SEMANA 6

JIRAM CESAR VILLALPANDO GUERRERO

```
#Importamos las librerías básicas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

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

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

```
print(df)
```

```
            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
29996     5190.0      0.0   1837.0   3526.0   8998.0   129.0      0.0     0.0
29997    20582.0  19357.0      0.0      0.0  22000.0  4200.0   2000.0  3100.0
29998    11855.0  48944.0  85900.0   3409.0   1178.0  1926.0  52964.0  1804.0
29999    32428.0  15313.0   2078.0   1800.0   1430.0  1000.0   1000.0  1000.0

              Y
```

```
0      1.0
1      1.0
2      0.0
3      0.0
4      0.0
...    ...
29995  0.0
29996  0.0
29997  1.0
29998  1.0
29999  1.0

[30000 rows x 25 columns]
```

```
print('Tamano de data set es:')
print(df.shape)
print('DataSet contiene:')
print(df.columns)
print('Encabezadosson:')
print(df.head())
print('Tipos de datos:')
print(df.dtypes)
print('Resumen de datos:-')
print(df.info())
print('Los datos nulos:')
print(df.isna())
```

```
 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
Los datos nulos:
          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
```

```
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]
```

#

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

```
categoricas=['genero','educacion','estado_civil', 'pago_sept','pago_ago','pago_jul','pago_
numericas = ['1','2','3','4','5','6','7','8','9','10','11','12']

for columnas in df.columns:
  promedio = df[columnas].mean()
  df[columnas].fillna(value = promedio, inplace=True)
print('total de datos nulos de mi Data Set son:------------------------------')
print(df.isna().sum())
```

```
total de datos nulos de mi Data Set son:------------------------------
ID     0
X1     0
X2     0
X3     0
X4     0
X5     0
X6     0
X7     0
X8     0
X9     0
X10    0
X11    0
```

```
X12    0
X13    0
X14    0
X15    0
X16    0
X17    0
X18    0
X19    0
X20    0
X21    0
X22    0
X23    0
Y      0
dtype: int64
```

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

```
print(df.describe())
df.describe()
```

	ID	X1	X2	X3	X4	\
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	
mean	15000.500000	167484.322667	1.603753	1.853057	1.551903	
std	8660.398374	129747.661567	0.489117	0.790293	0.521950	
min	1.000000	10000.000000	1.000000	0.000000	0.000000	
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	

	X5	X6	X7	X8	X9	\
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	
mean	35.484214	-0.016635	-0.133689	-0.166405	-0.220800	
std	9.217256	1.123773	1.197154	1.195908	1.168977	
min	21.000000	-2.000000	-2.000000	-2.000000	-2.000000	
25%	28.000000	-1.000000	-1.000000	-1.000000	-1.000000	
50%	34.000000	0.000000	0.000000	0.000000	0.000000	
75%	41.000000	0.000000	0.000000	0.000000	0.000000	
max	79.000000	8.000000	8.000000	8.000000	8.000000	

	...	X15	X16	X17	X18	\
count	...	30000.000000	30000.000000	30000.000000	30000.000000	
mean	...	43275.652326	40324.493980	38881.135745	5662.945886	
std	...	64329.411150	60792.752471	59551.384923	16561.956313	
min	...	-170000.000000	-81334.000000	-339603.000000	0.000000	
25%	...	2332.000000	1769.500000	1258.500000	1000.000000	
50%	...	19066.000000	18123.000000	17100.500000	2100.000000	
75%	...	54506.000000	50177.000000	49198.250000	5007.000000	
max	...	891586.000000	927171.000000	961664.000000	873552.000000	

	X19	X20	X21	X22	\
count	3.000000e+04	30000.000000	30000.000000	30000.000000	
mean	5.922489e+03	5225.623400	4827.252526	4800.297209	
std	2.304072e+04	17606.074601	15665.879011	15278.040231	
min	0.000000e+00	0.000000	0.000000	0.000000	
25%	8.360000e+02	390.000000	298.000000	254.750000	
50%	2.010000e+03	1800.000000	1500.000000	1500.000000	
75%	5.000000e+03	4512.000000	4016.500000	4043.750000	

Realiza el conteo de las variables categóricas

Escala los datos, si consideras necesario

