Actividad Semanal 6, Visualizacion

Hector Manuel Gonzalez Villarreal A00178679

Prof: Maria de Paz Rico

```
import pandas as pd
import numpy as np
import seaborn as sns

df.=.pd.read_csv('./sample_data/datossemana7.csv',.index_col=0)
df.describe()
```

₽		X1	X2	Х3	Х4	Х5	
	count	30000.000000	29999.000000	29998.000000	29998.000000	29995.000000	29997.0000
	mean	167484.322667	1.603753	1.853057	1.551903	35.484214	-0.0166
	std	129747.661567	0.489125	0.790320	0.521968	9.218024	1.1238
	min	10000.000000	1.000000	0.000000	0.000000	21.000000	- 2.000C
	25%	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000C
	50%	140000.000000	2.000000	2.000000	2.000000	34.000000	0.0000
	75%	240000.000000	2.000000	2.000000	2.000000	41.000000	0.0000
	max	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.0000

8 rows × 24 columns



#cehcamos si hay nulos
df.isnull().any()

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
             True
     X18
             True
     X19
     X20
             True
     X21
             True
     X22
             True
     X23
             True
             True
     Υ
     dtype: bool
#revisamos datos validos entre las columnas segun el modelo
#Columna Gender
df.groupby(['X2']).size()
#todos los datos son validos
     X2
     1.0
            11887
     2.0
            18112
     dtype: int64
df.groupby(['X3']).size()
#Aqui en esta columna encontramos datos no validos los datows perminitidos son 1,2,3 y 4 y ti
menoresde1 = df[ (df['X3'] < 1)].index
mayoresde4 = df[(df['X3'] > 4)].index
df.drop(menoresde1 , inplace=True)
df.drop(mayoresde4 , inplace=True)
df.groupby(['X3']).size()
     Х3
     1.0
            10585
     2.0
            14030
     3.0
             4915
     4.0
              123
     dtype: int64
# Columna Marital Estatus
df.groupby(['X4']).size()
#Aqui en esta columna encontramos datos no validos los datow perminitidos son 1,2,3 tienes va
menoresde1 = df[ (df['X4'] < 1)].index
df.drop(menoresde1 , inplace=True)
df.groupby(['X4']).size()
```

1.0

13475

```
2.0
            15806
     3.0
              318
     dtype: int64
# Columnas de Past Payment X6 a la X11
cols = ['X6', 'X7', 'X8', 'X9', 'X10', 'X11']
#df.loc[:,cols].describe()
#Aqui encontramos que el valor de dato minimo es de -2 siendo el permitido -1, se eliminaran
menoresdemenos1 = df[ (df['X6'] < -1)].index</pre>
df.drop(menoresdemenos1 , inplace=True)
menoresdemenos1 = df[(df['X7'] < -1)].index
df.drop(menoresdemenos1 , inplace=True)
menoresdemenos1 = df[ (df['X8'] < -1)].index</pre>
df.drop(menoresdemenos1 , inplace=True)
menoresdemenos1 = df[ (df['X9'] < -1)].index</pre>
df.drop(menoresdemenos1 , inplace=True)
menoresdemenos1 = df[ (df['X10'] < -1)].index</pre>
df.drop(menoresdemenos1 , inplace=True)
menoresdemenos1 = df[ (df['X11'] < -1)].index</pre>
df.drop(menoresdemenos1 , inplace=True)
#datos.loc[:,cols].describe()
#con esta limpia de datos correctos la base baja a 23,117 es decir un 77.05%
df.dropna(inplace=True)
df.isnull().any()
     X1
            False
     X2
            False
     Х3
            False
     Χ4
            False
     X5
            False
     Х6
            False
     X7
            False
     X8
            False
     Х9
            False
     X10
            False
     X11
            False
     X12
            False
     X13
            False
     X14
            False
     X15
            False
     X16
            False
     X17
            False
     X18
            False
     X19
            False
     X20
            False
     X21
            False
     X22
            False
     X23
            False
```

Y False

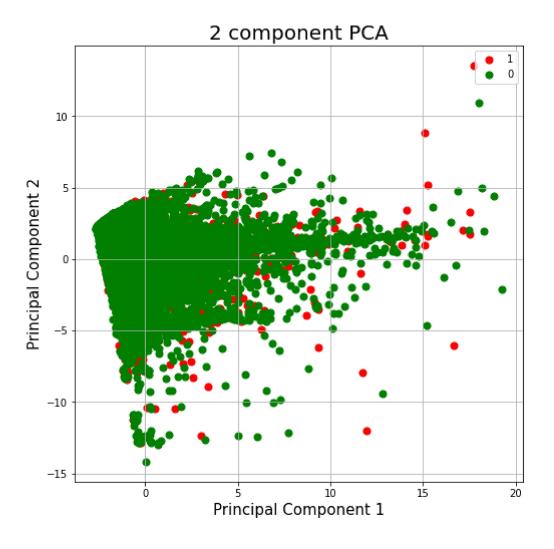
df.describe()

	X1	X2	Х3	X4	Х5	3
count	23117.000000	23117.000000	23117.000000	23117.000000	23117.000000	23117.00000
mean	156350.896743	1.591945	1.851797	1.564260	35.240732	0.1803(
std	127580.413394	0.491484	0.699920	0.518718	9.286893	0.9850
min	10000.000000	1.000000	1.000000	1.000000	21.000000	-1.0000(
25%	50000.000000	1.000000	1.000000	1.000000	28.000000	0.00000
50%	120000.000000	2.000000	2.000000	2.000000	34.000000	0.0000(
75%	220000.000000	2.000000	2.000000	2.000000	41.000000	0.00000
max	1000000.000000	2.000000	4.000000	3.000000	79.000000	8.00000

8 rows × 24 columns



