

▼ CIENCIA Y ANALITICA DE DATOS

Actividad Semanal 6 - Visualizacion

Profesor Titular: María de la Paz Rico Fernández

Profesor Tutor: Juan Miguel Meza Méndez

Alumno: Samuel Elias Flores Gonzalez

Matrícula: A01793668

Fecha: 1/Noviembre/2022

```
# Modulos, Librerias y Paquetes
```

```
import numpy as np
import pandas as pd
```

```
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder, StandardScaler
from sklearn.decomposition import PCA
```

▼ 1. Cargar los datos

```
url = "https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje-/main/default%20"
```

```
df = pd.read_csv(url)
```

▼ 2. Informacion del dataframe

```
df.shape #Se verifican dimensiones del dataframe
```

```
(30000, 25)
```

df.columns #Se verifican los nombres actuales de las columnas del dataframe

```
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')
```

df.head(5) #Se despliegan los primeros 5 datos

	ID	Amount_Credit	Gender	Education	Marital_Status	Age	Payment_Sep_2005	Payment
0	1	20000	2.0	2.0	1.0	24.0	2.0	
1	2	120000	2.0	2.0	2.0	26.0	-1.0	
2	3	90000	2.0	2.0	2.0	34.0	0.0	
3	4	50000	2.0	2.0	1.0	37.0	0.0	
4	5	50000	1.0	2.0	1.0	57.0	-1.0	

5 rows × 25 columns



#Se reemplazan los nombres de las columnas

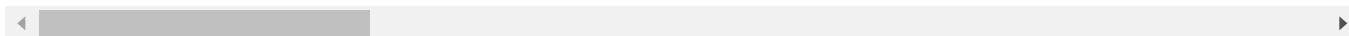
```
df = df.rename(columns = {'X1' : 'Amount_Credit',
                          'X2' : 'Gender',
                          'X3' : 'Education',
                          'X4' : 'Marital_Status',
                          'X5' : 'Age',
                          'X6' : 'Payment_Sep_2005',
                          'X7' : 'Payment_Aug_2005',
                          'X8' : 'Payment_Jul_2005',
                          'X9' : 'Payment_Jun_2005',
                          'X10' : 'Payment_May_2005',
                          'X11' : 'Payment_Apr_2005',
                          'X12' : 'Bill_State_Sep_2005',
                          'X13' : 'Bill_State_Aug_2005',
                          'X14' : 'Bill_State_Jul_2005',
                          'X15' : 'Bill_State_Jun_2005',
                          'X16' : 'Bill_State_May_2005',
                          'X17' : 'Bill_State_Apr_2005',
                          'X18' : 'Previous_Pay_Sep_2005',
                          'X19' : 'Previous_Pay_Aug_2005',
                          'X20' : 'Previous_Pay_Jul_2005',
                          'X21' : 'Previous_Pay_Jun_2005',
                          'X22' : 'Previous_Pay_May_2005',
```

```
'X23' : 'Previous_Pay_Apr_2005' },
inplace = False
)
```

df.head(5) #Se vuelven a desplegar los primeros 5 valores del dataframe

	ID	Amount_Credit	Gender	Education	Marital_Status	Age	Payment_Sep_2005	Payment
0	1	20000	2.0	2.0	1.0	24.0	2.0	
1	2	120000	2.0	2.0	2.0	26.0	-1.0	
2	3	90000	2.0	2.0	2.0	34.0	0.0	
3	4	50000	2.0	2.0	1.0	37.0	0.0	
4	5	50000	1.0	2.0	1.0	57.0	-1.0	

5 rows × 25 columns



df.dtypes

```
ID                int64
Amount_Credit     int64
Gender            float64
Education         float64
Marital_Status    float64
Age              float64
Payment_Sep_2005  float64
Payment_Aug_2005  float64
Payment_Jul_2005  float64
Payment_Jun_2005  float64
Payment_May_2005  float64
Payment_Apr_2005  float64
Bill_State_Sep_2005 float64
Bill_State_Aug_2005 float64
Bill_State_Jul_2005 float64
Bill_State_Jun_2005 float64
Bill_State_May_2005 float64
Bill_State_Apr_2005 float64
Previous_Pay_Sep_2005 float64
Previous_Pay_Aug_2005 float64
Previous_Pay_Jul_2005 float64
Previous_Pay_Jun_2005 float64
Previous_Pay_May_2005 float64
Previous_Pay_Apr_2005 float64
Y                float64
dtype: object
```

```
df.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   Amount_Credit                        30000 non-null  int64
2   Gender                               29999 non-null  float64
3   Education                            29998 non-null  float64
4   Marital_Status                       29998 non-null  float64
5   Age                                  29995 non-null  float64
6   Payment_Sep_2005                     29997 non-null  float64
7   Payment_Aug_2005                     29995 non-null  float64
8   Payment_Jul_2005                     29993 non-null  float64
9   Payment_Jun_2005                     29991 non-null  float64
10  Payment_May_2005                     29984 non-null  float64
11  Payment_Apr_2005                     29986 non-null  float64
12  Bill_State_Sep_2005                  29989 non-null  float64
13  Bill_State_Aug_2005                  29989 non-null  float64
14  Bill_State_Jul_2005                  29987 non-null  float64
15  Bill_State_Jun_2005                  29985 non-null  float64
16  Bill_State_May_2005                  29983 non-null  float64
17  Bill_State_Apr_2005                  29990 non-null  float64
18  Previous_Pay_Sep_2005                29992 non-null  float64
19  Previous_Pay_Aug_2005                29991 non-null  float64
20  Previous_Pay_Jul_2005                29992 non-null  float64
21  Previous_Pay_Jun_2005                29989 non-null  float64
22  Previous_Pay_May_2005                29989 non-null  float64
23  Previous_Pay_Apr_2005                29995 non-null  float64
24  Y                                     29997 non-null  float64
dtypes: float64(23), int64(2)
memory usage: 5.7 MB
```

```
df.isna()
```

	ID	Amount_Credit	Gender	Education	Marital_Status	Age	Payment_Sep_2005
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False

```
df.isna().values.any() # Se verifica si hay algun dato vacio
```

```
True
```

29995	False	False	False	False	False	False	False
-------	-------	-------	-------	-------	-------	-------	-------

▼ 3. Limpieza de los datos

29995	False	False	False	False	False	False	False
-------	-------	-------	-------	-------	-------	-------	-------

```
df.dropna(inplace = True) #Eliminamos los datos NaN o nulos
```

```
df.isna().values.any() #Comprobamos si existen datos nulos
```

```
False
```

▼ 4. Calculo de la estadística descriptiva

```
df.describe()
```

Decidimos eliminar los registros vacios debido a que la cantidad era muy pequeña en comparacion con la cantidad total de registros, es decir se eliminaron 42 registros de 30000, menos del 1%.

