lab16

December 2, 2024

1 Laboratorio 16: Laboratorio Final

1.1 Integrante:

- Escriba Flores, Daniel Agustin

---

[6]:
```
# Librerias necesarias

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import mahalanobis



from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

# Configuración warnings
# ==============================
import warnings
warnings.filterwarnings('ignore')
```

2 Pregunta 1

2.1 Seccion A:

Lea dicha data a través de un data frame, determine el tipo para cada variable, contabilice la cantidad de datos perdidos, la cantidad de filas y columnas, elimine la variable 'Concrete compressive strength' y calcule las distancias de Mahalanobis para el resto de variables. Luego elimine aquellas filas cuyas distancias sean mayores a 15 e indique cuántas de estas se eliminaron.

[7]:
```
# Leemos los datos
data = pd.read_csv("concreto.csv")
data
```

[7]:
```
      Cement  Blast Furnace Slag  Fly Ash  Water  Superplasticizer  \
0      540.0                 0.0      0.0  162.0               2.5
1      540.0                 0.0      0.0  162.0               2.5
2      332.5               142.5      0.0  228.0               0.0
3      332.5               142.5      0.0  228.0               0.0
4      198.6               132.4      0.0  192.0               0.0
…        …                   …        …    …                   …
1025   276.4               116.0     90.3  179.6               8.9
1026   322.2                 0.0    115.6  196.0              10.4
1027   148.5               139.4    108.6  192.7               6.1
1028   159.1               186.7      0.0  175.6              11.3
1029   260.9               100.5     78.3  200.6               8.6

      Coarse Aggregate  Fine Aggregate  Age  Concrete compressive strength
0               1040.0           676.0   28                          79.99
1               1055.0           676.0   28                          61.89
2                932.0           594.0  270                          40.27
3                932.0           594.0  365                          41.05
4                978.4           825.5  360                          44.30
…                  …               …    …                            …
1025             870.1           768.3   28                          44.28
1026             817.9           813.4   28                          31.18
1027             892.4           780.0   28                          23.70
1028             989.6           788.9   28                          32.77
1029             864.5           761.5   28                          32.40

[1030 rows x 9 columns]
```

[8]:
```
# Examinamos los datos y sus tipos de variables
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Cement                        1030 non-null   float64
 1   Blast Furnace Slag            1030 non-null   float64
 2   Fly Ash                       1030 non-null   float64
 3   Water                         1030 non-null   float64
 4   Superplasticizer              1030 non-null   float64
 5   Coarse Aggregate              1030 non-null   float64
 6   Fine Aggregate                1030 non-null   float64
 7   Age                           1030 non-null   int64
 8   Concrete compressive strength  1030 non-null   float64
dtypes: float64(8), int64(1)
memory usage: 72.6 KB
```

[9]:
```
#Verificamos que no cuente con valores nulos
data.isnull().sum()
```

[9]:
```
Cement                          0
Blast Furnace Slag              0
Fly Ash                         0
Water                           0
Superplasticizer                0
Coarse Aggregate                0
Fine Aggregate                  0
Age                             0
Concrete compressive strength   0
dtype: int64
```

[10]:
```
# contabilizamos la cantida de filas y columnas
columnasiniciales, filas = data.shape

print(f"El dataset tiene {columnasiniciales} columnas y {filas} filas")
```

```
El dataset tiene 1030 columnas y 9 filas
```

[11]:
```
#Eliminamos la variable 'Concrete compressive strength'

data = data.drop(['Concrete compressive strength'],axis=1)
data
```

[11]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	\
0	540.0	0.0	0.0	162.0	2.5	
1	540.0	0.0	0.0	162.0	2.5	
2	332.5	142.5	0.0	228.0	0.0	
3	332.5	142.5	0.0	228.0	0.0	
4	198.6	132.4	0.0	192.0	0.0	
…	…	…	…	…	…	
1025	276.4	116.0	90.3	179.6	8.9	
1026	322.2	0.0	115.6	196.0	10.4	
1027	148.5	139.4	108.6	192.7	6.1	
1028	159.1	186.7	0.0	175.6	11.3	
1029	260.9	100.5	78.3	200.6	8.6	

	Coarse Aggregate	Fine Aggregate	Age
0	1040.0	676.0	28
1	1055.0	676.0	28
2	932.0	594.0	270
3	932.0	594.0	365
4	978.4	825.5	360
…	…	…	…
1025	870.1	768.3	28
1026	817.9	813.4	28

```
1027             892.4           780.0   28
1028             989.6           788.9   28
1029             864.5           761.5   28

[1030 rows x 8 columns]
```

[12]:
```
# convirtiendo el dataframe a matriz

X = data.to_numpy()
print(X)
```

```
[[ 540.     0.     0.  … 1040.   676.    28. ]
 [ 540.     0.     0.  … 1055.   676.    28. ]
 [ 332.5  142.5    0.  …  932.   594.   270. ]
 …
 [ 148.5  139.4  108.6 …  892.4  780.    28. ]
 [ 159.1  186.7    0.  …  989.6  788.9   28. ]
 [ 260.9  100.5   78.3 …  864.5  761.5   28. ]]
```

[13]:
```
# Vector de medias para cada variables

Medias = np.mean(X,axis = 0)
Medias
```

[13]:
```
array([281.16786408,  73.89582524,  54.18834951, 181.56728155,
         6.20466019, 972.91893204, 773.58048544,  45.66213592])
```

[14]:
```
# Matriz de covariana de las variables
Covarianza = np.cov(X, rowvar=False)
Covarianza

# Inversa de la matriz  de covarianza de las variables

Inversa = np.linalg.inv(Covarianza)
Inversa
```

[14]:
```
array([[ 6.85701486e-04,  7.28241223e-04,  8.93831612e-04,
         2.28446878e-03,  7.20131538e-05,  5.90025165e-04,
         7.41752489e-04,  2.69029278e-05],
       [ 7.28241223e-04,  9.77544477e-04,  1.05057074e-03,
         2.62337652e-03,  9.25996236e-05,  7.20233472e-04,
         8.86507925e-04,  5.84460836e-05],
       [ 8.93831612e-04,  1.05057074e-03,  1.50664357e-03,
         2.97006774e-03, -1.36070035e-03,  7.81092220e-04,
         1.00641670e-03,  7.66984261e-05],
       [ 2.28446878e-03,  2.62337652e-03,  2.97006774e-03,
         1.53594649e-02,  1.88661310e-02,  2.94777884e-03,
         3.29663164e-03, -1.20251101e-04],
```

```
        [ 7.20131538e-05,  9.25996236e-05, -1.36070035e-03,
          1.88661310e-02,  8.30496802e-02,  2.94819022e-03,
          1.51155690e-03, -1.63660375e-04],
        [ 5.90025165e-04,  7.20233472e-04,  7.81092220e-04,
          2.94777884e-03,  2.94819022e-03,  8.39379394e-04,
          7.93250176e-04,  2.30498703e-05],
        [ 7.41752489e-04,  8.86507925e-04,  1.00641670e-03,
          3.29663164e-03,  1.51155690e-03,  7.93250176e-04,
          1.08974436e-03,  4.74180215e-05],
        [ 2.69029278e-05,  5.84460836e-05,  7.66984261e-05,
         -1.20251101e-04, -1.63660375e-04,  2.30498703e-05,
          4.74180215e-05,  2.80261615e-04]])
```

[15]:
```
f = X.shape[0]
valoresmaha = []
for i in range (f):
    valor = mahalanobis(X[i],Medias,Inversa)**2
    valoresmaha.append(valor)
valoresmaha
```