```
X = df.loc[:, df.columns != 'Y'].values

Y = df['Y'].values
X_escalada = StandardScaler().fit_transform(X)
df_escalado = pd.DataFrame(X_escalada)
df_escalado
```

	0	1	2	3	4	5	6	
0	-1.731993	-1.136720	0.810140	0.185938	-1.057405	-1.245968	1.794552	1.7823
1	-1.731878	-0.365981	0.810140	0.185938	0.858519	-1.028980	-0.875071	1.7823
2	-1.731762	-0.597202	0.810140	0.185938	0.858519	-0.161028	0.014803	0.1116
3	-1.731647	-0.905498	0.810140	0.185938	-1.057405	0.164454	0.014803	0.1116
4	-1.731531	-0.905498	-1.234395	0.185938	-1.057405	2.334333	-0.875071	0.1116
...	...	...	...	...	...	...	...	
29995	1.731531	0.404759	-1.234395	1.451312	-1.057405	0.381442	0.014803	0.1116
29996	1.731647	-0.134759	-1.234395	1.451312	0.858519	0.815417	-0.875071	-0.7236
29997	1.731762	-1.059646	-1.234395	0.185938	0.858519	0.164454	3.574301	2.6176

Reduce las dimensiones con PCA, si consideras necesario.

```
mi_pca = PCA(n_components = 10)
componentes_principales = mi_pca.fit_transform(X_escalada)

print('Reduccion PCA: ', componentes_principales.shape)

df_con_pca = pd.DataFrame( data=componentes_principales, columns=['PC1', 'PC2', 'PC3', 'PC

target_names = {0.0:'Ladron',1.0:'No ladron'}

df_con_pca['Y'] = Y

df_con_pca['Y'] = df_con_pca['Y'].map(target_names)

df_con_pca.head()

print(mi_pca.explained_variance_ratio_)
mi_pca.explained_variance_ratio_.sum()
df_con_pca
```

```
Reduccion PCA:  (30000, 10)
[0.27262814 0.1708037  0.06470636 0.06140222 0.04366079 0.04106471
 0.03956639 0.03780478 0.03696943 0.03629662]
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	I
0	-1.899818	-0.945019	-0.378367	-0.641522	-0.188392	-1.783357	0.927361	-0.494
1	-0.776972	-2.148734	1.199100	-0.546535	-0.194729	-1.637933	0.525055	0.168

Indica la varianza de los datos explicada por cada componente seleccionado.

3	-0.208204	-0.848400	-0.723996	0.181109	-0.318532	-1.878022	0.597673	0.360

```
mi_pca.explained_variance_
```

```
array([6.54329339, 4.09942544, 1.55300449, 1.47370239, 1.04789397,
       0.98558577, 0.949625  , 0.90734486, 0.88729598, 0.87114803])
```

29996	-1.764310	-0.010414	-0.529504	0.616481	0.622690	2.313207	0.648460	-0.333

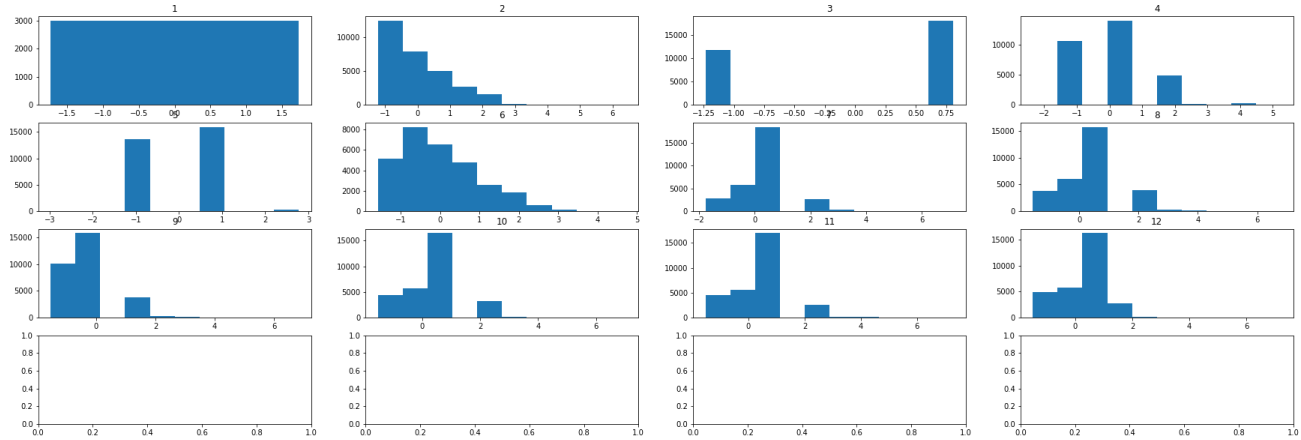
Para actividades de exploración de los datos la varianza > 70%

