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

Lectura de Datos

```
df= pd.read_csv( 'https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje-/main/default%20of%20credi
```

```
df.head()
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X17	X18	X19	X20	X21
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	0.0	0.0
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	1000.0	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	1000.0	1000.0
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	1200.0	1100.0
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	10000.0	9000.0

5 rows × 25 columns

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

```
df.shape
```

```
(30000, 25)
```

```
df.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')
```

```
df.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
```

```
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   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
```

```
df.isna()
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X17	X18	X19	X20	X21
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
29996	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
29997	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
29998	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False
29999	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False

30000 rows × 25 columns

```
df.isnull().values.any()
```

```
True
```

```
df.isnull().any()
```

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

```
df.isna().any()
```

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

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

```
df.dropna(inplace = True)
df
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X17	X18	X19	X20	
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	0.0	
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	1000.0	1
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	1000.0	1
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	1200.0	1
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	10000.0	9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1.0	3.0	1.0	39.0	0.0	0.0	0.0	0.0	...	88004.0	31237.0	15980.0	8500.0	20000.0	5003.0	3
29996	29997	150000	1.0	3.0	2.0	43.0	-1.0	-1.0	-1.0	-1.0	...	8979.0	5190.0	0.0	1837.0	3526.0	8998.0	
29997	29998	30000	1.0	2.0	2.0	37.0	4.0	3.0	2.0	-1.0	...	20878.0	20582.0	19357.0	0.0	0.0	22000.0	4
29998	29999	80000	1.0	3.0	1.0	41.0	1.0	-1.0	0.0	0.0	...	52774.0	11855.0	48944.0	85900.0	3409.0	1178.0	1
29999	30000	50000	1.0	2.0	1.0	46.0	0.0	0.0	0.0	0.0	...	36535.0	32428.0	15313.0	2078.0	1800.0	1430.0	1

29975 rows × 25 columns

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

```
df.describe()
```

	ID	X1	X2	X3	X4	X5	X6	X7	
count	29975.000000	29975.000000	29975.000000	29975.000000	29975.000000	29975.000000	29975.000000	29975.000000	2
mean	14999.137581	167538.604837	1.604070	1.853078	1.551827	35.485486	-0.016981	-0.134012	
std	8657.789302	129742.083982	0.489058	0.790468	0.521994	9.217725	1.123791	1.197162	

```
df.shape
```

```
(29975, 25)
```

50%	14999.000000	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.000000

Realiza el conteo de las variables categóricas.

```
numericas=df.drop(['ID','X2','X3','X4','X6','X7','X8','X9','X10','X11','Y'], axis=1)
```

```
numericas.shape
```

```
(29975, 14)
```

El conteo de variables categoricas es el numero de variables total menos el numero de variables categoricas. El dataframe numericas solo queda conformado por 14 variables numericas. Por lo tanto el conteo de variables categoricas es 11.

Escala los datos, si consideras necesario.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled = scaler.fit_transform(numericas)
#Son muchos valores. Así que imprimamos los primeros 5 resultados mejor.
scaled[:5]
```

```
array([[-1.13718742, -1.24604254, -0.64248634, -0.6473368 , -0.66789986,
        -0.67243242, -0.66304357, -0.65268019, -0.34190246, -0.22712701,
        -0.29676957, -0.30811617, -0.31414687, -0.29346219],
       [-0.36641463, -1.02906565, -0.6591989 , -0.6666781 , -0.63916972,
        -0.621586  , -0.6062293 , -0.59793705, -0.34190246, -0.21363445,
        -0.23999307, -0.24430798, -0.31414687, -0.1810025 ],
       [-0.59764647, -0.1611581 , -0.29865028, -0.49388461, -0.48237208,
        -0.4497307 , -0.41723755, -0.39165562, -0.25028162, -0.19194222,
        -0.23999307, -0.24430798, -0.24871751, -0.01231298],
       [-0.90595558,  0.16430723, -0.0576555 , -0.01342823,  0.03272337,
        -0.23243685, -0.18683962, -0.15666804, -0.22118989, -0.16942569,
        -0.22863777, -0.23792716, -0.24420288, -0.23723234],
       [-0.90595558,  2.33407612, -0.57862293, -0.61126676, -0.1612519 ,
        -0.34702777, -0.34820532, -0.33152378, -0.22118989,  1.334366  ,
         0.27099537,  0.2661576 , -0.26906604, -0.25528212]])
```