[15]:
```
[np.float64(13.070994050487844),
 np.float64(12.271674998255163),
 np.float64(16.505195087643862),
 np.float64(28.549966164813465),
 np.float64(30.457393999609504),
 np.float64(6.146690597681649),
 np.float64(28.527052648133584),
 np.float64(9.666122000595292),
 np.float64(7.2509210177219945),
 np.float64(14.439064343703444),
 np.float64(3.645016642861315),
 np.float64(3.257005451170856),
 np.float64(17.96174238669625),
 np.float64(6.967772127865922),
 np.float64(7.884711158265409),
 np.float64(9.393566080108023),
 np.float64(12.503685417833719),
 np.float64(27.471086287505514),
 np.float64(8.357624960910524),
 np.float64(14.15414867598596),
 np.float64(11.756650488900451),
 np.float64(11.667746347395699),
 np.float64(11.94024315494591),
 np.float64(17.551124256690475),
 np.float64(28.49125919594152),
 np.float64(17.101400679007927),
 np.float64(16.766958551992687),
```

```
np.float64(9.513694616395327),
np.float64(11.586082986971997),
np.float64(15.496618141075569),
np.float64(27.048046416096547),
np.float64(27.222139581714615),
np.float64(8.042237593213548),
np.float64(20.089546591754562),
np.float64(29.36172502403188),
np.float64(18.78072927001046),
np.float64(9.50436126359365),
np.float64(7.556473532507296),
np.float64(12.7589889287812),
np.float64(11.496861444105866),
np.float64(7.714141215605523),
np.float64(29.437159501808434),
np.float64(31.39485430923749),
np.float64(10.977364295276063),
np.float64(10.09179675966849),
np.float64(12.580708188537773),
np.float64(6.014538369688258),
np.float64(10.29217267216969),
np.float64(10.247272081934039),
np.float64(10.597818606354727),
np.float64(9.760715225793662),
np.float64(9.630346271142768),
np.float64(8.753231786765095),
np.float64(6.631849388130072),
np.float64(11.873101690049028),
np.float64(3.6140783011419075),
np.float64(31.28028672583803),
np.float64(3.710118338861215),
np.float64(8.647157974541635),
np.float64(8.679181384573326),
np.float64(15.20270472951702),
np.float64(15.149056310312119),
np.float64(4.413461576339503),
np.float64(16.833158582983437),
np.float64(8.377754369714971),
np.float64(15.853486185748952),
np.float64(41.26671644009548),
np.float64(9.547949394526409),
np.float64(7.774739847716288),
np.float64(17.9677001387995),
np.float64(4.862248044294715),
np.float64(10.752673582148052),
np.float64(8.5837012776003),
np.float64(7.915860375825624),
```

```
np.float64(17.48132429754765),
np.float64(8.137710270805684),
np.float64(28.5185699478297),
np.float64(8.5837012776003),
np.float64(8.467684472252689),
np.float64(25.466988086369042),
np.float64(8.5837012776003),
np.float64(8.986459083858149),
np.float64(5.291336480708372),
np.float64(4.588400140296019),
np.float64(9.609788112673694),
np.float64(18.619010582268082),
np.float64(4.588400140296019),
np.float64(6.7802046790521775),
np.float64(4.588400140296019),
np.float64(7.723295717641557),
np.float64(12.211652546820153),
np.float64(4.588400140296019),
np.float64(4.097376608232803),
np.float64(4.802617385043728),
np.float64(10.678188875125839),
np.float64(8.539696461963874),
np.float64(7.8549600534128965),
np.float64(17.4775784966772),
np.float64(8.053651597923494),
np.float64(28.4660018179915),
np.float64(8.539696461963874),
np.float64(8.427714583774561),
np.float64(25.39459953448461),
np.float64(8.539696461963874),
np.float64(8.967475730663207),
np.float64(5.253645817865142),
np.float64(4.5323493525252),
np.float64(9.535995272407447),
np.float64(18.53163209680442),
np.float64(4.5323493525252),
np.float64(6.762468786035861),
np.float64(4.5323493525252),
np.float64(7.629235787264522),
np.float64(12.166138982562426),
np.float64(4.5323493525252),
np.float64(4.0291392081815856),
np.float64(4.636693772031066),
np.float64(10.434281511314525),
np.float64(8.455808527927934),
np.float64(7.682370708801383),
np.float64(17.605050390162607),
```

```
np.float64(7.759480913347307),
np.float64(28.337156484396264),
np.float64(8.455808527927934),
np.float64(8.365010017319705),
np.float64(25.16169698514667),
np.float64(8.455808527927934),
np.float64(9.014950474445056),
np.float64(5.202907185993475),
np.float64(4.385220064783706),
np.float64(9.295720209064962),
np.float64(18.22003239617554),
np.float64(4.385220064783706),
np.float64(6.816492695755526),
np.float64(4.385220064783706),
np.float64(7.2825585008404135),
np.float64(12.074330118264655),
np.float64(4.385220064783706),
np.float64(3.818030205967989),
np.float64(4.799981224265389),
np.float64(10.493590629150642),
np.float64(8.728476885464556),
np.float64(7.836770518903913),
np.float64(18.159531851061008),
np.float64(7.751772270163592),
np.float64(28.549881642520486),
np.float64(8.728476885464556),
np.float64(8.665922864964433),
np.float64(25.235679188947287),
np.float64(8.728476885464556),
np.float64(9.462769069072063),
np.float64(5.519774613082459),
np.float64(4.573566617379585),
np.float64(9.359872394192855),
np.float64(18.18908506492156),
np.float64(4.573566617379585),
np.float64(7.273043511632948),
np.float64(4.573566617379585),
np.float64(7.2048410551928015),
np.float64(12.336437235452163),
np.float64(4.573566617379585),
np.float64(3.92107047260107),
np.float64(5.622067401390593),
np.float64(11.185703888278095),
np.float64(9.687289194217636),
np.float64(8.64774714336437),
np.float64(19.470610539016327),
np.float64(8.360113328016249),
```

```
np.float64(29.43376495200806),
np.float64(9.687289194217636),
np.float64(9.660040786352644),
np.float64(25.946133805530355),
np.float64(9.687289194217636),
np.float64(10.640519174188123),
np.float64(6.53383575877599),
np.float64(5.4269766699567334),
np.float64(18.768377762686402),
np.float64(5.4269766699567334),
np.float64(8.461708893312027),
np.float64(5.4269766699567334),
np.float64(7.725671109965597),
np.float64(13.282047993768861),
np.float64(5.4269766699567334),
np.float64(4.667847667724722),
np.float64(5.238004164754565),
np.float64(5.034432600240457),
np.float64(4.873433083501801),
np.float64(4.881021709668385),
np.float64(5.78081549105248),
np.float64(5.966964652207911),
np.float64(5.714410804844755),
np.float64(5.491070200843672),
np.float64(5.373976652485404),
np.float64(6.077841302473303),
np.float64(4.07252402850521),
np.float64(3.9679338595717386),
np.float64(3.93291066448116),
np.float64(4.192451933943904),
np.float64(5.488171297650535),
np.float64(4.13761734252524),
np.float64(4.016945891001239),
np.float64(3.9614556089772575),
np.float64(4.180062704573189),
np.float64(5.411456937917693),
np.float64(3.3685202115821085),
np.float64(3.171318338285799),
np.float64(3.0184257012788813),
np.float64(3.0422280869089424),
np.float64(3.967500633164218),
np.float64(4.755973541086965),
np.float64(4.491652041555561),
np.float64(4.253334425703983),
np.float64(4.106286853644719),
np.float64(4.763080894959623),
np.float64(4.631206247962443),
```

```
np.float64(4.500504613471573),
np.float64(4.43224864403522),
np.float64(4.625324364806399),
np.float64(5.816597866283439),
np.float64(5.1117990728632),
np.float64(4.973449852961908),
np.float64(4.895460593003194),
np.float64(5.069069732729662),
np.float64(6.229752892565009),
np.float64(34.48661645726665),
np.float64(34.54480847525951),
np.float64(34.716962608984275),
np.float64(35.39085853607769),
np.float64(37.337706647489654),
np.float64(5.060024491532967),
np.float64(4.828562088372132),
np.float64(4.632065140628546),
np.float64(4.56865890478527),
np.float64(5.3568893315824475),
np.float64(4.928809296347976),
np.float64(4.698331924355351),
np.float64(4.503088652644035),
np.float64(4.442189768865295),
np.float64(5.234360320335315),
np.float64(5.178204098263351),
np.float64(4.907226744993127),
np.float64(4.6604380425648655),
np.float64(4.496448297352229),
np.float64(5.1266189237118445),
np.float64(2.8967710593703484),
np.float64(2.6629531097255774),
np.float64(2.4634581937297106),
np.float64(2.3940560213818713),
np.float64(3.172864262243303),
np.float64(4.54839790850138),
np.float64(4.3462349253382),
np.float64(4.187028148500718),
np.float64(4.198202254469651),
np.float64(5.103630361257442),
np.float64(3.705235475978413),
np.float64(3.621668509796653),
np.float64(3.613402118208258),
np.float64(3.9264569946753634),
np.float64(5.30626916938885),
np.float64(3.8801471052677123),
np.float64(3.788434125783942),
np.float64(3.7698000809020775),
```

```
np.float64(4.06211965078224),
np.float64(5.409347772287673),
np.float64(3.8876544595713747),
np.float64(3.779751394855258),
np.float64(3.740511786950406),
np.float64(3.9916202307845987),
np.float64(5.274088011360652),
np.float64(2.830684231327378),
np.float64(2.656765287479629),
np.float64(2.5335054697708816),
np.float64(2.6165734939972807),
np.float64(3.6349777580468037),
np.float64(3.581649513823282),
np.float64(3.342745974321382),
np.float64(3.1367784894164443),
np.float64(3.0544311792504626),
np.float64(3.812897060683384),
np.float64(3.7885595328600035),
np.float64(3.5474956764075185),
np.float64(3.3387786972018336),
np.float64(3.250932398434359),
np.float64(4.000757012064936),
np.float64(4.524528685115667),
np.float64(4.404650078760781),
np.float64(4.35016887240659),
np.float64(4.570794119342095),
np.float64(5.8053597333630735),
np.float64(3.064988601613169),
np.float64(2.9288570650472416),
np.float64(2.8536903111517233),
np.float64(3.0329444630045774),
np.float64(4.202498356181387),
np.float64(4.551066676566054),
np.float64(4.456347003985528),
np.float64(4.433886258798701),
np.float64(4.7185524280689375),
np.float64(6.053753777187345),
np.float64(2.626348759858164),
np.float64(2.49210200307126),
np.float64(2.4193340598035866),
np.float64(2.6033858329121347),
np.float64(3.7804788452050344),
np.float64(2.8387694164723922),
np.float64(2.7273738816166317),
np.float64(2.6836893117158707),
np.float64(2.9259078315582436),
np.float64(4.194405731575722),
```

```
np.float64(2.732126106911891),
np.float64(2.5836654619514188),
np.float64(2.4928071155537492),
np.float64(2.640678082402309),
np.float64(3.760915542000936),
np.float64(4.0658472747895775),
np.float64(4.021418940924743),
np.float64(4.062965354103344),
np.float64(4.475645840104432),
np.float64(6.012012544085618),
np.float64(3.6841269456102994),
np.float64(3.5825622840945326),
np.float64(3.551389734808308),
np.float64(3.8186322958797514),
np.float64(5.1264536892571995),
np.float64(4.5643256761353594),
np.float64(4.490831426880027),
np.float64(4.495384856834357),
np.float64(4.834079376386907),
np.float64(6.254182418806097),
np.float64(5.538919477462997),
np.float64(5.480702426440792),
np.float64(5.504699563237279),
np.float64(5.882281496474154),
