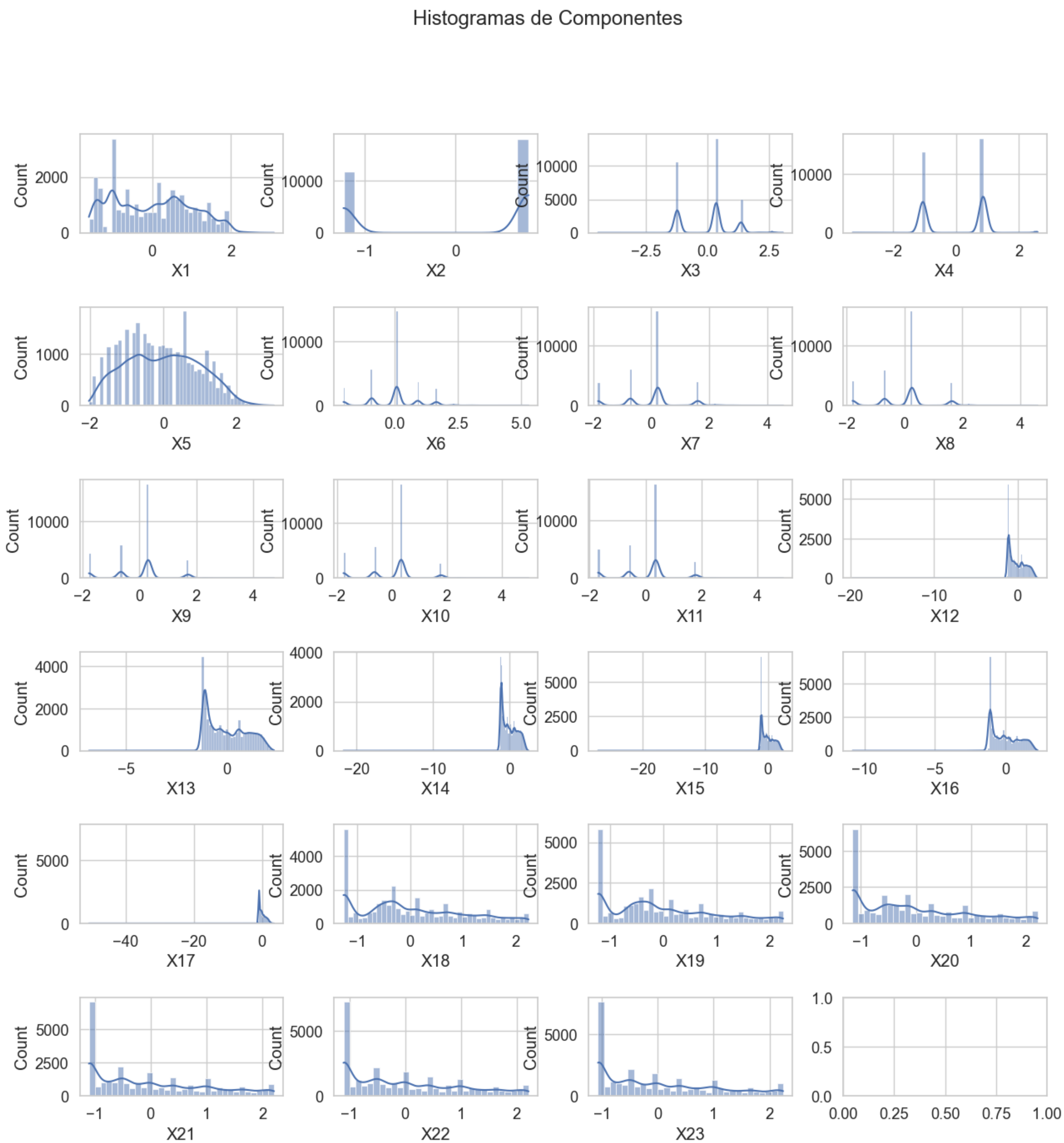
8. Visualizacion de información en Histogramas