Segun la estadistica descriptiva, las columnas o variables que presentan mayor desviacion estandar son: Amount_Credit, los Bill_state y los Previous_Pay; mientras que todas las demas columnas presentas una desviacion estandar alrededor de 0 y 1.

Podemos observar que las columnas presentan distintas magnitudes, es decir mientras que Amount_Credit esta en el orden de los miles, otras variables o columnas estan entre los valores 0 y 1. Esto quiere decir que se debe aplicar una normalizacion a los datos para poder tener una menor diferencia en los rangos de los mismos.

▼ 5. Conteo de variables categoricas

Se separan las variables de entrada y la salida.

```
X = df.drop("Y", axis=1) #Eliminamos la columna y y almacenamos dataframe en X
Y = df["Y"] #Almacenamos columna Y
```

Ahora nos interesa identificar a las variables categoricas para proceder a eliminarlas, puesto que solo nos interesan las numericas

ID, Gender, Education, Marital status, Age, Payment September - April 2005.

```
X = X.drop(['ID', 'Gender', 'Education', 'Marital_Status', 'Age', 'Payment_Sep_2005', 'Paymen
X.head() #Mostramos dataframe con datos numericos
```

	Amount_Credit	Bill_State_Sep_2005	Bill_State_Aug_2005	Bill_State_Jul_2005	Bill_S
0	20000	3913.0	3102.0	689.0	
1	120000	2682.0	1725.0	2682.0	
2	90000	29239.0	14027.0	13559.0	
3	50000	46990.0	48233.0	49291.0	
4	50000	8617.0	5670.0	35835.0	



▼ 6. Escalamiento de los datos

Prodemos a escalar los datos para evitar sesgos creados por la diferencias de magnitudes de cada una de las columnas.

```
scaler = MinMaxScaler() #Definimos escalamiento
scaled = scaler.fit_transform(X)
scaled_X = pd.DataFrame(scaled, columns=X.columns) #Realizamos escalamiento
scaled_X.head()
```

	Amount_Credit	Bill_State_Sep_2005	Bill_State_Aug_2005	Bill_State_Jul_2005	Bill_S
0	0.010101	0.149982	0.069164	0.086723	
1	0.111111	0.148892	0.067858	0.087817	
2	0.080808	0.172392	0.079532	0.093789	
3	0.040404	0.188100	0.111995	0.113407	
4	0.040404	0.154144	0.071601	0.106020	



Como se puede observar, ahora mismo, los datos todas las columnas se encuentran en un rango de 0 a 1.

▼ 7. Reduccion de dimensiones con PCA

```
pcs = PCA() #Aplicamos PCA
pcs_t = pcs.fit_transform(scaled_X)
pcs_t

array([[ -1.70220739e-01,  4.55874383e-02, -1.20064468e-04, ...,
         3.58960605e-04,  3.36984589e-04,  6.77740494e-04],
       [-9.90982643e-02, -2.60772497e-02, -4.67276907e-03, ...,
         7.06196680e-04, -2.85343235e-04, -3.61080529e-05],
       [-9.70277616e-02,  1.68328789e-02, -1.47786552e-03, ...,
        -5.17236765e-03, -1.22349588e-03, -4.13295690e-04],
       ...,
       [-1.45295136e-01,  5.31650933e-02,  1.57707551e-02, ...,
         4.14252460e-03,  8.99087513e-04, -1.48836952e-03],
       [-5.78218997e-02,  5.70665513e-02,  1.07110150e-01, ...,
```

```
-1.28164910e-02, -7.85206363e-03, -9.97520841e-03],
[-9.19042877e-02, 7.78591548e-02, -8.01930001e-03, ...,
-4.45835633e-05, 7.39169942e-03, -2.01033628e-03]])
```

▼ 7.1. Explicación de la varianza de los datos

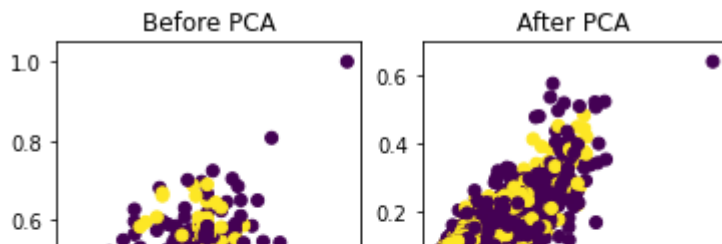
```
pcsSummary_df = pd.DataFrame({'% Varianza Explicada': np.round(pcs.explained_variance_,6) * 100,
                             "Desviación Estándar": np.round(np.sqrt(pcs.explained_variance_),6),
                             "%Proporción de Varianza": np.round(pcs.explained_variance_ratio_,6),
                             '% Varianza Acumulada': np.round(np.cumsum(pcs.explained_variance_ratio_),6)}
                             )
pcs_labels = [f'PC{i + 1}' for i in range(len(scaled_X.columns))]
pcsSummary_df.index = pcs_labels
pcsSummary_df = pcsSummary_df.round(2).transpose()
pcsSummary_df
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	P
% Varianza Explicada	2.31	1.22	0.14	0.11	0.10	0.06	0.04	0.04	0.02	0.01	0.01	C
Desviación Estándar	0.15	0.11	0.04	0.03	0.03	0.02	0.02	0.02	0.02	0.01	0.01	C
%Proporción de Varianza	56.50	29.90	3.30	2.80	2.50	1.50	1.10	0.90	0.60	0.40	0.30	C

Como vemos en la tabla, las primeras dos componentes principales nos explican el 86.5%, por lo cual podemos decir que este es el número mínimo de componentes principales. A partir de la tercera componente la varianza explicada es muy pequeña como para ser considerada.

Double-click (or enter) to edit