np.float64(7.363493331825844),
np.float64(2.929518511903414),
np.float64(2.8052651662108894),
np.float64(2.7452161097906065),
np.float64(2.9547056565939354),
np.float64(4.171772313264348),
np.float64(2.7168180917593907),
np.float64(2.519372607998482),
np.float64(2.3661699213093494),
np.float64(2.389352207574976),
np.float64(3.313650311971856),
np.float64(5.318433651203116),
np.float64(5.257931984557376),
np.float64(5.2790214287421),
np.float64(5.650787976755436),
np.float64(7.1228613496130055),
np.float64(5.2159491383396634),
np.float64(5.148066335187264),
np.float64(5.159761605636235),
np.float64(5.512739806178067),
np.float64(6.955288633008989),
np.float64(4.284897549639947),
np.float64(4.125602161394508),
```

```
np.float64(4.020954141725063),
np.float64(4.141245762030065),
np.float64(5.218144248488821),
np.float64(5.723556895232696),
np.float64(5.709215256622743),
np.float64(5.7890538273984635),
np.float64(6.278318628593785),
np.float64(7.9350321135944855),
np.float64(4.111234308741533),
np.float64(4.016073886760865),
np.float64(3.993052187792044),
np.float64(4.276596449498289),
np.float64(5.610034801016142),
np.float64(4.367037009833367),
np.float64(4.242561312364573),
np.float64(4.182229262774498),
np.float64(4.391152823238242),
np.float64(5.607330072803587),
np.float64(10.063069763947),
np.float64(9.642408392764379),
np.float64(9.205112758447605),
np.float64(8.660109149457957),
np.float64(8.691543704167978),
np.float64(15.877426618832764),
np.float64(26.476316532557227),
np.float64(9.497990756060492),
np.float64(12.683683692207769),
np.float64(6.353462816327349),
np.float64(26.656094337863866),
np.float64(11.053474903184497),
np.float64(6.951235546250237),
np.float64(9.150408421890036),
np.float64(7.91719390518389),
np.float64(16.339285634090015),
np.float64(15.309140082307342),
np.float64(14.25404936876882),
np.float64(10.640260465074789),
np.float64(10.83953787292571),
np.float64(7.153645316889812),
np.float64(11.110251564159547),
np.float64(26.540795716562418),
np.float64(7.666658332777368),
np.float64(8.307386108792302),
np.float64(2.25100993593487),
np.float64(8.23497560740445),
np.float64(15.075705791539566),
np.float64(7.540214879805121),
```

```
np.float64(10.722457828962998),
np.float64(10.85054900470569),
np.float64(23.960357446198522),
np.float64(11.324588240393409),
np.float64(3.773999928684869),
np.float64(4.788647299341674),
np.float64(3.6968589529270237),
np.float64(9.081010245856346),
np.float64(5.621093231471439),
np.float64(4.8477873235844156),
np.float64(4.325084460855209),
np.float64(4.534493350770903),
np.float64(5.277566230264282),
np.float64(4.810186061441089),
np.float64(4.1430985278224775),
np.float64(6.360085660630392),
np.float64(11.30128939145327),
np.float64(3.6766009576168113),
np.float64(4.682307836996053),
np.float64(3.5970503143766943),
np.float64(9.085863613203966),
np.float64(5.579912727396723),
np.float64(4.797545605257302),
np.float64(4.237027146858216),
np.float64(4.382883341495947),
np.float64(5.056253483151001),
np.float64(4.721966594432),
np.float64(4.017802225940195),
np.float64(6.1137028232339645),
np.float64(11.369727876354219),
np.float64(3.6507301961731264),
np.float64(4.64505826847183),
np.float64(3.568112703410116),
np.float64(9.190132191562045),
np.float64(5.625592742126378),
np.float64(4.831693165483902),
np.float64(4.22304576714134),
np.float64(4.288016713243485),
np.float64(4.8726742794679385),
np.float64(4.707778838154277),
np.float64(3.9564257707329507),
np.float64(5.898216231917813),
np.float64(11.8361925058),
np.float64(3.9285763329296484),
np.float64(4.900146791067279),
np.float64(3.8398251411208535),
np.float64(9.728257007922101),
```

```
np.float64(6.046540431229583),
np.float64(5.229575945580999),
np.float64(4.52467066735148),
np.float64(4.427871116382449),
np.float64(4.8351035317457125),
np.float64(5.008990985242722),
np.float64(4.16326051996236),
np.float64(5.796830708929392),
np.float64(13.457077149479973),
np.float64(5.253060488097935),
np.float64(6.188868981125306),
np.float64(5.154670626360047),
np.float64(11.461750516753087),
np.float64(7.595898454371235),
np.float64(6.742689111713047),
np.float64(5.886521450804021),
np.float64(5.535511118723151),
np.float64(5.66393258273311),
np.float64(6.370193156646886),
np.float64(5.376155351873748),
np.float64(6.525379398784199),
np.float64(4.810085534816314),
np.float64(4.810085534816314),
np.float64(4.810085534816314),
np.float64(5.094040300932321),
np.float64(5.141010260013794),
np.float64(5.141010260013794),
np.float64(5.141010260013794),
np.float64(5.064520328293346),
np.float64(5.064520328293346),
np.float64(5.064520328293346),
np.float64(4.8553580797648115),
np.float64(4.8553580797648115),
np.float64(4.8553580797648115),
np.float64(5.1512273211840505),
np.float64(5.582753229197076),
np.float64(5.46389535904228),
np.float64(5.296201119821255),
np.float64(5.657330833236496),
np.float64(5.535199755580812),
np.float64(5.348684573229676),
np.float64(5.621856440901341),
np.float64(5.500249076445798),
np.float64(5.316745244995479),
np.float64(5.915134549842602),
np.float64(5.7999426720888),
np.float64(5.653327889173495),
```

```
np.float64(5.240779619064789),
np.float64(5.362014678684475),
np.float64(29.49241920972562),
np.float64(18.926539102613127),
np.float64(29.94260826117978),
np.float64(29.84703603725826),
np.float64(29.404115709266623),
np.float64(19.39176476986166),
np.float64(19.293786687413057),
np.float64(18.821394592464433),
np.float64(11.069389335999766),
np.float64(9.30934622684019),
np.float64(14.318207776785524),
np.float64(11.307055916040131),
np.float64(11.137019498863822),
np.float64(14.572318000376214),
np.float64(14.508118388912857),
np.float64(14.449512563533625),
np.float64(7.3933675403532835),
np.float64(7.854075496269496),
np.float64(7.756820247634052),
np.float64(7.293282866896797),
np.float64(10.513917603818976),
np.float64(10.836640176158147),
np.float64(10.761462588895029),
np.float64(10.568376559968781),
np.float64(6.065626141361638),
np.float64(6.065626141361638),
np.float64(6.054398591509465),
np.float64(6.054398591509465),
np.float64(6.032653023796965),
np.float64(6.032653023796965),
np.float64(6.494109234365744),
np.float64(6.494109234365744),
np.float64(13.152106780591739),
np.float64(6.620797110216149),
np.float64(7.294778823196542),
np.float64(5.550803634608778),
np.float64(4.568444155023238),
np.float64(4.834110924048981),
np.float64(8.316003673236784),
np.float64(8.946620168788385),
np.float64(7.8964098160703875),
np.float64(9.094763758046843),
np.float64(5.627363521729302),
np.float64(9.592659645693075),
np.float64(9.992773717591456),
```

```
np.float64(7.170744301937902),
np.float64(9.526639413572529),
np.float64(4.954269949842694),
np.float64(5.496126574538595),
np.float64(6.66670041338535),
np.float64(4.7967290447264554),
np.float64(5.412590825230359),
np.float64(10.109587365605593),
np.float64(7.235663011137345),
np.float64(18.209867426182505),
np.float64(3.7048960939594564),
np.float64(8.65297226315795),
np.float64(9.945705271393681),
np.float64(3.9411455038156187),
np.float64(4.462028849165455),
np.float64(18.13149996817265),
np.float64(3.4091880223863273),
np.float64(3.9169329531287627),
np.float64(4.29228774348012),
np.float64(12.565122998013424),
np.float64(7.003734177172993),
np.float64(4.9843904086741855),
np.float64(3.618334786886666),
np.float64(4.435978481810365),
np.float64(4.464432134905154),
np.float64(10.080468859794403),
np.float64(4.44142671737081),
np.float64(15.623334513313507),
np.float64(3.760586433287195),
np.float64(6.731454116217005),
np.float64(7.426512790417312),
np.float64(6.525160177613937),
np.float64(4.267618497049432),
np.float64(7.167556502282594),
np.float64(10.111573447863181),
np.float64(4.087199903240483),
np.float64(5.272328914252863),
np.float64(8.840997432288585),
np.float64(8.640335789414708),
np.float64(12.431870188294733),
np.float64(15.723613490433232),
np.float64(4.978351364847067),
np.float64(10.48405894328637),
np.float64(3.3677771440577575),
np.float64(4.722234114499716),
np.float64(3.722328409962026),
np.float64(7.066963495396607),
```

```
np.float64(5.88099904515826),
np.float64(3.745332743955234),
np.float64(6.423459537489353),
np.float64(7.004342167998628),
np.float64(4.342361001923846),
np.float64(9.095746028896265),
np.float64(4.153578219849622),
np.float64(5.111666167183107),
np.float64(4.960328797347111),
np.float64(4.717068544515564),
np.float64(4.312944953763446),
np.float64(3.843847497645229),
np.float64(6.996878475371972),
np.float64(27.736418499074762),
np.float64(6.7585527203197024),
np.float64(6.419422068974524),
np.float64(6.085892805359961),
np.float64(5.92942736942236),
np.float64(9.64649318441534),
np.float64(31.363379723909908),
np.float64(5.220367396944453),
np.float64(4.889320353479961),
np.float64(5.557547732457432),
np.float64(4.743847606778381),
np.float64(8.366656388497017),
np.float64(29.232988457625765),
np.float64(6.029621142141187),
np.float64(5.0195448090147465),
np.float64(8.439205789925737),
np.float64(29.726818597951702),
np.float64(7.983262608954024),
np.float64(29.12612830230222),
np.float64(5.712165741158483),
np.float64(4.967947996609818),
np.float64(4.634019143291682),
np.float64(6.69599244027169),
np.float64(8.228172928328691),
np.float64(7.789039564591304),
np.float64(6.890776868955752),
np.float64(6.43932521574285),
np.float64(3.892776123361037),
np.float64(3.3911825555538),
np.float64(4.890372737343284),
np.float64(4.411324853722532),
np.float64(4.244990463269837),
np.float64(3.7506646387167324),
np.float64(3.81856864760553),
```

```
np.float64(3.289378849390436),
np.float64(3.6156871709621208),
np.float64(4.160154910109563),
np.float64(5.760616478508168),
np.float64(5.293886884362925),
np.float64(3.681294882613689),
np.float64(3.1593728276527244),
np.float64(10.11438025580982),
np.float64(2.8882332998139697),
np.float64(6.737384549096771),
np.float64(5.001160914836221),
np.float64(6.543686311786254),
np.float64(5.507708283484871),
np.float64(4.233322380278488),
np.float64(11.230484176968204),
np.float64(11.517316246520584),
np.float64(5.378554778705058),
np.float64(6.00458108397911),
np.float64(3.597796591659687),
np.float64(6.714419797258977),
np.float64(6.07197442679249),
np.float64(6.870412008096463),
np.float64(8.621030259637562),
np.float64(5.4454019912479),
np.float64(6.120706235926309),
np.float64(6.310947407535629),
np.float64(3.560025949034253),
np.float64(5.042299868337858),
np.float64(9.02224017280859),
np.float64(8.699572087093841),