Elabora los histogramas de los atributos para visualizar su distribución

In [345…

```
from sklearn.preprocessing import PowerTransformer
```

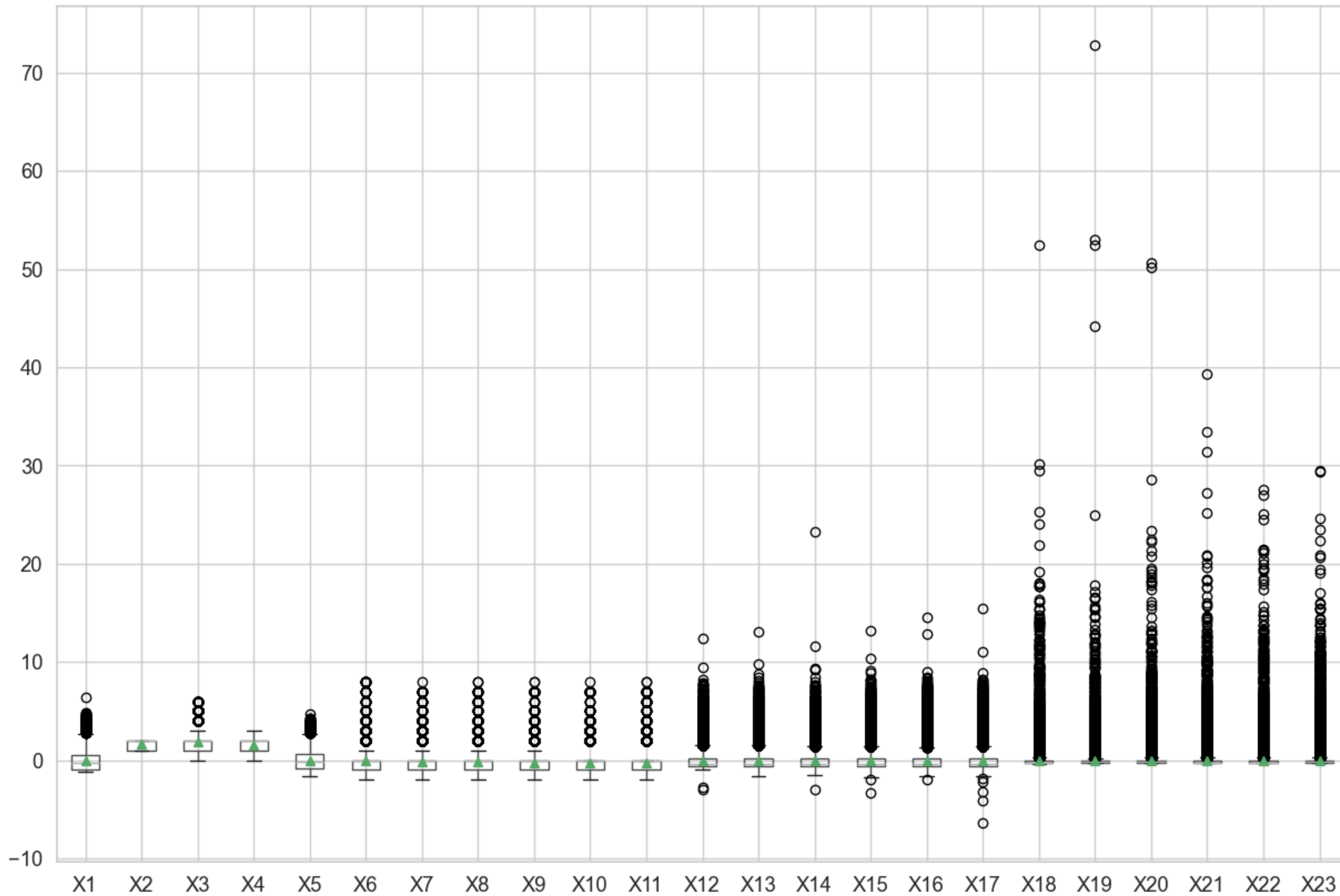
```python
pt = PowerTransformer()
X_escaladas = pd.DataFrame(pt.fit_transform(X), columns=X.columns)

fig, ax  = plt.subplots(6, 4, figsize=(15, 15))
plt.suptitle('Histogramas de Componentes')
plt.subplots_adjust(hspace=0.75, wspace=0.25)
for i, col in enumerate(X_escaladas.columns):
    sns.histplot(data=X_escaladas, x=col, ax=ax[i//4, i%4], kde=True).set(xlabel=col)

plt.show()
```



9. Visualización de informacion en 3 Graficos e interpretación

In [346…

```python
val_atp = X.boxplot(figsize = (15,10),showmeans = True)
val_atp.plot()
plt.show()
```



In [351…
```
corrmat = X.corr().abs()

sns.heatmap(corrmat, annot=True)
```

In [ ]:
```
plt.scatter(X, y, c ="blue")
```