Indica la importancia de las variables en cada componente Elabora los histogramas de los atributos para visualizar su distribución Realiza la visualización de los datos usando por lo menos 3 gráficos que consideres adecuados: plot, scatter, jointplot, boxplot, areaplot, pie chart, pairplot, bar chart, etc. Interpreta y explica cada uno de los gráficos indicando cuál es la información más relevante que podría ayudar en el proceso de toma de decisiones

```
df_escalado.columns
```

```
RangeIndex(start=0, stop=24, step=1)
```

```
lista_de_numericas = numericas

grafica_df = df_escalado[[0,1,2,3,4,5,6,7,8,9,10,11,12]]
fig, axs = plt.subplots(4, 4, figsize=(30,10))

for i in range(len(lista_de_numericas)):
    plt.subplot(4,4,i+1)
    plot_1 = grafica_df[grafica_df.columns[i]]
    plt.hist(plot_1)
    plt.title(lista_de_numericas [i])

plt.show()
```

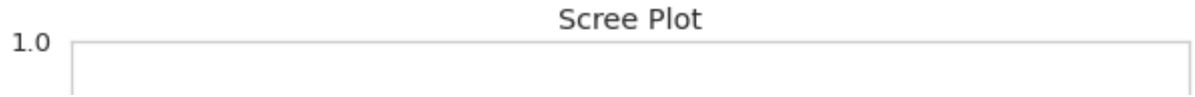



```
sns.set()

plt.bar(
    range(1,len(mi_pca.explained_variance_)+1),
    mi_pca.explained_variance_
    )

plt.plot(
    range(1,len(mi_pca.explained_variance_)+1),
    np.cumsum(mi_pca.explained_variance_),
    c='red',
    label='Suma de varianza')
```

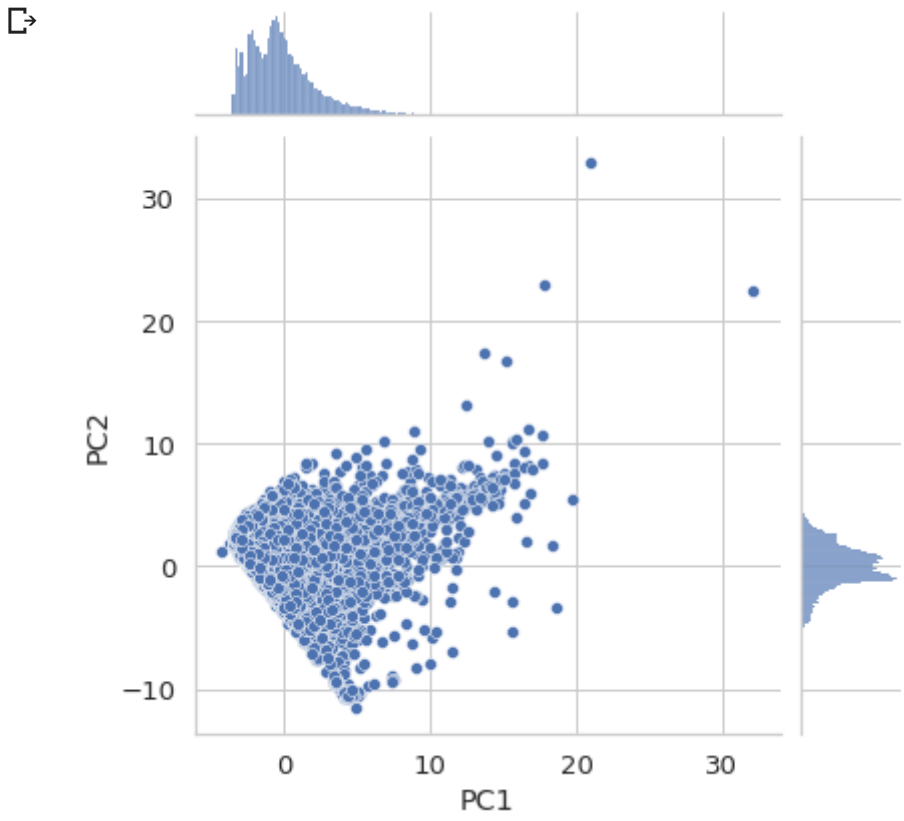
```
[<matplotlib.lines.Line2D at 0x7f919964cb50>]
```


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


_ = sns.set(style = 'whitegrid',
            font_scale = 1.2
            )

fig, ax = plt.subplots(figsize=(10, 7))

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

_ = sns.lineplot(x = PC_components-1,
                 y = np.cumsum(mi_pca.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)
```

```
(0.0, 1.0)
```




```
sns.jointplot(x = "PC1", y = "PC2", kind = "scatter", data = df_con_pca)
plt.show()
```



+ Código + Texto

Interpreta y explica cada uno de los gráficos indicando cuál es la información más relevante que podría ayudar en el proceso de toma de decisiones.

Son 3 gráficos: el primero nos indica la suma de la varianza El segundo es un plot de la varianza y principales componentes y finalmente un joinplot entre dos PC.

En esta práctica se usaron distintos plot con base a estadística.

Productos de pago de Colab - Cancelar contratos

✓ 3 s completado a las 23:55