np.float64(9.829024064562716),
np.float64(5.1642549929529835),
np.float64(5.551971724267735),
np.float64(3.224342960547564),
np.float64(5.145471286491651),
np.float64(3.2631450262711104),
np.float64(3.330882861717695),
np.float64(3.471724893357039),
np.float64(6.433887499648262),
np.float64(11.15062500203291),
np.float64(4.696745776997534),
np.float64(2.7297759896894163),
np.float64(10.878501681677928),
np.float64(4.6477872710857575),
np.float64(3.580067023932512),
np.float64(6.904824229312971),
np.float64(9.411086387683815),
```

```
np.float64(6.598291576348221),
np.float64(6.199048722141074),
np.float64(3.037409135662385),
np.float64(7.2610887256158545),
np.float64(6.4003273189444565),
np.float64(5.065911542711263),
np.float64(3.392765290011493),
np.float64(2.597293678053216),
np.float64(9.400367006700089),
np.float64(2.812145827459688),
np.float64(8.019979047938081),
np.float64(4.105259547574283),
np.float64(7.150079721389905),
np.float64(9.725315724180968),
np.float64(10.216513947178237),
np.float64(6.681049960513536),
np.float64(5.049054074590567),
np.float64(3.492619878900271),
np.float64(3.087579484968651),
np.float64(3.689932540533786),
np.float64(5.9360457659561625),
np.float64(9.849028656363318),
np.float64(3.3339722135374306),
np.float64(8.387188701753828),
np.float64(4.9252620660416815),
np.float64(3.0251536917170148),
np.float64(5.4589519227043946),
np.float64(8.947071303405377),
np.float64(6.4816146391489236),
np.float64(9.736896959615848),
np.float64(5.1318182951305396),
np.float64(8.30489318145446),
np.float64(5.558045500670185),
np.float64(5.8233052323071375),
np.float64(6.179331272179935),
np.float64(10.018875284310312),
np.float64(4.048110086174444),
np.float64(3.9220426998843623),
np.float64(3.4073262699167475),
np.float64(3.3299135587602136),
np.float64(4.065977853645288),
np.float64(4.603122372673267),
np.float64(4.4805577672279755),
np.float64(3.984230936695495),
np.float64(3.961111328633167),
np.float64(4.723446479854145),
np.float64(7.122489479935294),
```

```
np.float64(7.004963133880125),
np.float64(6.535087165145784),
np.float64(6.590060577630338),
np.float64(7.390182674277228),
np.float64(7.073908795212914),
np.float64(6.530073473107029),
np.float64(4.96137373332948),
np.float64(4.415168063306701),
np.float64(12.095748526574939),
np.float64(7.634457187084256),
np.float64(7.266034238181567),
np.float64(3.8959572901870767),
np.float64(5.462873554001974),
np.float64(4.967093604375157),
np.float64(15.81281212220372),
np.float64(15.732273178356001),
np.float64(15.577921569428792),
np.float64(15.329386398187639),
np.float64(14.914712970616282),
np.float64(14.98489421299596),
np.float64(14.796879597335455),
np.float64(14.503247280925413),
np.float64(14.523468325464972),
np.float64(18.386800352533644),
np.float64(26.790370548166145),
np.float64(4.230194124670037),
np.float64(3.9789289055861294),
np.float64(3.5587953823292904),
np.float64(3.0481159954595074),
np.float64(3.018796796547325),
np.float64(6.068906779600877),
np.float64(7.5378963624085475),
np.float64(7.465507228740785),
np.float64(7.327455240173499),
np.float64(7.107444404562194),
np.float64(6.749819648250554),
np.float64(6.775422790695317),
np.float64(27.675251794193123),
np.float64(5.928417858844247),
np.float64(3.9333754825261678),
np.float64(4.283821639121958),
np.float64(3.733414571531884),
np.float64(11.678549591910269),
np.float64(4.923230147715572),
np.float64(4.367098693253248),
np.float64(3.824326917945365),
np.float64(5.534103815110189),
```

```
np.float64(4.971488782698784),
np.float64(6.573408777314804),
np.float64(4.951614793759357),
np.float64(4.362299146026612),
np.float64(4.0646411252777925),
np.float64(14.063937893294016),
np.float64(3.803661502520236),
np.float64(3.2698004555443982),
np.float64(3.1358655417921595),
np.float64(5.8080938896087915),
np.float64(5.323530399177959),
np.float64(5.335140652368109),
np.float64(9.185973143606988),
np.float64(30.508352631776233),
np.float64(4.465272803611572),
np.float64(6.787722511263832),
np.float64(16.99293913397842),
np.float64(14.398894808584325),
np.float64(17.484105406758854),
np.float64(25.109554173497212),
np.float64(15.104662647305648),
np.float64(6.50096298852073),
np.float64(5.960294593256693),
np.float64(3.980344173998865),
np.float64(7.920910010997701),
np.float64(7.1380889346736405),
np.float64(6.708423818552528),
np.float64(4.42798285905336),
np.float64(3.8826263376846653),
np.float64(6.4368524255388655),
np.float64(5.960294593256693),
np.float64(4.296478319593059),
np.float64(3.780933056393949),
np.float64(3.7010733137920506),
np.float64(7.419126778881184),
np.float64(28.47594821475094),
np.float64(14.534758208843458),
np.float64(14.39636928232827),
np.float64(14.175768805308154),
np.float64(13.81696476617889),
np.float64(13.548568181533321),
np.float64(24.97754825498734),
np.float64(4.137719631154814),
np.float64(3.729686827516322),
np.float64(6.371177466358524),
np.float64(4.064745338613318),
np.float64(4.050870213171266),
```

```
np.float64(8.39258384114282),
np.float64(26.042375489408258),
np.float64(26.018671642341122),
np.float64(5.702378022736233),
np.float64(10.609370115743022),
np.float64(12.10955704467248),
np.float64(6.143721534273743),
np.float64(10.520110741416808),
np.float64(13.938325235695677),
np.float64(11.253774294222868),
np.float64(5.423796023426847),
np.float64(6.080518552372943),
np.float64(6.958868220462346),
np.float64(7.889871495086196),
np.float64(3.223306560029222),
np.float64(5.644167441761135),
np.float64(8.15565728504288),
np.float64(5.200878329514838),
np.float64(5.972682572697287),
np.float64(3.1337800682723698),
np.float64(5.534318382097016),
np.float64(6.592808506932045),
np.float64(3.751514112082739),
np.float64(9.245921967895779),
np.float64(10.870551562584554),
np.float64(9.105374156339376),
np.float64(4.072740502221691),
np.float64(7.07330338498844),
np.float64(2.93996350284321),
np.float64(3.2962175928453856),
np.float64(9.060564320266586),
np.float64(13.475634940820752),
np.float64(2.6653536107372604),
np.float64(9.318462662333376),
np.float64(3.857124942760097),
np.float64(4.906493520088452),
np.float64(15.543413901464618),
np.float64(2.5014921248670543),
np.float64(9.233002517700871),
np.float64(4.257536378061129),
np.float64(13.55121470447357),
np.float64(9.358489527891038),
np.float64(2.793996917789708),
np.float64(7.023847017834657),
np.float64(3.507965692772611),
np.float64(7.76551158229342),
np.float64(3.567766786771131),
```

```
np.float64(19.32104601605702),
np.float64(5.306293841656575),
np.float64(11.548122175549626),
np.float64(9.969573738236408),
np.float64(14.513585931976733),
np.float64(4.960600982900667),
np.float64(3.3350311101610006),
np.float64(5.140977358788124),
np.float64(5.3970117368703585),
np.float64(3.7380779042819654),
np.float64(10.482281523489553),
np.float64(8.96393329455772),
np.float64(4.879103441083147),
np.float64(11.002195078351185),
np.float64(9.07679110346247),
np.float64(6.255296747005956),
np.float64(2.4139458600964034),
np.float64(16.320176051175974),
np.float64(18.004484979614748),
np.float64(6.652514670411847),
np.float64(2.3601348510167206),
np.float64(7.459426732705263),
np.float64(2.0947522960763356),
np.float64(2.709381494095339),
np.float64(3.112673362593604),
np.float64(3.4520803991614715),
np.float64(6.594894088041703),
np.float64(15.23819788965607),
np.float64(7.8082443909423205),
np.float64(5.60205288950269),
np.float64(10.085363820043604),
np.float64(7.952282067707887),
np.float64(6.22785474468459),
np.float64(2.1318025052768705),
np.float64(7.376739721616621),
np.float64(18.059987650813408),
np.float64(4.512654608875315),
np.float64(3.4057300469477094),
np.float64(10.784116760540444),
np.float64(3.0894052444914286),
np.float64(4.801058808817512),
np.float64(4.172968865812449),
np.float64(6.670052026326346),
np.float64(1.9178883557567343),
np.float64(6.595816997759878),
np.float64(11.205736848268735),
np.float64(5.687325298702032),
```

```
np.float64(4.980522253943774),
np.float64(7.0625766970907735),
np.float64(1.3073244303397216),
np.float64(14.98163455416627),
np.float64(6.145419076840172),
np.float64(6.494942636876296),
np.float64(11.74479502163196),
np.float64(10.448942933783087),
np.float64(11.58780861861979),
np.float64(9.854534786711106),
np.float64(11.155009496871566),
np.float64(9.770088987306401),
np.float64(17.267563971398616),
np.float64(14.782256081842208),
np.float64(11.17858203831194),
np.float64(15.592232908372756),
np.float64(19.276344335838882),
np.float64(5.175501196102253),
np.float64(11.717847053958591),
np.float64(10.035413930759185),
np.float64(14.122497077185804),
np.float64(4.95676521946863),
np.float64(3.2580136733201748),
np.float64(5.108490475082922),
np.float64(5.285344524718898),
np.float64(3.8988795610457134),
np.float64(10.185400594258367),
np.float64(8.728977209734644),
np.float64(4.770798878789279),
np.float64(11.107278676459991),
np.float64(9.021934102971922),
np.float64(6.285815153815096),
np.float64(2.398486172007965),
np.float64(16.18681850741552),
np.float64(17.681158668326617),
np.float64(6.218640417550032),
np.float64(2.3699372841352746),
np.float64(7.440836060480269),
np.float64(2.2161668429917585),
np.float64(2.6665216440718047),
np.float64(2.9651122215050845),
np.float64(3.4157284948070115),
np.float64(6.746827293641126),
np.float64(15.192685588347075),
np.float64(7.622957949764386),
np.float64(5.587136690968614),
np.float64(10.216561550036179),
```

```
 np.float64(8.087559284751244),
 np.float64(6.246699858121725),
 np.float64(2.1575815042588466),
 np.float64(7.3910904324721995),
 np.float64(17.849966601469017),
 np.float64(4.515946927834682),
 np.float64(3.1854205455014046),
 np.float64(10.787490341248166),
 np.float64(3.151940573346622),
 np.float64(4.816756125727998),
 np.float64(4.077578140205367),
 np.float64(6.6970453618927674),
 np.float64(1.7804719425226772),
 np.float64(6.391913221512314),
 np.float64(11.117212496416808),
 np.float64(5.697253882742145),
 np.float64(4.994584009298884),
 np.float64(7.128499942505703),
 np.float64(1.2502060699520425),
 np.float64(5.636872240939222),
 np.float64(10.405875758539562),
 np.float64(12.294045398154509),
 np.float64(6.019066541464972),
 np.float64(10.948770617168295),
 np.float64(14.076961376992049),
 np.float64(11.098650400025486),
 np.float64(5.424499977676839),
 np.float64(5.812322366024701),
 np.float64(6.9972749789679085),
 np.float64(8.135472784651219),
 np.float64(3.2377866162929236),
 np.float64(5.502448734323372),
 np.float64(8.23595644975953),
 …]
```