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

	X1	X5	X12	X13	X14	X15	X16	X17	X18	X19	X20	
0	-1.137187	-1.246043	-0.642486	-0.647337	-0.667900	-0.672432	-0.663044	-0.652680	-0.341902	-0.227127	-0.296770	-0.3
1	-0.366415	-1.029066	-0.659199	-0.666678	-0.639170	-0.621586	-0.606229	-0.597937	-0.341902	-0.213634	-0.239993	-0.2
2	-0.597646	-0.161158	-0.298650	-0.493885	-0.482372	-0.449731	-0.417238	-0.391656	-0.250282	-0.191942	-0.239993	-0.2
3	-0.905956	0.164307	-0.057655	-0.013428	0.032723	-0.232437	-0.186840	-0.156668	-0.221190	-0.169426	-0.228638	-0.2
4	-0.905956	2.334076	-0.578623	-0.611267	-0.161252	-0.347028	-0.348205	-0.331524	-0.221190	1.334366	0.270995	0.2

Reduce las dimensiones con PCA, si consideras necesario.

```
from sklearn.decomposition import PCA
pcs = PCA()
pcs_t = pcs.fit_transform(scaled_df)
```

```
pcsSummary_df = pd.DataFrame({'% varianza explicada': np.round(pcs.explained_variance_ratio_,4) * 100,
'% varianza acumulada': np.cumsum(pcs.explained_variance_ratio_) * 100})
pcsSummary_df
```

	% varianza explicada	% varianza acumulada
0	42.28	42.279619
1	12.26	54.537070
2	7.46	62.001690
3	6.61	68.608539
4	6.31	74.920341
5	6.23	81.153789
6	5.57	86.722019
7	5.19	91.912021
8	5.05	96.963223
9	1.89	98.852053
10	0.51	99.359387
11	0.29	99.652813
12	0.18	99.833971
13	0.17	100.000000

```
pcs_labels = [f'PC{i + 1}' for i in range(len(scaled_df.columns))]
pcsSummary_df.index = pcs_labels
pcsSummary_df
```

	% varianza explicada	% varianza acumulada
PC1	42.28	42.279619
PC2	12.26	54.537070
PC3	7.46	62.001690
PC4	6.61	68.608539
PC5	6.31	74.920341
PC6	6.23	81.153789
PC7	5.57	86.722019
PC8	5.19	91.912021
PC9	5.05	96.963223
PC10	1.89	98.852053
PC11	0.51	99.359387
PC12	0.29	99.652813
PC13	0.18	99.833971
PC14	0.17	100.000000

```
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())

pcsSummary_df
```

```
Varianza total variables originales:  14.000467071461934
Varianza total de los componentes:  14.000467071461943
```

	% varianza explicada	% varianza acumulada
PC1	42.28	42.279619
PC2	12.26	54.537070

Indica la varianza de los datos explicada por cada componente seleccionado. Para actividades de exploración de los datos la varianza > 70%

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

	Porcentaje Varianza	Porcentaje Varianza Acumulado
X1	7.142857	7.142857
X5	7.142857	14.285714
X12	7.142857	21.428571
X13	7.142857	28.571429
X14	7.142857	35.714286
X15	7.142857	42.857143
X16	7.142857	50.000000
X17	7.142857	57.142857
X18	7.142857	64.285714
X19	7.142857	71.428571
X20	7.142857	78.571429
X21	7.142857	85.714286
X22	7.142857	92.857143
X23	7.142857	100.000000

Indica la importancia de las variables en cada componente

```
comps_df = pd.DataFrame(
pcs.components_.round(4),
columns = pcs_df.columns,
index = scaled_df.columns)
comps_df.iloc[:,:7]

comps_df.iloc[:,:7].abs().idxmax()
comps_df.iloc[:,:10].abs().idxmax()
```

```
PC1     X18
PC2     X12
PC3     X19
PC4     X23
PC5     X23
PC6     X20
PC7     X22
PC8     X20
PC9     X17
PC10    X17
dtype: object
```