```
fig, axes = plt.subplots(1,2)
axes[0].scatter(scaled_X["Amount_Credit"], scaled_X["Bill_State_Sep_2005"], c=Y)
axes[0].set_title('Before PCA')
axes[1].scatter(pcs_t[:,0], pcs_t[:,1], c=Y)
axes[1].set_title('After PCA')
plt.show()
```

Se puede observar como despues de aplicar el PCA, utilizando las dos primeras componentes, se reduce su varianza distribuyendose de mejor manera, si se visualizan demasiado juntas es debido a la cantidad de registros en el conjunto de datos.



7.2. Importancia de las variables de cada componente

```
pcsComponents_df = pd.DataFrame(pcs.components_.transpose(),
                                columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9'],
                                index=X.columns)

pcsComponents_df.round(3)
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	
Amount_Credit	0.681	-0.727	-0.077	-0.022	-0.025	-0.015	-0.015	-0.011	-0.004	-
Bill_State_Sep_2005	0.334	0.317	-0.194	0.424	-0.229	0.231	-0.204	0.366	0.088	
Bill_State_Aug_2005	0.350	0.343	-0.177	0.329	-0.161	0.106	0.276	-0.225	-0.271	-
Bill_State_Jul_2005	0.196	0.188	-0.021	0.041	-0.012	-0.106	0.105	-0.247	0.534	
Bill_State_Jun_2005	0.317	0.298	0.028	-0.227	0.155	-0.395	0.138	0.194	0.187	-
Bill_State_May_2005	0.313	0.289	0.095	-0.429	0.260	0.074	-0.197	-0.099	-0.128	
Bill_State_Apr_2005	0.232	0.212	0.240	-0.332	-0.026	-0.020	-0.232	-0.136	-0.259	
Previous_Pay_Sep_2005	0.034	0.004	0.097	0.028	0.092	0.054	0.712	-0.322	-0.245	
Previous_Pay_Aug_2005	0.020	-0.002	0.094	-0.031	0.064	-0.030	0.223	-0.031	0.626	
Previous_Pay_Jul_2005	0.035	-0.001	0.140	-0.095	0.141	-0.017	0.406	0.762	-0.148	
Previous_Pay_Jun_2005	0.043	-0.004	0.215	-0.124	0.196	0.855	0.048	0.019	0.193	-
Previous_Pay_May_2005	0.062	-0.013	0.816	0.150	-0.501	-0.084	-0.018	0.004	0.018	-
Previous_Pay_Apr_2005	0.057	-0.014	0.329	0.562	0.712	-0.141	-0.186	-0.057	-0.063	

```
pcsComponents_df.PC1.abs().idxmax()
```

```
'Amount_Credit'
```

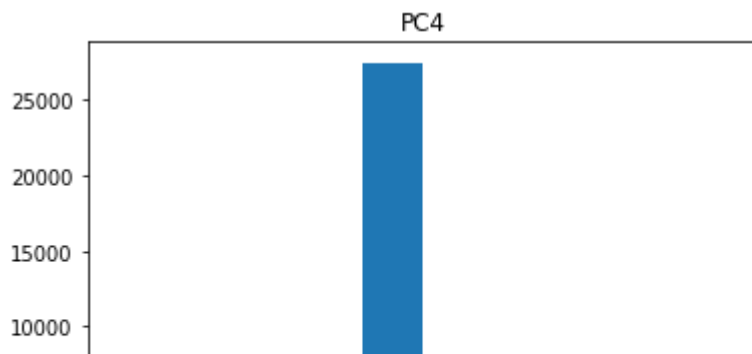
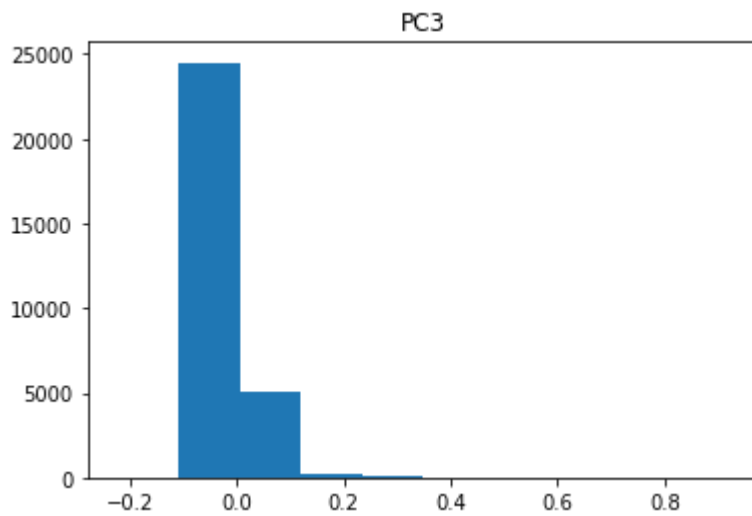
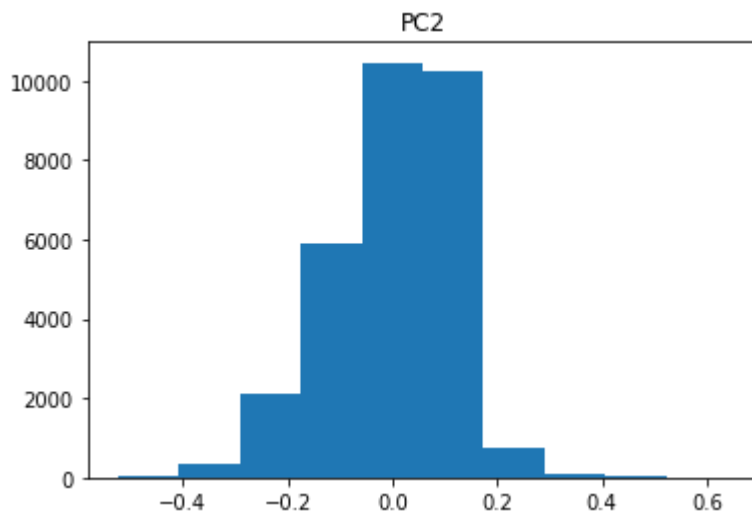
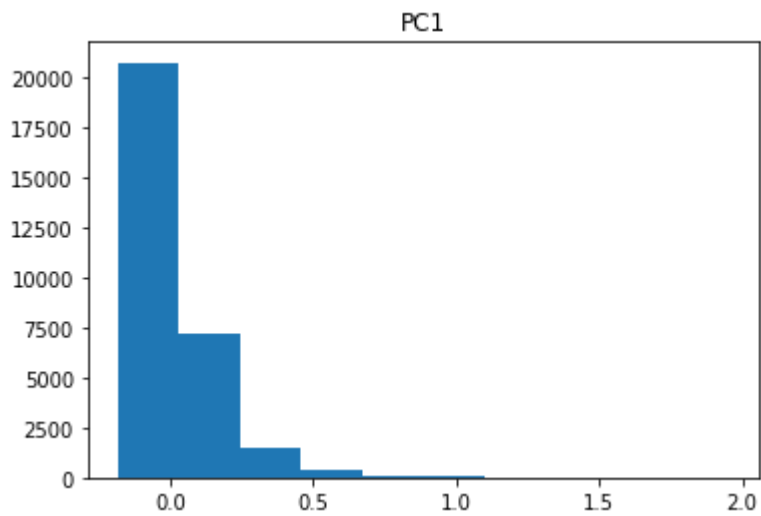
```
pcsComponents_df.PC2.abs().idxmax()
```