[16]:
```
data['Mahalanobis'] = valoresmaha
#data ['Indices'] = data.index
data
```

[16]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	\
0	540.0	0.0	0.0	162.0	2.5	
1	540.0	0.0	0.0	162.0	2.5	
2	332.5	142.5	0.0	228.0	0.0	
3	332.5	142.5	0.0	228.0	0.0	
4	198.6	132.4	0.0	192.0	0.0	
…	…	…	…	…	…	
1025	276.4	116.0	90.3	179.6	8.9	

```
1026   322.2                0.0    115.6  196.0              10.4
1027   148.5              139.4    108.6  192.7               6.1
1028   159.1              186.7      0.0  175.6              11.3
1029   260.9              100.5     78.3  200.6               8.6

      Coarse Aggregate  Fine Aggregate  Age  Mahalanobis
0               1040.0           676.0   28    13.070994
1               1055.0           676.0   28    12.271675
2                932.0           594.0  270    16.505195
3                932.0           594.0  365    28.549966
4                978.4           825.5  360    30.457394
…                    …               …    …            …
1025             870.1           768.3   28     2.868944
1026             817.9           813.4   28     7.121385
1027             892.4           780.0   28     3.486395
1028             989.6           788.9   28     8.086373
1029             864.5           761.5   28     3.321240

[1030 rows x 9 columns]
```

