DESARROLLO LABORATORIO 10

In [1]:
```
#Importando librerias necesarias
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math as math
import scipy.stats as stats #Para calculo de probabilidades

from sklearn.decomposition import FactorAnalysis
from sklearn.model_selection import train_test_split #Particionamiento
from sklearn.preprocessing import MinMaxScaler #Utilizar la normalizacion
from sklearn.preprocessing import StandardScaler #Utilizar la estandarizacion
from sklearn.decomposition import PCA #Para la descomposicion de la varianza en el PCA
```

In [2]:
```
os.chdir("D:\Social Data Consulting\Python for Data Science\data")
```

In [3]:
```
miarchivo="nba_logreg2.csv"
df_arrest=pd.read_csv(miarchivo,sep=";")
df_arrest["TARGET_5Yrs"]=df_arrest["TARGET_5Yrs"].astype('int64')
df_arrest.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1329 entries, 0 to 1328
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         1329 non-null   object
 1   GP           1329 non-null   int64
 2   MIN          1329 non-null   float64
 3   PTS          1329 non-null   float64
 4   FGM          1329 non-null   float64
 5   FGA          1329 non-null   float64
 6   FG%          1329 non-null   float64
 7   3P Made      1329 non-null   float64
 8   3PA          1329 non-null   float64
 9   3P%          1329 non-null   float64
 10  FTM          1329 non-null   float64
 11  FTA          1329 non-null   float64
 12  FT%          1329 non-null   float64
 13  OREB         1329 non-null   float64
 14  DREB         1329 non-null   float64
 15  REB          1329 non-null   float64
 16  AST          1329 non-null   float64
 17  STL          1329 non-null   float64
 18  BLK          1329 non-null   float64
 19  TOV          1329 non-null   float64
 20  TARGET_5Yrs  1329 non-null   int64
dtypes: float64(18), int64(2), object(1)
memory usage: 218.2+ KB
```

1. Encontrar el numero de factores adecuados segun el Analisis Factorial.

In [4]:
```
#Separando las variables continuas
continuas = ['GP','MIN','PTS','FGM','FGA','FG%','3P Made','3PA','3P%','FTM','FTA','FT%','OREB'
            ,'DREB','REB','AST','STL','BLK','TOV'
            ]
x = df_arrest.loc[:, continuas].values
#Separando las variable target
y = df_arrest['TARGET_5Yrs'].values
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(x, #predictores
                                                    y, #target
                                                    test_size=0.3, #tamaño de los datos de teste
                                                    stratify=y, # variable de estratificación
                                                    random_state=0) #semilla
```

In [6]:
```
sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)
```

In [7]:
```
fa = FactorAnalysis()
X_train_fa = fa.fit(X_train_std)
```

In [8]:
```
X_train_fa.components_
```

Out[8]:
```
array([[ 5.79636841e-01,  9.38900118e-01,  9.65353599e-01,
         9.55727187e-01,  9.38483162e-01,  2.99375439e-01,
         2.94708405e-01,  3.01805885e-01,  6.71180967e-02,
         8.93810593e-01,  8.94227980e-01,  2.14814278e-01,
         6.72280875e-01,  7.95085949e-01,  7.79112044e-01,
         5.42579872e-01,  7.05298842e-01,  4.90845322e-01,
         8.65495838e-01],
       [-1.79270054e-02,  1.14052616e-01,  9.92892903e-02,
         5.50970434e-02,  1.88725470e-01, -5.26897483e-01,
         7.29842257e-01,  7.51203208e-01,  6.16641230e-01,
        -2.49439249e-02, -1.20783388e-01,  3.92570461e-01,
        -6.04376787e-01, -4.12835096e-01, -5.01476323e-01,
         5.08545964e-01,  2.87879539e-01, -5.16367159e-01,
         1.85858677e-01],
       [ 6.47927583e-02, -7.76258286e-02,  2.61412946e-02,
         2.83778711e-02,  1.14102825e-02,  4.12068962e-02,
        -4.00732452e-01, -3.74512017e-01, -2.32008092e-01,
         1.72647235e-01,  1.58625006e-01,  7.13489303e-02,
        -1.34737134e-01, -2.37170731e-01, -2.06791248e-01,
```

Los factores resultantes deben ser un máximo de 7, no 19, porque los 7 factores tienen conexiones significativas con las características originales.

2. Crear un data frame a partir de los factores sin el target

In [10]:
```python
transformer = FactorAnalysis(n_components=7, #n componentes
                             random_state=0) #semilla
#generamos el objeto transformer de tipo función
df_factores = transformer.fit_transform(X_train_std) #fit_transform aplica a los datos de
#entrenamiento
df_factores = pd.DataFrame(df_factores)
df_factores.columns = ['FACT1', 'FACT2', 'FACT3', 'FACT4', 'FACT5', 'FACT6','FACT7']
df_factores.head()
```

Out[10]:

	FACT1	FACT2	FACT3	FACT4	FACT5	FACT6	FACT7
0	-1.032383	0.586029	0.293037	-0.143571	1.454930	-0.896071	0.337433
1	-0.821369	1.295543	-0.601320	0.177825	-1.042699	0.322726	1.391791
2	-0.355213	0.867851	0.006115	0.199696	0.236346	-0.281901	0.248473
3	-0.670436	1.641298	0.069191	-0.527535	-0.448241	0.065023	0.228539
4	-1.162351	-0.236358	0.472900	-0.119226	0.485008	0.421424	0.458227