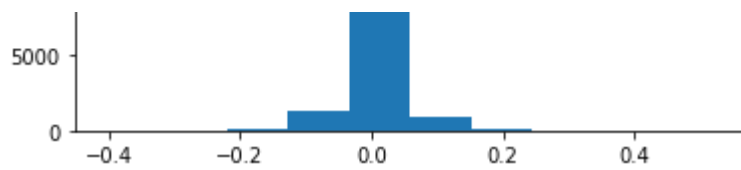
```
'Amount_Credit'
```

Observamos que tanto la componente 1 y 2 tienen un mayor peso sobre la variable Amount_Credit

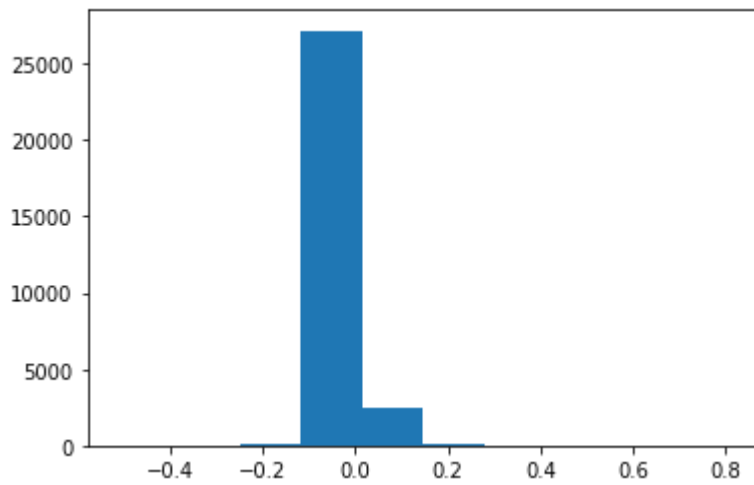
▼ 8. Histograma de los atributos

```
fig, ax = plt.subplots()
for i in range(0,13):
    plt.hist(pcs_t[:,i])
    plt.title("PC%d"%(i+1))
    plt.show()
```