[17]:
```
data = data[data["Mahalanobis"] > 15]
data = data.drop(['Mahalanobis'], axis =1)
data
```

[17]:
```
      Cement  Blast Furnace Slag  Fly Ash  Water  Superplasticizer  \
2      332.5               142.5      0.0  228.0               0.0
3      332.5               142.5      0.0  228.0               0.0
4      198.6               132.4      0.0  192.0               0.0
6      380.0                95.0      0.0  228.0               0.0
12     427.5                47.5      0.0  228.0               0.0
…          …                   …        …      …                 …
953    158.4                 0.0    194.9  219.7              11.0
954    150.7                 0.0    185.3  166.7              15.6
963    150.0               236.8      0.0  173.8              11.9
971    312.7               144.7      0.0  127.3               8.0
1019   139.7               163.9    127.7  236.7               5.8

      Coarse Aggregate  Fine Aggregate  Age
2                932.0           594.0  270
3                932.0           594.0  365
4                978.4           825.5  360
6                932.0           594.0  365
12               932.0           594.0  270
…                    …               …    …
953              897.7           712.9   28
954             1074.5           678.0   28
963             1069.3           674.8   28
```

```
971              999.7           822.2   28
1019             868.6           655.6   28

[105 rows x 8 columns]
```

[18]:
```
# contabilizamos la cantidad nuevas filas y columnas
columnasfinales, filasf = data.shape

print(f"Se eleiminaron {columnasiniciales-columnasfinales} columnas ")
```

```
Se eleiminaron 925 columnas
```

2.2 Seccion B:

Realice un análisis de componentes principales para la nueva data, pero estandarizadas y por medio del gráfico de codo tomando un porcentaje de varianza explicada de al menos 80%, indique la cantidad de componentes que quedarían. Además, actualice los nuevos datos, pero con las componentes elegidas como nuevas variables.

[19]:
```
# Entrenando y estandarizando los datos
pca_pipe = make_pipeline(StandardScaler(),PCA())
pca_pipe.fit(data)
modelo_pca = pca_pipe.named_steps['pca']
```

[20]:
```
# convirtiendo el array a dataframe
pd.DataFrame(data = modelo_pca.components_,columns = data.columns,index =␣
 ↪['CP1','CP2','CP3','CP4','CP5','CP6','CP7','CP8'])
```

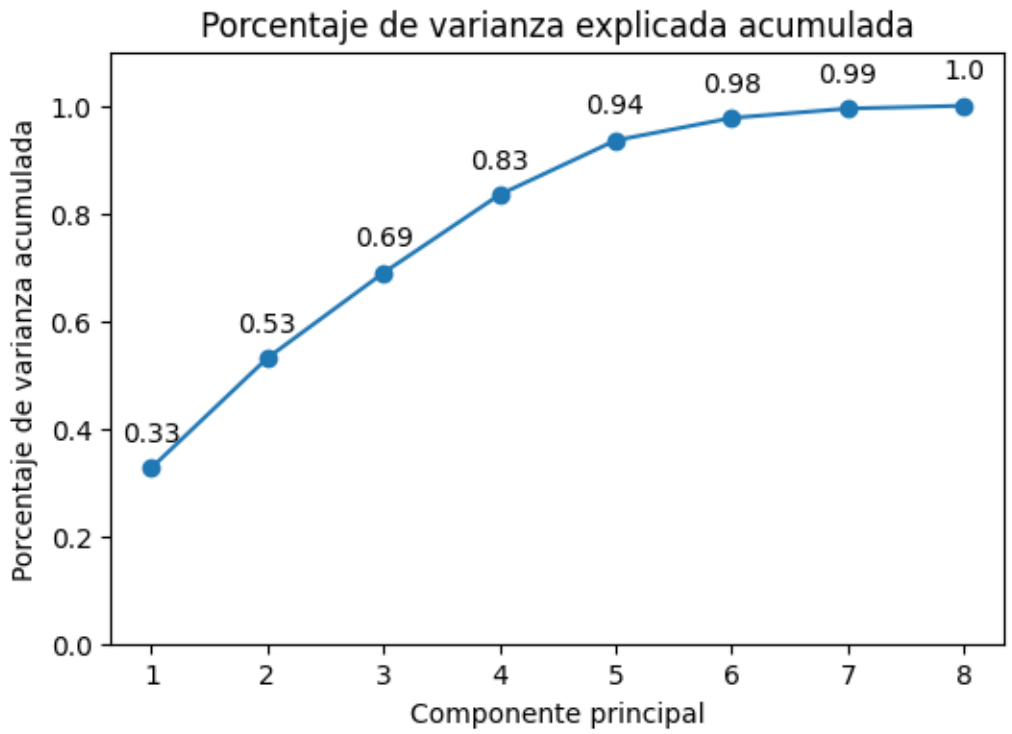
[20]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	\
CP1	0.193308	-0.096110	0.022758	-0.515243	0.526835	
CP2	0.645563	-0.509217	-0.466747	-0.049653	-0.098640	
CP3	0.087718	-0.270653	0.510151	-0.138004	-0.089472	
CP4	-0.141130	0.522511	-0.470316	-0.358022	0.073832	
CP5	-0.441202	-0.377618	0.187598	-0.303709	-0.185539	
CP6	-0.119555	-0.160150	0.092893	0.125208	0.795118	
CP7	-0.368595	-0.306638	-0.377563	0.546716	0.142929	
CP8	0.416478	0.350598	0.337345	0.421923	0.110421	

	Coarse Aggregate	Fine Aggregate	Age
CP1	-0.199641	0.472909	-0.382554
CP2	0.235040	-0.033130	0.193794
CP3	0.485310	-0.371987	-0.507788
CP4	0.582360	-0.093418	-0.066078
CP5	0.341010	0.455117	0.421272
CP6	0.153150	-0.356323	0.391304
CP7	0.187132	0.251109	-0.458427
CP8	0.394298	0.480630	0.114617

[21]:
```
# Porcentaje de varianza explicada acumulada
por_var_acum = modelo_pca.explained_variance_ratio_.cumsum()
print('Porcentaje de varianza explicada acumulada')
print(por_var_acum)

fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(6,4))
ax.plot(np.arange(len(data.columns)) + 1,por_var_acum, marker='o')

for x, y in zip(np.arange(len(data.columns)) + 1, por_var_acum):
    label = round(y,2)
    ax.annotate(label,(x,y),textcoords="offset␣
 ↪points",xytext=(0,10),ha='center')

ax.set_ylim(0,1.1)
ax.set_xticks(np.arange(modelo_pca.n_components_) + 1)
ax.set_title('Porcentaje de varianza explicada acumulada')
ax.set_xlabel('Componente principal')
ax.set_ylabel('Porcentaje de varianza acumulada');
```

```
Porcentaje de varianza explicada acumulada
[0.32671143 0.53098874 0.68929234 0.83452576 0.93556425 0.97728099
 0.99476577 1.        ]
```



Del analisis graficamos, podemos interpretar que solo tomamos 3 componestes pues estos estan en

menos del 80%

[22]:
```
# actualizando los datos con los componentes encontrados

# Realizamos el PCA Con los 3 componentes

pca_pipe_final = make_pipeline(StandardScaler(),PCA(n_components = 3))
pca_pipe_final.fit(data)
modelo_pca_final = pca_pipe_final.named_steps['pca']
```

[24]:
```
# convirtiendo el array a dataframe
data2 = pd.DataFrame(data = modelo_pca_final.components_,columns = data.
 ↪columns,index = ['CP1','CP2','CP3'])
data2
```