```
from sklearn.preprocessing import StandardScaler
features = ['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9', 'X10', 'X11', 'X12', 'X13', 'X1
# Separating out the features
x = df.loc[:, features].values
# Separating out the target
y = df.loc[:,['Y']].values
# Standardizing the features
x = StandardScaler().fit transform(x)
from sklearn.decomposition import PCA
pca = PCA(n components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['pc1', 'pc2'])
finalDf = pd.concat([principalDf, df[['Y']]], axis = 1)
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
```



```
df.corr()
corrs = df.corr()
sns.set(rc = {'figure.figsize':(15,10)})
sns.heatmap(corrs, vmin = -1, vmax = 1, cmap = "BuGn", annot= True)
# Aqui vemos que el las correlaciones fuerets estan entre la X12 y X17
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f2476d512d0>

```
1 0.0190.240.0980.12-0.23-0.25-0.24-0.23-0.22-0.21 0.4 0.4 0.4 0.41 0.41 0.4 0.22 0.22 0.24 0.23 0.25 0.24-0.1
                           1 0.01 f0.03 ±0.11 0.03 ±0.04 B.04 B.04 D.04 D.03 ±0.03 ±0.01 ±0.01 ±0.01 ±0.04 D.03 ±0.
Q
          -0.09800330.16 1 -0.40.00359e-0070120.0110.0120.0410.040.0420.040.040.030.0062004500330.00.0040100600.03
          9
         -0.250.0430.13.9e-07.02 0.71 1 0.68 0.57 0.53 0.48 0.1 0.09 0 0970.09 7 0.1 0.11-0.150.09 0 0 4 0.740.068 0.650.3
D
         -0.240.0410.10.0120.0270.580.68 1 0.7 0.6 0.540.06m.08m.08m.0870.0940.0970.0490.130.0940.0810.070.0670.3
         -0.230.040.090.0110.0210.54 0.57 0.7 1 0.75 0.640.050.068 0880.090.096.090.0660.050.140.090.074.0610.2
X21 X20 X19 X18 X17 X16 X15 X14 X13 X12 X11 X10
          -0.220.03 \cdot 0.070.0120.0260.5 \quad 0.53 \quad 0.6 \quad 0.75 \quad 1 \quad 0.750.0630.0730.0890.11 \quad 0.12 \quad 0.120.0640.0480.0460.140.0810.060.2
          -0.210.030.060.0150.020.46 0.48 0.54 0.64 0.75 1 0 0640.0740 0860.11 0.13 0.13 0.0540.0540.0540.130.0740.2
            0.4 \cdot 0.016 \cdot 0.0329 \cdot 0.040 \cdot 0.0810 \cdot 12 \cdot 0.1 \cdot 0.066 \cdot 0.055 \cdot 0.062 \cdot 0.064 \cdot 1 \cdot 0.96 \cdot 0.92 \cdot 0.88 \cdot 0.85 \cdot 0.82 \cdot 0.14 \cdot 0.12 \cdot 0.16 \cdot 0.17 \cdot 0.19 \cdot 0.05 \cdot 0.012 \cdot 
            0.4-0.01-5.0360.040.0790.130.0990.0880.0680.0715.0740.96 1 0.95 0.9 0.870.850.270.120.160.160.170.180.02
            0.4-0.01-D.04-D.04-D.040.0790.120.0970.0840.0840.0840.0850.0860.92 0.95 1 0.94 0.9 0.87 0.24 0.27 0.13 0.15 0.17 0.19 0.02
          0.410.0091.054.040.0760.120.0970.0870.09 0.11 0.11 0.88 0.9 0.94 1 0.95 0.91 0.23 0.23 0.28 0.13 0.17 0.19 0.0
          0.440.0064.0540.040.0730.13 0.1 0.0940.0960.12 0.13 0.85 0.87 0.9 0.95 1 0.95 0.22 0.2 0.25 0.28 0.14 0.170.01
           0.40.0073.0540.030.0710.13\ 0.110.0970.0990.12\ 0.13\ 0.82\ 0.85\ 0.87\ 0.91\ 0.95\quad 1\quad 0.2\ 0.17\ 0.23\ 0.24\ 0.29\ 0.120.01
          0.240.00148.059.006020260.0920.150.0440.0640.0640.0540.14 0.27 0.24 0.23 0.22 0.2 1 0.36 0.27 0.22 0.14 0.160.07
          0.240.00102.050.004050290.0802.0980.13-0.050.0409.05@.12 0.12 0.27 0.23 0.2 0.17 0.36
                                                                                                                                                                                                                                                                    1 0.29 0.21 0.14 0.190.06
          0.240.0076.065.003030330.0749.0940.0960.140.0440.0510.16 0.16 0.13 0.28 0.25 0.23 0.27 0.29
           0.240.0028.0560.010.0220.0749.0749.0840.0910.140.0440.17.0.16.0.15.0.13.0.28.0.24.0.22.0.21.0.22
X22.)
           0.250.0060.0602.004010240.0600.0680.070.0740.0810.130.17.0.17.0.17.0.17.0.14.0.29.0.14.0.14.0.15.0.15
```