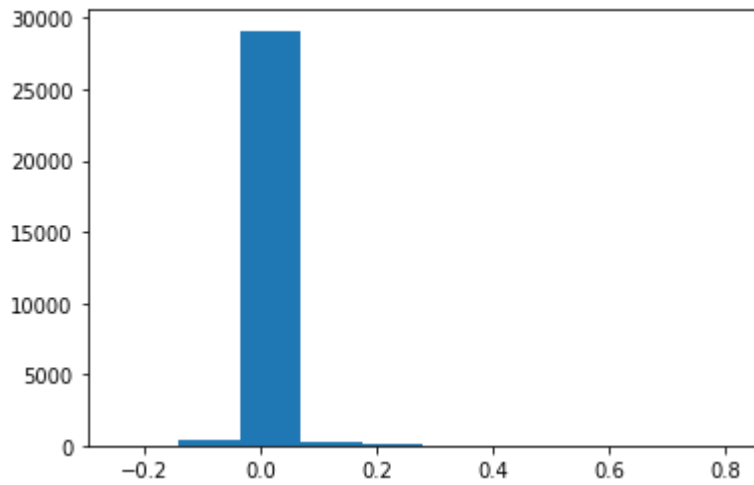




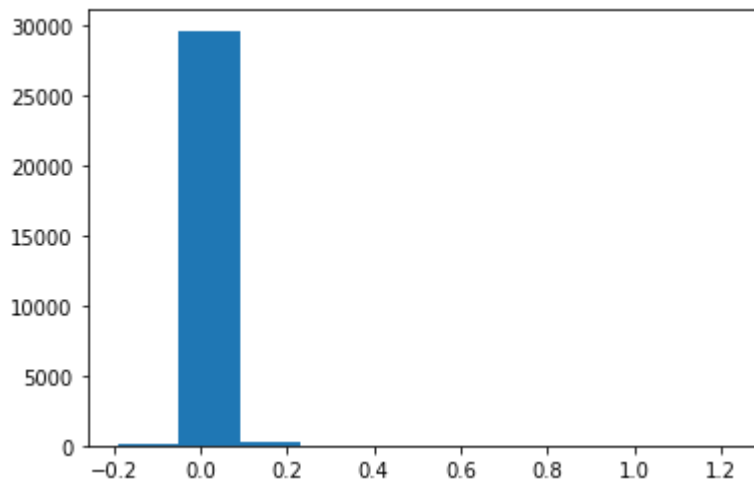
PC5



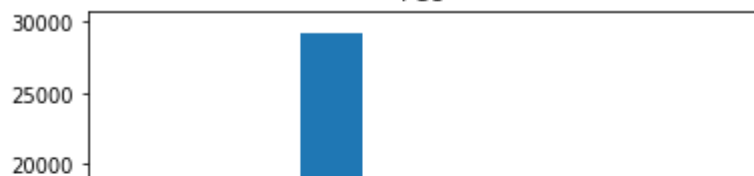
PC6

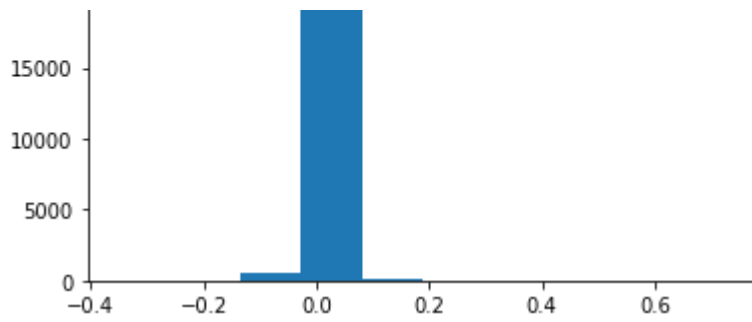


PC7

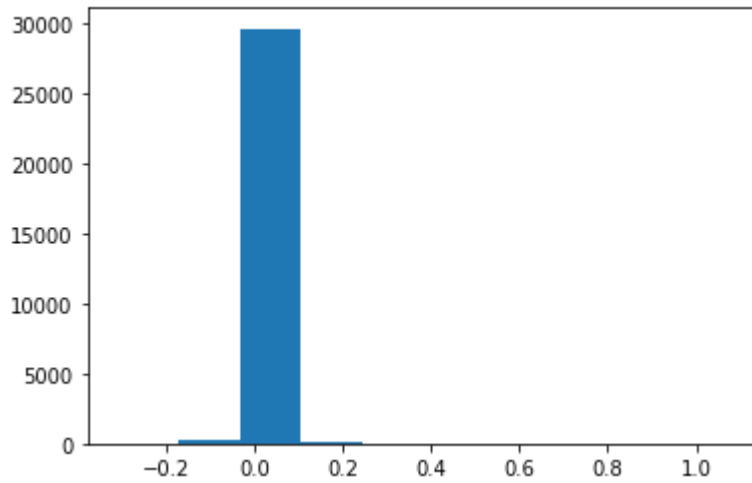


PC8

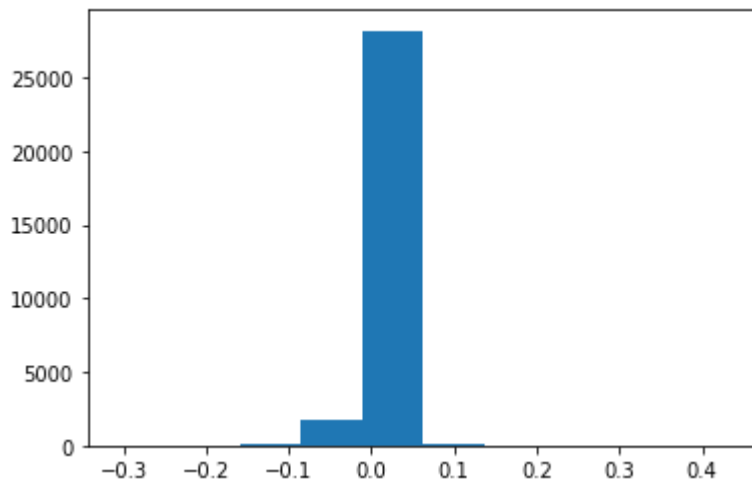




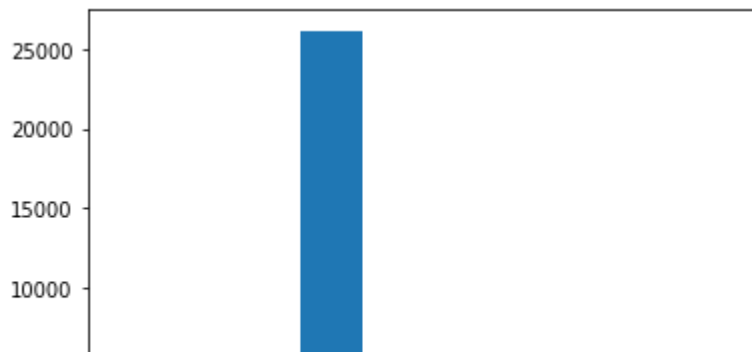
PC9



PC10



PC11