3. Encontrar el numero de registros finales despues de aplicar el Z-score con umbral de 3

In [11]:
```python
z = np.abs(stats.zscore(df_factores)) #valor absoluto de las z-score
print(z)
```

```
[[1.03240156e+00 5.86117837e-01 2.93494268e-01 ... 1.49002085e+00
  9.58514356e-01 3.76599510e-01]
 [8.21384231e-01 1.29574061e+00 6.02257872e-01 ... 1.06784745e+00
  3.45215868e-01 1.55333937e+00]
 [3.55219951e-01 8.67983231e-01 6.12451143e-03 ... 2.42046314e-01
  3.01545310e-01 2.77313568e-01]
 ...
 [1.90989811e-01 2.39232189e+00 3.87583755e-01 ... 7.52392553e-01
  1.20719887e+00 5.43719417e-01]
 [5.83105404e-01 3.74924893e-01 1.79550343e+00 ... 2.54558097e+00
  1.06100955e+00 2.28058547e-01]
 [1.11363039e+00 6.12354385e-04 2.32510998e-01 ... 1.18861155e-01
  4.75488939e-01 4.35795042e-01]]
```

In [12]:
```python
k = 3
print(np.where(z > k))
```

```
(array([ 18,  22,  25,  47,  47,  50,  83,  85,  85,  92, 109, 113, 121,
       133, 136, 136, 138, 155, 161, 167, 169, 179, 198, 213, 213, 215,
       240, 246, 253, 259, 268, 294, 352, 355, 355, 374, 384, 395, 396,
       396, 398, 402, 410, 411, 415, 424, 429, 454, 460, 479, 504, 510,
       539, 540, 555, 571, 592, 592, 632, 642, 677, 685, 695, 703, 712,
       719, 722, 724, 739, 752, 760, 760, 771, 802, 806, 806, 834, 838,
       841, 847, 849, 849, 853, 853, 882, 883, 908, 908, 909, 916, 925],
      dtype=int64), array([3, 1, 3, 0, 6, 5, 3, 0, 6, 5, 2, 1, 5, 3, 0, 6, 4, 0, 2, 1, 4, 3,
       0, 1, 3, 5, 3, 6, 2, 3, 2, 2, 5, 0, 3, 1, 5, 3, 3, 5, 2, 0, 6, 3,
       6, 5, 6, 4, 1, 5, 4, 4, 0, 4, 6, 5, 0, 6, 5, 6, 0, 3, 1, 5, 0, 3,
       0, 4, 2, 4, 0, 2, 4, 2, 2, 3, 3, 5, 2, 5, 2, 3, 0, 2, 4, 6, 0, 2,
       5, 6, 3], dtype=int64))
```

In [13]:
```
df_factores_o = df_factores[(z < 3).all(axis=1)]
df_factores_o.head()
```

Out[13]:

	FACT1	FACT2	FACT3	FACT4	FACT5	FACT6	FACT7
0	-1.032383	0.586029	0.293037	-0.143571	1.454930	-0.896071	0.337433
1	-0.821369	1.295543	-0.601320	0.177825	-1.042699	0.322726	1.391791
2	-0.355213	0.867851	0.006115	0.199696	0.236346	-0.281901	0.248473
3	-0.670436	1.641298	0.069191	-0.527535	-0.448241	0.065023	0.228539
4	-1.162351	-0.236358	0.472900	-0.119226	0.485008	0.421424	0.458227

In [14]:
```
len(df_factores)
```

Out[14]:
```
930
```

In [15]:
```
len(df_factores_o)
```

Out[15]:
```
851
```

4. Encontrar el numero de registros finales despues de aplicar el IQR score

In [16]:
```
Q1 = df_factores.quantile(0.25)
Q3 = df_factores.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
FACT1    1.234631
FACT2    1.180184
FACT3    1.043966
FACT4    0.989454
FACT5    1.190952
FACT6    0.957809
FACT7    1.075705
dtype: float64
```

In [17]:
```
print((df_factores < (Q1 - 1.5 * IQR)) | (df_factores > (Q3 + 1.5 * IQR)))
```

```
     FACT1  FACT2  FACT3  FACT4  FACT5  FACT6  FACT7
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
4    False  False  False  False  False  False  False
..     ...    ...    ...    ...    ...    ...    ...
925  False   True   True   True  False  False  False
926  False  False  False  False  False  False  False
927  False   True  False  False  False  False  False
928  False  False  False  False   True  False  False
929  False  False  False  False  False  False  False

[930 rows x 7 columns]
```

In [18]:
```
df_factores_out = df_factores[~((df_factores < (Q1 - 1.5 * IQR)) |(df_factores > (Q3 + 1.5 * IQR
```

In [19]:
```
df_factores_out.head()
```

Out[19]:

	FACT1	FACT2	FACT3	FACT4	FACT5	FACT6	FACT7
0	-1.032383	0.586029	0.293037	-0.143571	1.454930	-0.896071	0.337433
1	-0.821369	1.295543	-0.601320	0.177825	-1.042699	0.322726	1.391791
2	-0.355213	0.867851	0.006115	0.199696	0.236346	-0.281901	0.248473
3	-0.670436	1.641298	0.069191	-0.527535	-0.448241	0.065023	0.228539
4	-1.162351	-0.236358	0.472900	-0.119226	0.485008	0.421424	0.458227

In [20]:
```
len(df_factores)
```

Out[20]: 930

In [21]:
```
len(df_factores_out)
```

Out[21]: 748