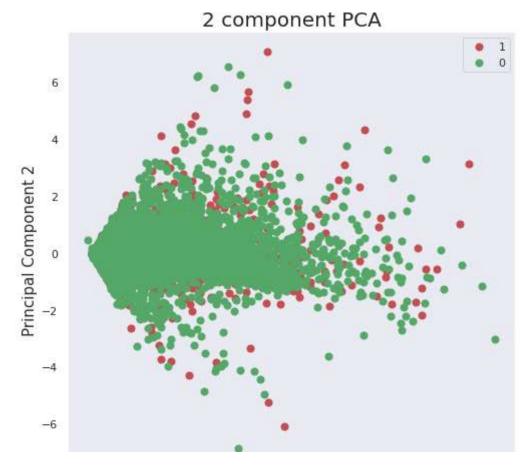
Buscando una correlacion de las variables vemos que las variables X12 a 17 son las que tienen mayor correlacion entre las 23, haremos un df solo con esas a continuación

```
ndf = df[['X12', 'X13', 'X14', 'X15', 'X16', 'X17','Y']]
# Cambiamos nombres para mejor entendimiento o claridad
ndf.columns=['Sept', 'Ago', 'Jul', 'Jun', 'May', 'Abr', 'Pago']
ndf = ndf.dropna(axis=0)
#ndf.columns revision de que el renombrado esta correcto
var=np.cov([ndf.Sept,ndf.Ago,ndf.Jul,ndf.Jun,ndf.May,ndf.Abr,ndf.Pago])
t=ndf.var().sum()
VarSept = var[0][0]
VarAgo = var[1][1]
VarJul = var[2][2]
VarJun = var[3][3]
VarMay = var[4][4]
VarAbr= var[5][5]
VarPago= var[6][6]
print('Varianza Total \t\t\t :', t.round(2))
print('Varianza de Septiembre \t\t :', ((VarSept/t)*100).round(2), '%')
print('Varianza de Agosto \t\t :', ((VarAgo/t)*100).round(2), '%')
```

```
print('Varianza de Julio \t\t :', ((VarJul/t)*100).round(2), '%')
print('Varianza de Junio \t\t :', ((VarJun/t)*100).round(2), '%')
print('Varianza de Mayo \t\t :', ((VarMay/t)*100).round(2), '%')
print('Varianza de Abril \t\t :', ((VarAbr/t)*100).round(2), '%')
print('Varianza de Pago \t\t :', ((VarPago/t)*100).round(2), '%')
    Varianza Total
                                    : 29706264599.69
    Varianza de Septiembre
                                  : 20.3 %
    Varianza de Agosto
                                   : 19.06 %
                                  : 17.65 %
    Varianza de Julio
    Varianza de Junio
                                   : 15.61 %
                                   : 13.95 %
    Varianza de Mayo
    Varianza de Abril
                                   : 13.42 %
    Varianza de Pago
                                   : 0.0 %
```

Aqui con la varianza nos damos cuenta que los valores de septiempre, agosto y julio son los que mas determinan la aprbacion del credito

```
features = ['Sept', 'Ago', 'Jul', 'Jun', 'May', 'Abr']
# Separating out the features
x = ndf.loc[:, features].values
# Separating out the
y = ndf.loc[:,['Pago']].values
# Standardizing the features
x = StandardScaler().fit transform(x)
pca = PCA(n components=2)
principalComponents = pca.fit transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['pc1', 'pc2'])
finalDf = pd.concat([principalDf, ndf[['Pago']]], axis = 1)
fig = plt.figure(figsize = (8,8))
ax = fig.add subplot(1,1,1)
ax.set xlabel('Principal Component 1', fontsize = 15)
ax.set ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = [1, 0]
colors = ['r', 'g']
for target, color in zip(targets, colors):
    indicesToKeep = finalDf['Pago'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'pc1']
               , finalDf.loc[indicesToKeep, 'pc2']
               , c = color
               s = 50
ax.legend(targets)
ax.grid()
```



Aun haciendo el PCA sobre solo las 6 columnas con una buena correlacion nos da el mismo resultado

✓ 0 s completado a las 16:38