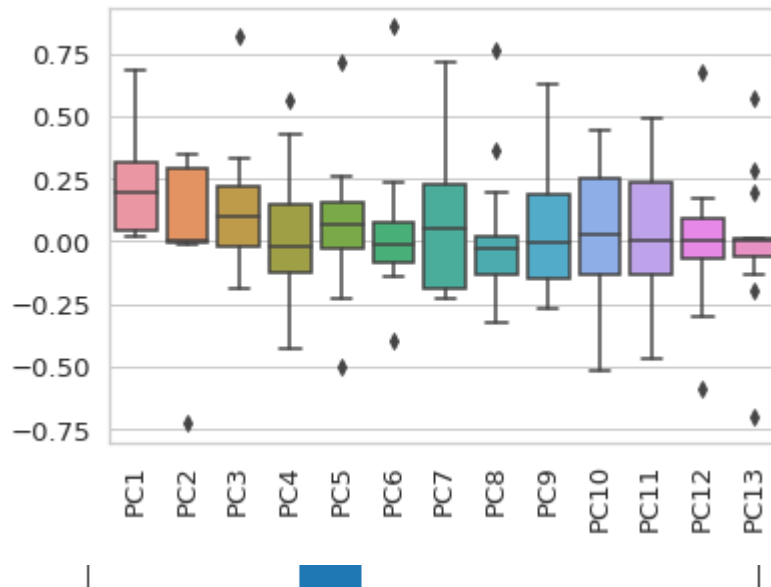


▼ 9. Visualizacion de los datos usando 3 graficos

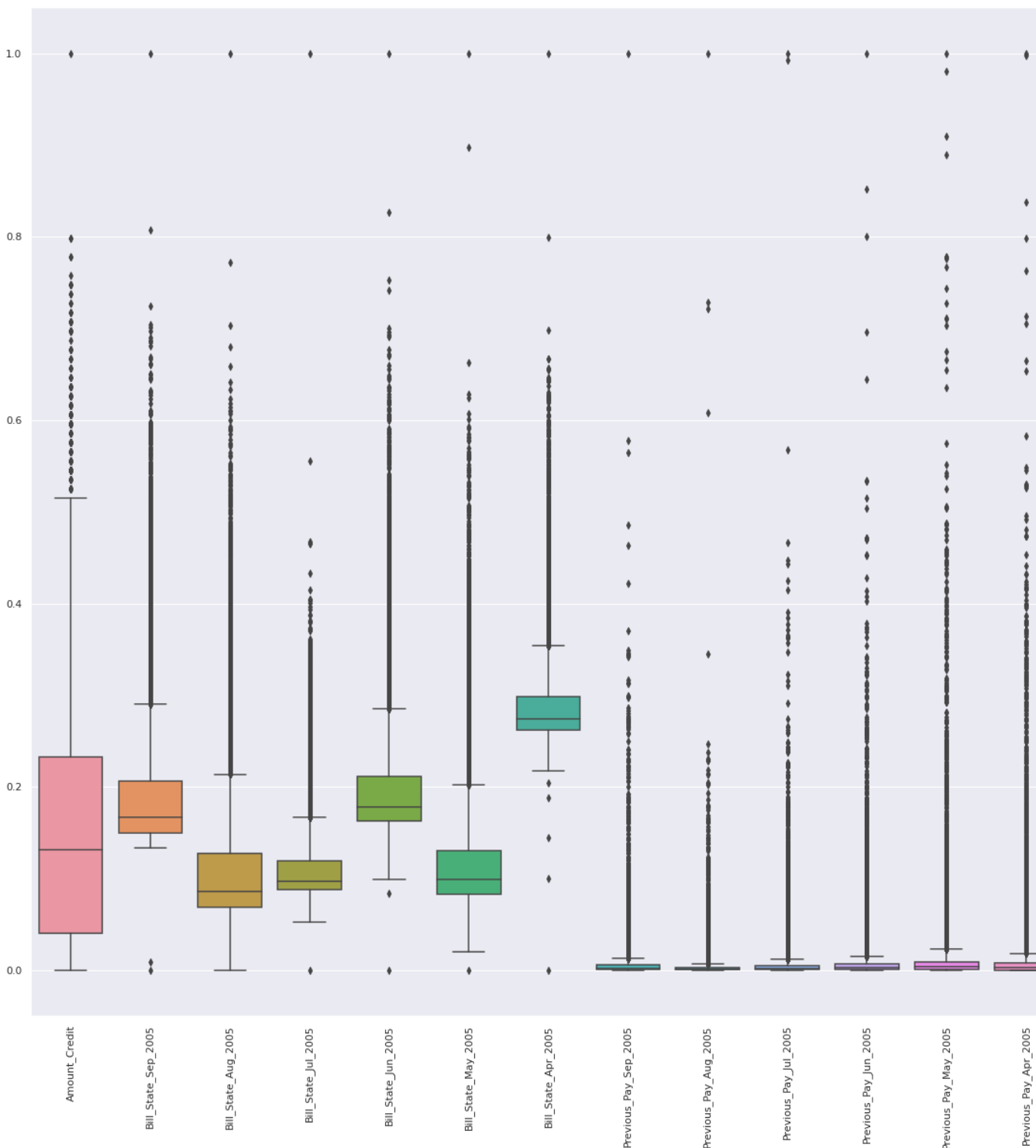
PC12

Grafica 1

```
plt.xticks(rotation = 'vertical')
g = sns.boxplot(data=pcsComponents_df)
```



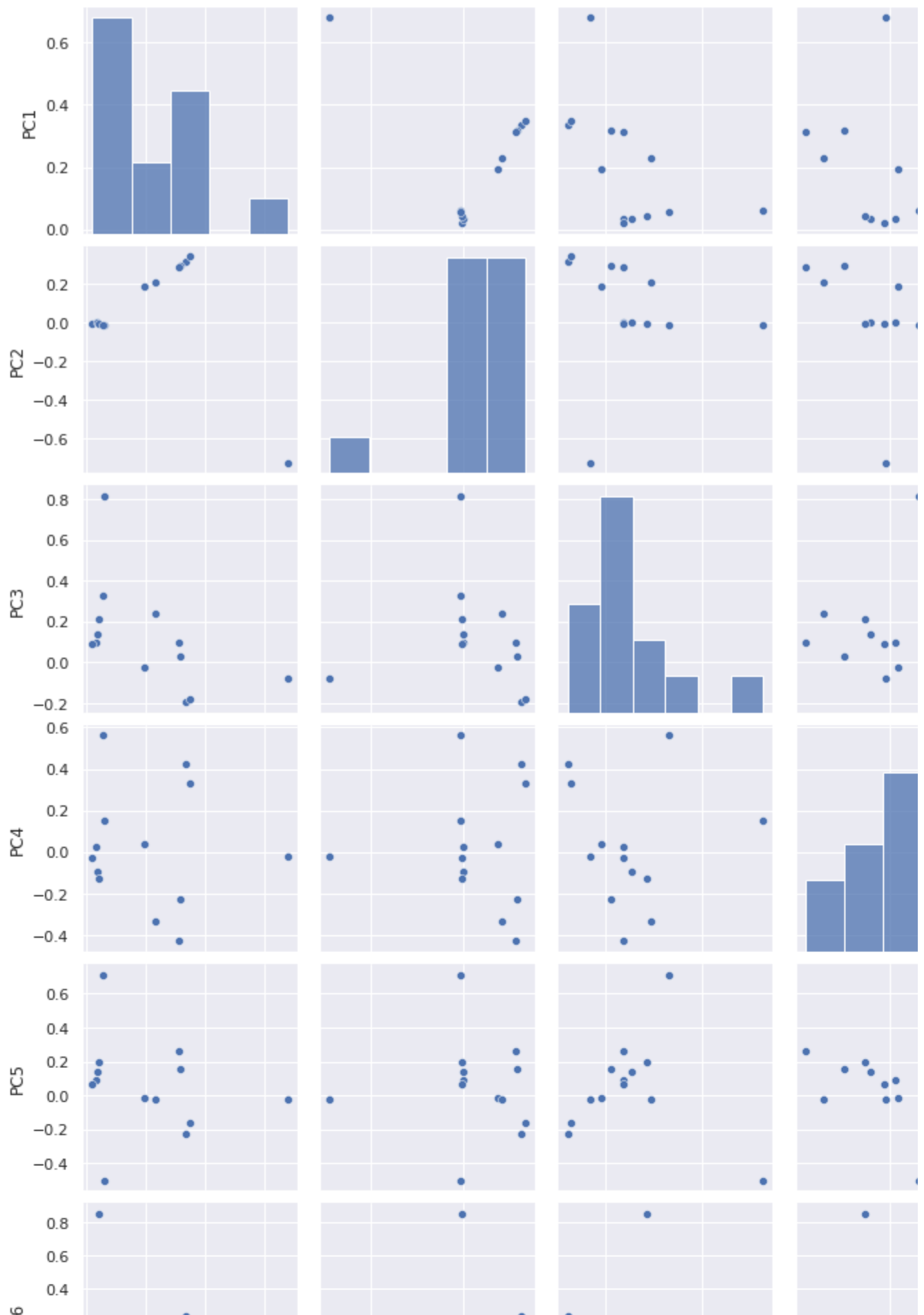
```
plt.xticks(rotation = 'vertical')
g = sns.boxplot(data=scaled_X)
```