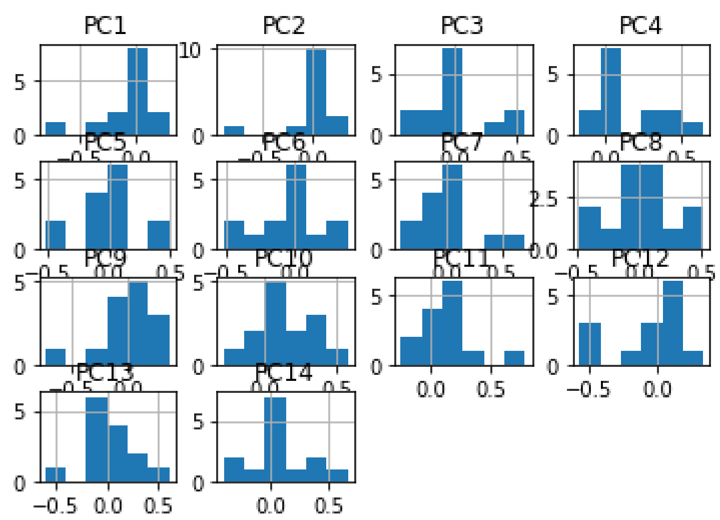
```
comps_df
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14
X1	0.1655	0.0327	0.3724	0.3832	0.3883	0.3915	0.3885	0.3807	0.1352	0.1168	0.1281	0.1169	0.1138	0.1055
X5	0.3008	0.0718	-0.1909	-0.1747	-0.1269	-0.1204	-0.1060	-0.0942	0.3833	0.4083	0.3923	0.3495	0.3041	0.3235
X12	-0.3785	-0.8697	-0.0342	-0.0018	0.0347	0.0340	0.0339	0.0185	0.1735	0.2008	0.1220	0.0622	-0.0605	-0.0507
X13	-0.2004	0.3386	-0.0640	0.0073	0.0605	0.0748	0.0397	-0.0703	0.3613	0.3464	0.2453	-0.0942	-0.6089	-0.3672
X14	0.0347	-0.0390	0.0411	0.0831	0.1142	0.0286	-0.1070	-0.1649	0.2262	0.1506	-0.2392	-0.5795	-0.1928	0.6573
X15	-0.0783	0.0711	-0.0440	-0.0290	0.0988	0.0144	-0.0990	0.0698	0.0398	0.4072	-0.1079	-0.4992	0.6036	-0.4110
X16	0.1112	-0.0787	0.0082	-0.0323	-0.1213	0.1264	-0.0076	0.0080	-0.2011	-0.2796	0.7852	-0.4621	0.0147	0.0253
X17	-0.0493	0.0285	0.0095	-0.1357	0.0928	0.0392	0.0498	0.0001	-0.7490	0.5778	0.0695	0.0777	-0.1631	0.1825

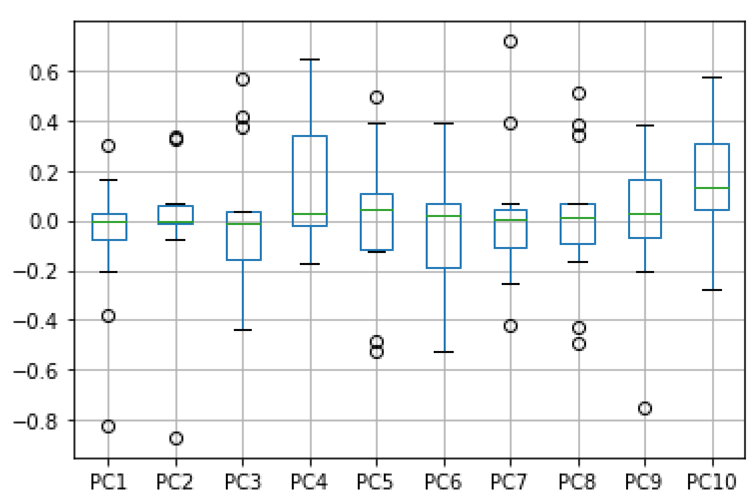
Elabora los histogramas de los atributos para visualizar su distribución

X20	-0.0062	0.0001	0.4159	0.0384	-0.4845	-0.5233	0.0680	0.5136	0.0476	0.1472	0.0002	-0.1158	-0.0995	0.0350

```
hist = comps_df.hist(bins=6)
```

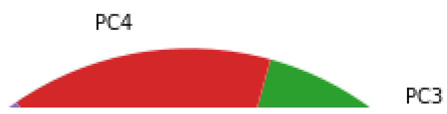



```
boxplot = comps_df.boxplot(column=['PC1', 'PC2', 'PC3', 'PC4','PC5', 'PC6', 'PC7', 'PC8','PC9', 'PC10'])
```




```
from matplotlib import pyplot as plt

Language = ['PC1', 'PC2', 'PC3', 'PC4','PC5', 'PC6', 'PC7', 'PC8','PC9', 'PC10']
data = [0.1655, 0.0328, 0.3724, 0.3833, 0.3883, 0.3916, 0.3885, 0.3807, 0.1351, 0.1168]
# Creating plot
fig = plt.figure(figsize =(10, 7))
plt.pie(data, labels = Language)
# show plot
plt.show()
```



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.

Como se puede observar en el histograma, las variables má
después de aplicar la tecnica de PCA tienen una menor var
distribución es uniforme. Lo mismo se puede observar con
En el ultimo grafico de pie, podemos ver cómo estan distr
cada componente que se analizó.
Con ello se podemos eliminar casi la mitad de variables d
original.

Como se puede observar en el histograma, las variables más representativas después de aplicar la tecnica de PCA tienen una menor varianza y su distribución es uniforme. Lo mismo se puede observar con la grafica de Boxplot. En el ultimo grafico de pie, podemos ver cómo estan distribuidos los pesos de cada componente que se analizó. Con ello se podemos eliminar casi la mitad de variables del set de datos original.