[24]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	\
CP1	0.193308	-0.096110	0.022758	-0.515243	0.526835	
CP2	0.645563	-0.509217	-0.466747	-0.049653	-0.098640	
CP3	0.087718	-0.270653	0.510151	-0.138004	-0.089472	

	Coarse Aggregate	Fine Aggregate	Age
CP1	-0.199641	0.472909	-0.382554
CP2	0.235040	-0.033130	0.193794
CP3	0.485310	-0.371987	-0.507788

[25]:
```
# Finalmente, para actualizar los valores de las componentes principales␣
 ↪extraemos los valores de la data con el método 'values' y le aplicamos el␣
 ↪método 'transform' a nuestro modelo PCA, todo esto almacenado dentro de un␣
 ↪data frame.
componentes_principales = pd.DataFrame(data = modelo_pca_final.transform(data.
 ↪values),columns=['CP1','CP2','CP3'])
componentes_principales
```

[25]:

	CP1	CP2	CP3
0	-75.342758	382.468019	53.378888
1	-111.685402	400.878454	5.139061
2	-15.921983	323.635463	-59.962362
3	-97.938059	455.730524	22.161697
4	-47.848072	492.172159	87.424159
..	…	…	…
100	74.859864	192.097726	238.274121
101	51.083255	236.496865	338.388957
102	17.889396	200.846885	177.723888
103	163.698217	334.230377	135.080454
104	21.176452	122.657929	163.303137

[105 rows x 3 columns]

3 Pregunta 2

Una empresa comercializa dos tipos de compuestos alimenticios para animales. El primero contiene 4 unidades de compuesto del nutriente A y 2 unidades del nutriente B, mientras que el segundo contiene 2 unidades del nutriente A y 3 unidades del nutriente B. Se desea conseguir una dieta que proporcione como mínimo 20 unidades de A y 18 unidades de B, además, el costo de una unidad del primer compuesto es de 2 soles y del segundo es de 2.5 soles. Tomando como referencia el enunciado del problema 2 y haciendo uso de Python, realice lo siguiente:

[31]:
```
import pulp
from scipy.optimize import linprog
from shapely.geometry import LineString
```

3.1 Seccion A

Escriba el problema de programación lineal que minimice el costo y utilizando la librería 'PuLP', encuentre las unidades para cada tipo de compuesto que minimicen el costo, así como el costo mínimo.

3.1.1 Programacion lineal

$x1$: Unidades del primer componente alimenticio.
$x2$: Unidades del segundo componente alimenticio.

Minimizar $z = 2x1 + 2.5x2$

sujeto a

$4x1 + 2x2 >= 20$
$2x1 + 3x2 >= 18$

$x1, x2 >= 0 $

[28]:
```
problema = pulp.LpProblem("costo",pulp.LpMinimize)

# Variables de decisión y marcamos la no negatividad

x1 = pulp.LpVariable("x1",lowBound = 0)
x2 = pulp.LpVariable("x2",lowBound = 0)

# Función objetivo
problema += 2*x1 + 2.5*x2

#Restricciones
problema += 4*x1 +  2*x2 >= 20
problema += 2*x1 +  3*x2 >= 18


# Resolvemos y mostramos la cantidad de soluciones
sol = problema.solve()
```

```
sol
```

[28]: 1

[29]:
```
print('z_max = {0:.4f}, x1 = {1:.4f}, x2 = {2:.4f}'.
     format(pulp.value(problema.objective),pulp.value(x1),pulp.value(x2)))
```

```
z_max = 16.0000, x1 = 3.0000, x2 = 4.0000
```

3.2 Seccion B

Despeje las variables de decisión a partir de las restricciones y grafique estas últimas en el plano cartesiano. Además, obtenga los vértices de la región factible, grafíquelas en el plano cartesiano e imprima las coordenadas de estos. Por último, evalúe a la función objetivo en los vértices de la región factible, imprima la solución óptima junto sus valores para la variable de decisión y grafique la región factible en el plano cartesiano.

Minimizar $z = 2x1 + 2.5x2$

sujeto a

$4x1 + 2x2 >= 20$
$2x1 + 3x2 >= 18$

$x1, x2 >= 0 $

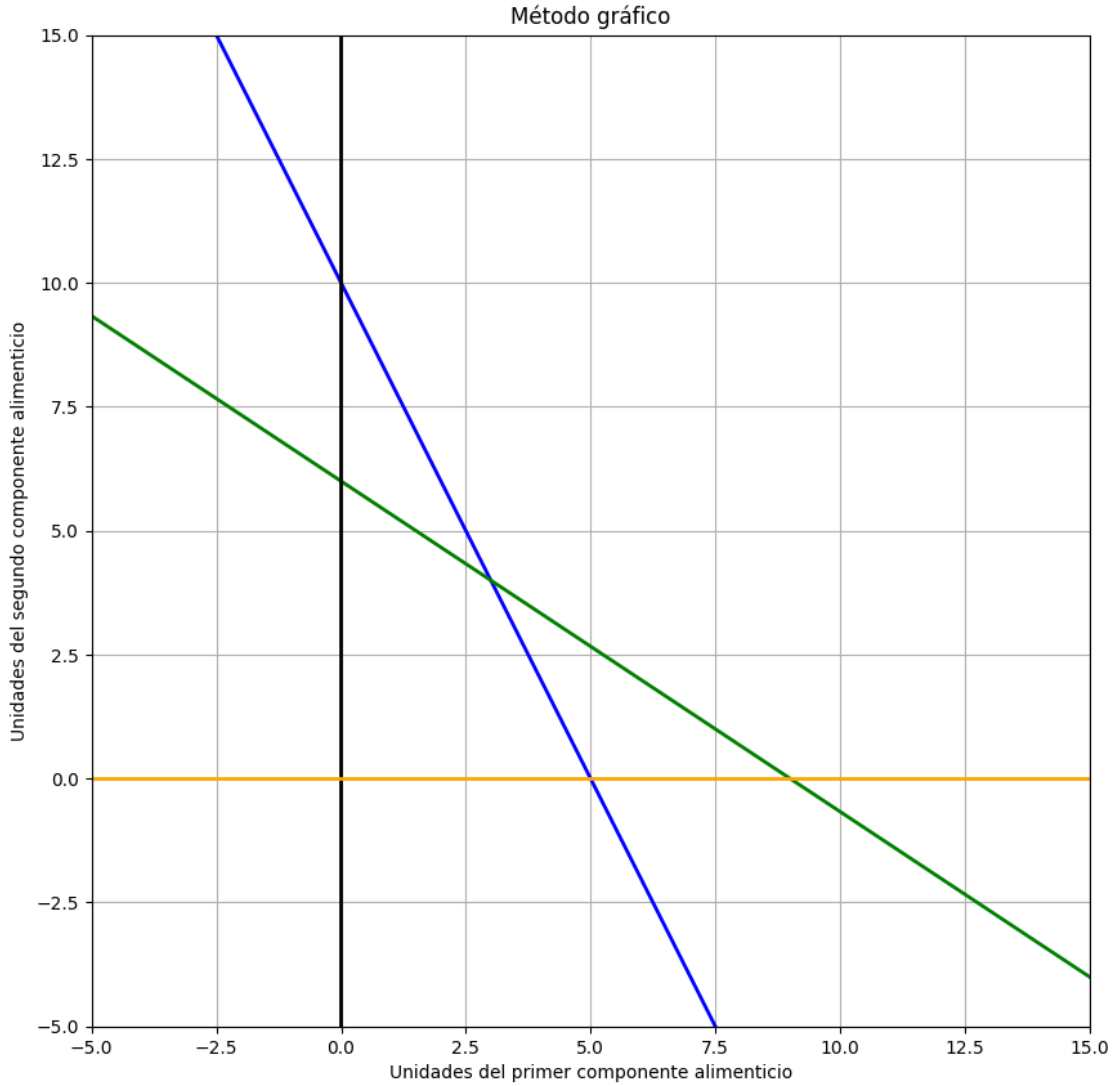
[35]:
```
# despejamos la variables y definimos las variables para tabular y graficar

x = np.arange(-150, 750, 1)
y = np.arange(-200, 1200, 1)
```

[36]:
```
x1=(20-2*y)/4
x2=(18-3*y)/2
x3 = 0*y
y1 = 0*x
```

[37]:
```
# Identificadores para las líneas
primera_linea = LineString(np.column_stack((x1, y)))
segunda_linea = LineString(np.column_stack((x2, y)))
tercera_linea = LineString(np.column_stack((x3, y)))
cuarta_linea = LineString(np.column_stack((x, y1)))
```