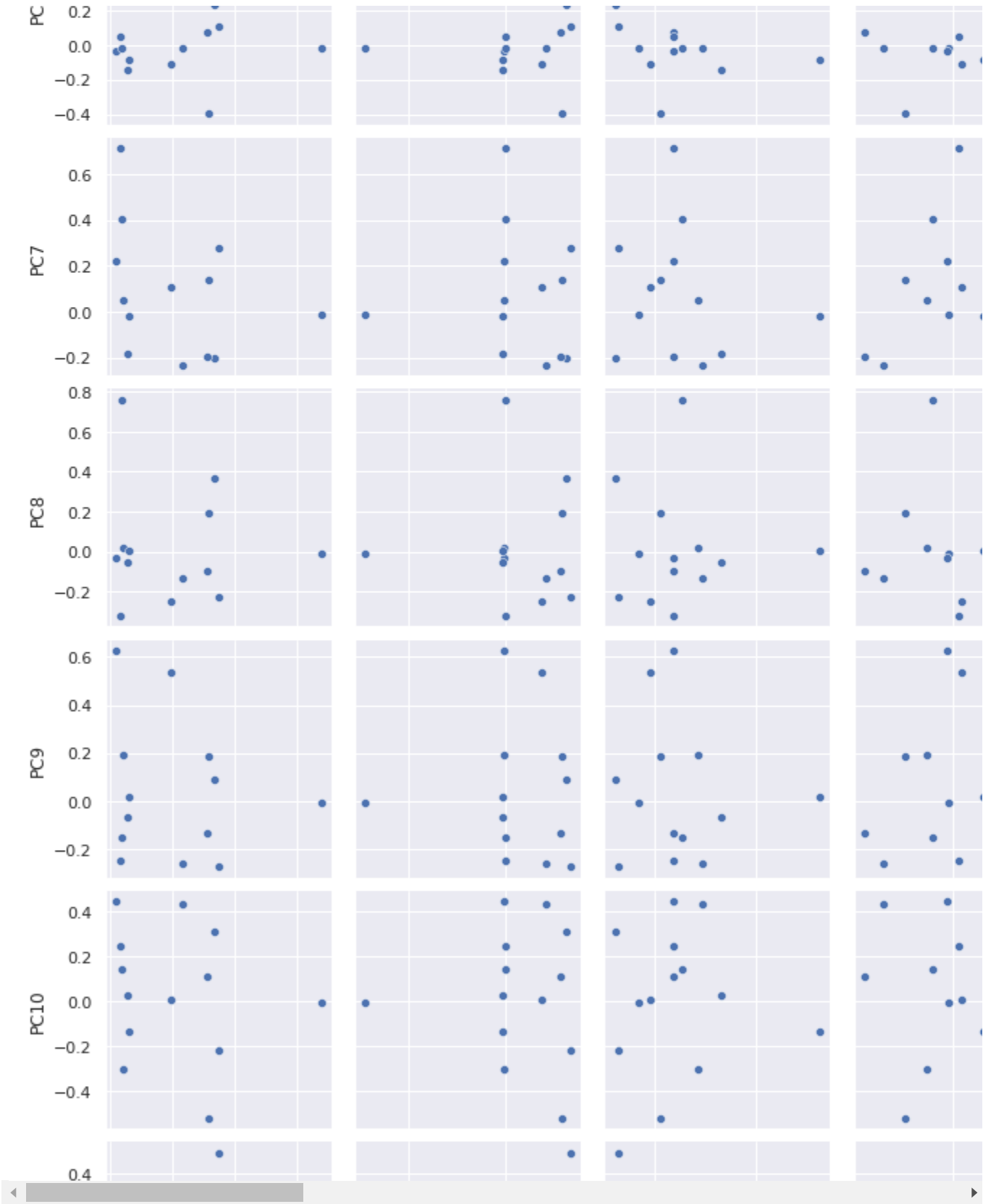


Grafica 2

```
sns.pairplot(pcsComponents_df)
```

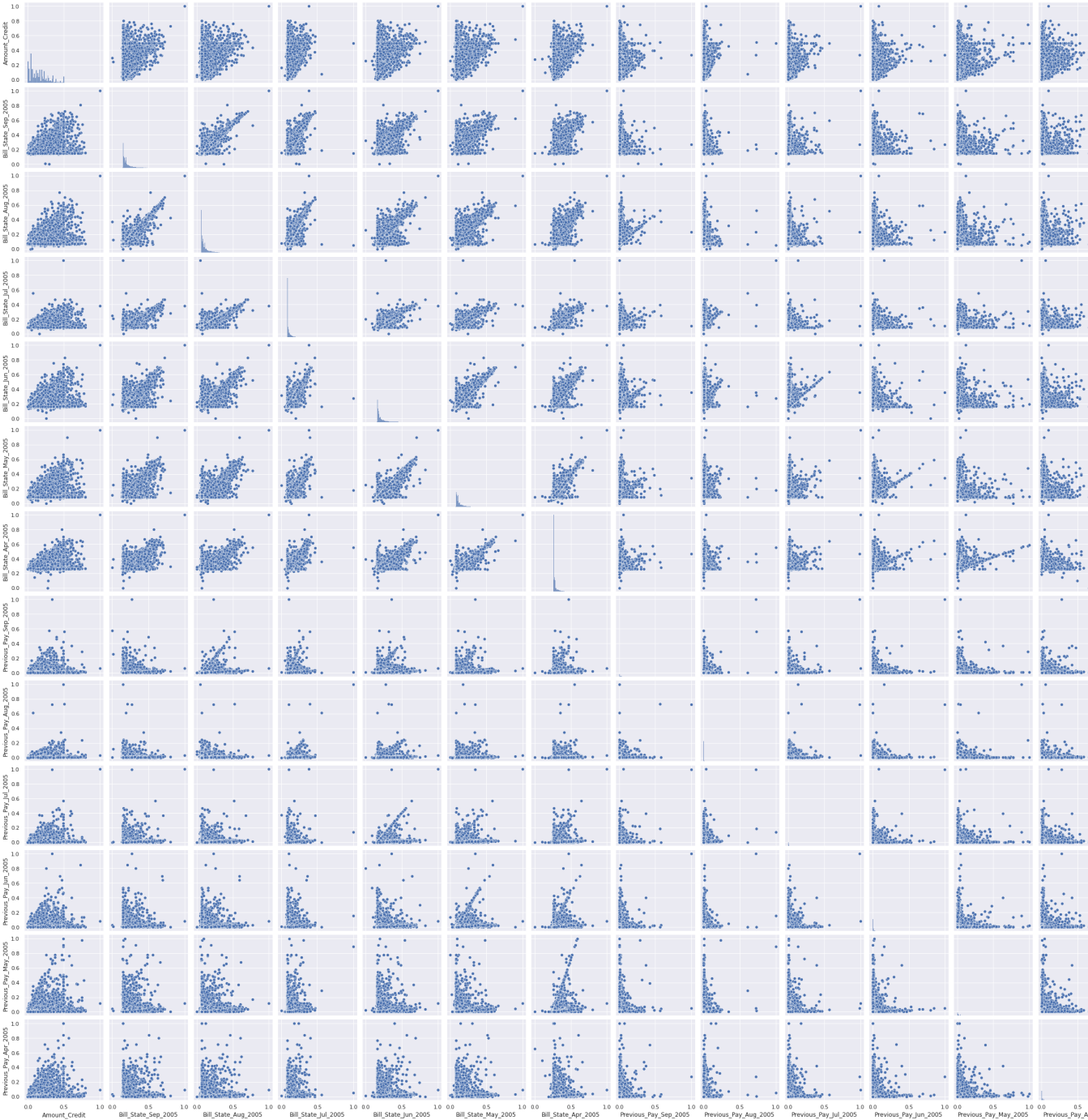
<seaborn.axisgrid.PairGrid at 0x7fda3a7e3dd0>





```
sns.pairplot(scaled_X)
```

<seaborn.axisgrid.PairGrid at 0x7fda34f1ad50>



Grafica 3

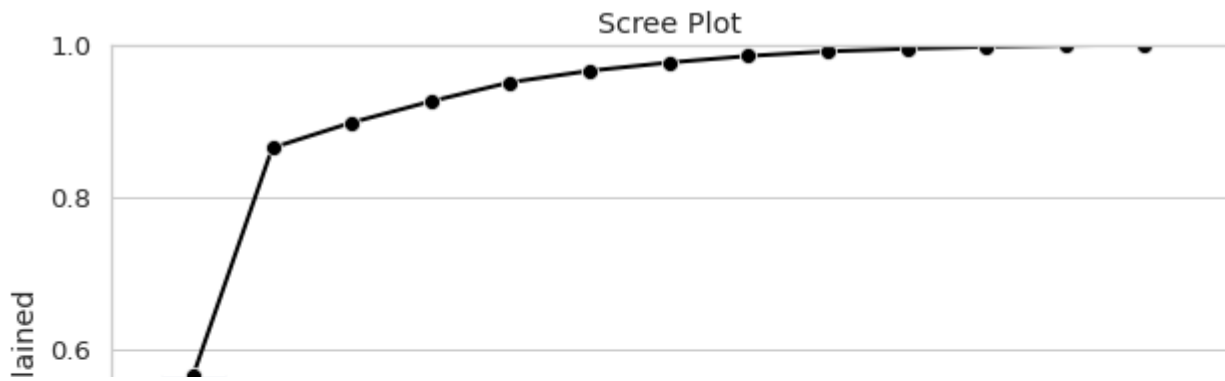
```
PC_components = np.arange(pcs.n_components_) + 1

_ = sns.set(style = 'whitegrid',
            font_scale = 1.2
            )
#### c. Gráfica 3
fig, ax = plt.subplots(figsize=(10, 7))

_ = sns.barplot(x = PC_components,
                y = pcs.explained_variance_ratio_,
                color = 'b'
                )

_ = sns.lineplot(x = PC_components-1,
                 y = np.cumsum(pcs.explained_variance_ratio_),
                 color = 'black',
                 linestyle = '-',
                 linewidth = 2,
                 marker = 'o',
                 markersize = 8
                 )

plt.title('Scree Plot')
plt.xlabel('N-th Principal Component')
plt.ylabel('Variance Explained')
plt.ylim(0, 1)
plt.show()
```



▼ 10. Interpreta y explica cada uno de los graficos

at

Diagrama Boxplot

Con este diagrama podemos observar la distribucion de cada elemento, observando el sesgo que presentan estas mismas, siendo positivos o negativos. El diagrama consta de 4 cuartiles, distribuidos en una caja dividida y dos bigote.

en el primer boxplot nos enfocamos en el PCA, donde podemos observar que los Componente 7, 10 y 11 son los que presentan una mayor varianza pero sin presentar outliers. Por el otro lado el componente 13 tiene una caja y bigotes reducida, pero con una gran cantidad de outliers.

En el segundo box plot se muestran las columnas y su distribucion, lo que nos permite ver el comportamiento mes a mes de cada variable, lo que nos podria ayudar a detectar anomalias en algun mes en especifico.

Diagrama Pairplot

El pairplot nos permite observar el comportamiento que presentan las variables con respecto a las componentes. Nos permiten visualizar como se correlacionan entre si, o en otras palabras, como las variables forman parte del PC.

Diagrama Scree Plot

Esta ultima grafica nos despliega la varianza explicada, las barras es la varianza de cada una de las componentes, mientras que la grafica de linea nos muestra la varianza acumulada. Visualmente, podemos observar que sobrepasamos el 90% de la variacion total en la componente PC4. Tambien podemos observar que las componentes que mas aportan a la variacion son las dos primeras.

<https://colab.research.google.com/drive/1fjr0cUbjTr98IXqJor7X3rWS4iUuukY7#scrollTo=F-B7uQsjv7uL>