[48]:
```
# Graficando las líneas
plt.figure(figsize=(10,10))
plt.plot(x1, y, '-', linewidth=2, color='blue')
plt.plot(x2, y, '-', linewidth=2, color='green')
plt.plot(x3, y, '-', linewidth=2, color='black')
plt.plot(x, y1, '-', linewidth=2, color='orange')
plt.grid()
plt.xlim(-5,15)
```

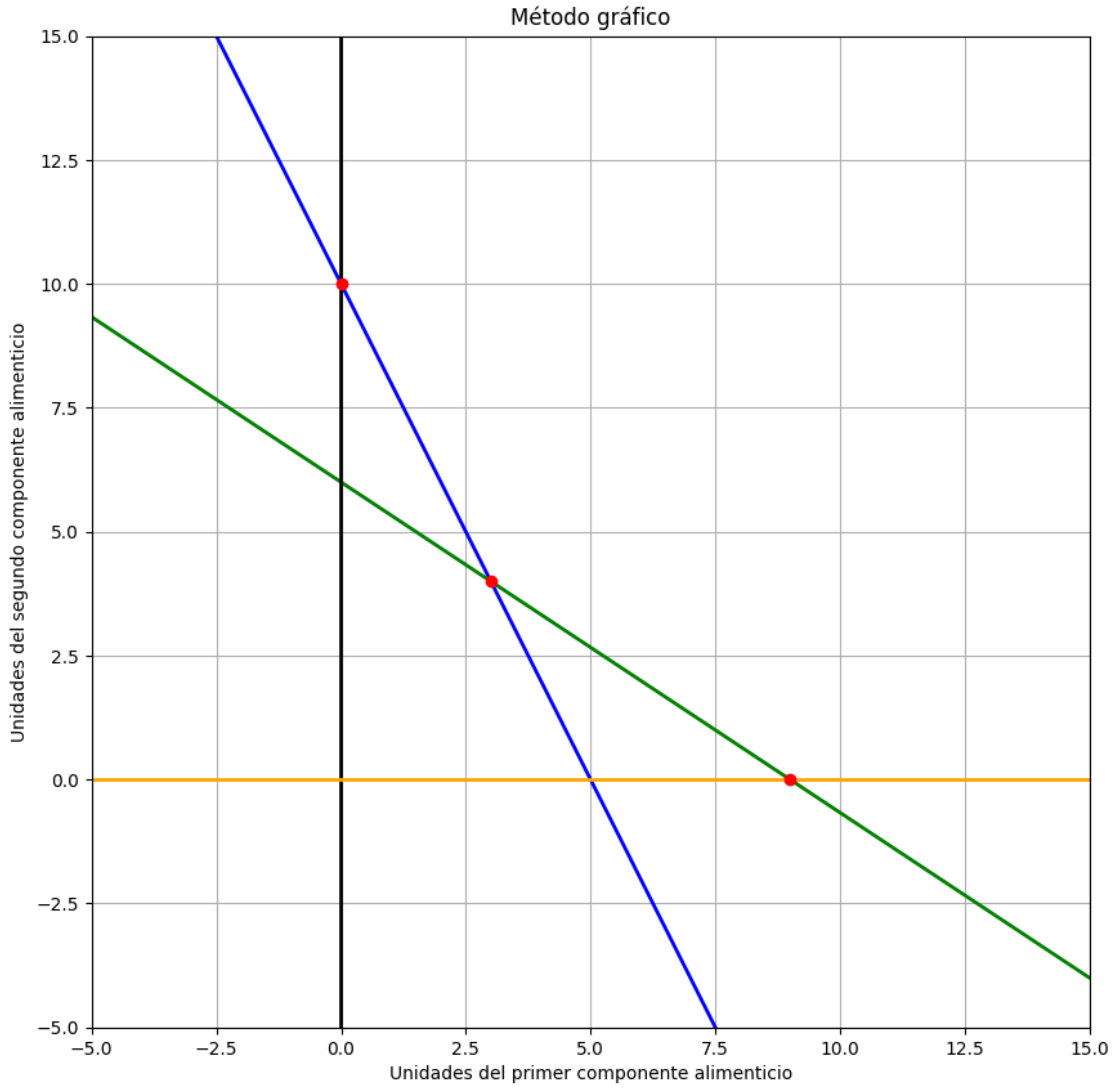
```
plt.ylim(-5,15)
plt.xlabel('Unidades del primer componente alimenticio')
plt.ylabel('Unidades del segundo componente alimenticio')
plt.title('Método gráfico')
plt.show()
```



[44]:
```
# Generando las intersecciones (vértices)
primera_interseccion = primera_linea.intersection(segunda_linea)
segunda_interseccion = primera_linea.intersection(tercera_linea)
tercera_interseccion = segunda_linea.intersection(cuarta_linea)
```

[47]:
```
# Graficando los vértices
plt.figure(figsize=(10,10))
```

```
plt.plot(x1, y, '-', linewidth=2, color='blue')
plt.plot(x2, y, '-', linewidth=2, color='green')
plt.plot(x3, y, '-', linewidth=2, color='black')
plt.plot(x, y1, '-', linewidth=2, color='orange')

plt.plot(*primera_interseccion.xy, 'o', color='red')
plt.plot(*segunda_interseccion.xy, 'o', color='red')
plt.plot(*tercera_interseccion.xy, 'o', color='red')


plt.grid()
plt.xlim(-5,15)
plt.ylim(-5,15)
plt.xlabel('Unidades del primer componente alimenticio')
plt.ylabel('Unidades del segundo componente alimenticio')
plt.title('Método gráfico')
plt.show()
```



[49]:
```
# Imprimiendo las coordenadas de los vértices
print('COORDENADAS DE LAS INTERSECCIONES \n')
print('(x1, y1): {} '.format(primera_interseccion))
print('(x2, y2): {} '.format(segunda_interseccion))
print('(x3, y3): {} '.format(tercera_interseccion))
```

```
COORDENADAS DE LAS INTERSECCIONES

(x1, y1): POINT (3 4)
(x2, y2): POINT (0 10)
(x3, y3): POINT (9 0)
```

[50]:
```
# Identificando los valores de las coordenadas x e y de cada vértice
xi1m, yi1m = primera_interseccion.xy
```

```
xi2m, yi2m = segunda_interseccion.xy
xi3m, yi3m = tercera_interseccion.xy
```

[51]:
```
# Cambiamos el formato de matriz a float
xi1 = np.float64(np.array(xi1m))
xi2 = np.float64(np.array(xi2m))
xi3 = np.float64(np.array(xi3m))

yi1 = np.float64(np.array(yi1m))
yi2 = np.float64(np.array(yi2m))
yi3 = np.float64(np.array(yi3m))
```

[57]:
```
# Evaluando la función objetivo en cada vértice
FOi1 = 2*xi1 + 2.5*yi1
FOi2 = 2*xi2 + 2.5*yi2
FOi3 = 2*xi3 + 2.5*yi3
```

[58]:
```
# Imprimiendo las evaluaciones de la función objetivo en cada vértice
print('EVALUACIÓN DE LA FO EN LOS VÉRTICES \n')
print('f(x1, y1) = {} soles'.format(FOi1))
print('f(x2, y2) = {} soles'.format(FOi2))
print('f(x3, y3) = {} soles'.format(FOi3))
```

```
EVALUACIÓN DE LA FO EN LOS VÉRTICES

f(x1, y1) = 16.0 soles
f(x2, y2) = 25.0 soles
f(x3, y3) = 18.0 soles
```

[61]:
```
# Calculando e imprimiendo la solución óptima
z_min = min(FOi1, FOi2, FOi3)
print('SOLUCIÓN ÓPTIMA \n')
print('z_max = {} soles'.format(z_min))
```

```
SOLUCIÓN ÓPTIMA

z_max = 16.0 soles
```

[64]:
```
# Graficando la región factible a partir de las coordenadas de los vértices
m = [xi1, xi2, xi3]
n = [yi1, yi2, yi3]
plt.figure(figsize=(10,10))
plt.plot(x1, y, '-', linewidth=2, color='blue')
plt.plot(x2, y, '-', linewidth=2, color='green')
plt.plot(x3, y, '-', linewidth=2, color='black')
plt.plot(x, y1, '-', linewidth=2, color='orange')

plt.plot(*primera_interseccion.xy, 'o', color='red')
```

```
plt.plot(*segunda_interseccion.xy, 'o', color='red')
plt.plot(*tercera_interseccion.xy, 'o', color='red')

plt.fill(m, n, color='silver')
plt.grid()
plt.xlim(-5,15)
plt.ylim(-5,15)
plt.xlabel('Número de lotes de tapas producidos por semana')
plt.ylabel('Número de lotes de envases producidos por semana')
plt.title('Método gráfico')
```

[64]: Text(0.5, 1.0, 'Método gráfico')