estad195173sticaavanzada2final

October 20, 2023

0.1 # Estadística Avanzada - Actividad 2

Diego Sú Gómez - A01620476

0.2 ## Importación de librerías

[1]:
```
#Importar las liibrerías
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.preprocessing import MinMaxScaler, StandardScaler, scale
from scipy.stats.mstats import winsorize
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import mean_squared_error
from sklearn.decomposition import PCA
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
```

0.3 ## Lectura del archivo

[2]:
```
#Cargar los datos
df = pd.read_csv("abalone.data",header=None)
df.head()
```

[2]:

	0	1	2	3	4	5	6	7	8
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

[3]:
```
#Nombrar las columnas
df = df.rename(columns={0:"Sex",1:"Length",2:"Diameter",3:"Height",4:
 ↪"Whole_weight",5:"Shucked_weight",6:"Viscera_weight",7:"Shell_weight",8:
 ↪"Rings"})
df.head()
```

[3]:
```
  Sex  Length  Diameter  Height  Whole_weight  Shucked_weight  Viscera_weight  \
0   M   0.455     0.365   0.095        0.5140          0.2245          0.1010
1   M   0.350     0.265   0.090        0.2255          0.0995          0.0485
2   F   0.530     0.420   0.135        0.6770          0.2565          0.1415
3   M   0.440     0.365   0.125        0.5160          0.2155          0.1140
4   I   0.330     0.255   0.080        0.2050          0.0895          0.0395

   Shell_weight  Rings
0         0.150     15
1         0.070      7
2         0.210      9
3         0.155     10
4         0.055      7
```

[4]:
```
df.shape
```

[4]:
```
(4177, 9)
```

[5]:
```
df.columns
```

[5]:
```
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight',
       'Viscera_weight', 'Shell_weight', 'Rings'],
      dtype='object')
```

[6]:
```
df.describe()
```

[6]:
```
            Length     Diameter       Height  Whole_weight  Shucked_weight  \
count  4177.000000  4177.000000  4177.000000   4177.000000     4177.000000
mean      0.523992     0.407881     0.139516      0.828742        0.359367
std       0.120093     0.099240     0.041827      0.490389        0.221963
min       0.075000     0.055000     0.000000      0.002000        0.001000
25%       0.450000     0.350000     0.115000      0.441500        0.186000
50%       0.545000     0.425000     0.140000      0.799500        0.336000
75%       0.615000     0.480000     0.165000      1.153000        0.502000
max       0.815000     0.650000     1.130000      2.825500        1.488000

       Viscera_weight  Shell_weight        Rings
count     4177.000000   4177.000000  4177.000000
mean         0.180594      0.238831     9.933684
std          0.109614      0.139203     3.224169
min          0.000500      0.001500     1.000000
25%          0.093500      0.130000     8.000000
50%          0.171000      0.234000     9.000000
75%          0.253000      0.329000    11.000000
max          0.760000      1.005000    29.000000
```

0.4 ## Generación del modelo

[7]:
```
#Definir las variables predictoras y la de respuesta
x = df[['Length', 'Diameter', 'Height', 'Whole_weight',␣
 ↪'Shucked_weight','Viscera_weight', 'Shell_weight']]
y = df["Rings"]
```

[8]:
```
#Generar el modelo
x = sm.add_constant(x)
model = sm.OLS(y,x)
model = model.fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.528
Model:                            OLS   Adj. R-squared:                  0.527
Method:                 Least Squares   F-statistic:                     665.2
Date:                Fri, 20 Oct 2023   Prob (F-statistic):               0.00
Time:                        23:22:24   Log-Likelihood:                -9250.0
No. Observations:                4177   AIC:                         1.852e+04
Df Residuals:                    4169   BIC:                         1.857e+04
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
==
                     coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--
const              2.9852      0.269     11.092      0.000       2.458
3.513
Length            -1.5719      1.825     -0.861      0.389      -5.149
2.006
Diameter          13.3609      2.237      5.972      0.000       8.975
17.747
Height            11.8261      1.548      7.639      0.000       8.791
14.861
Whole_weight       9.2474      0.733     12.622      0.000       7.811
10.684
Shucked_weight   -20.2139      0.823    -24.552      0.000     -21.828
-18.600
Viscera_weight    -9.8297      1.304     -7.538      0.000     -12.386
-7.273
Shell_weight       8.5762      1.137      7.545      0.000       6.348
10.805
==============================================================================
Omnibus:                      933.799   Durbin-Watson:                   1.387
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2602.745
Skew:                           1.174   Prob(JB):                         0.00
Kurtosis:                       6.072   Cond. No.                         131.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[9]:

```python
#Obtener los parámetros del modelo
print(model.params)
```

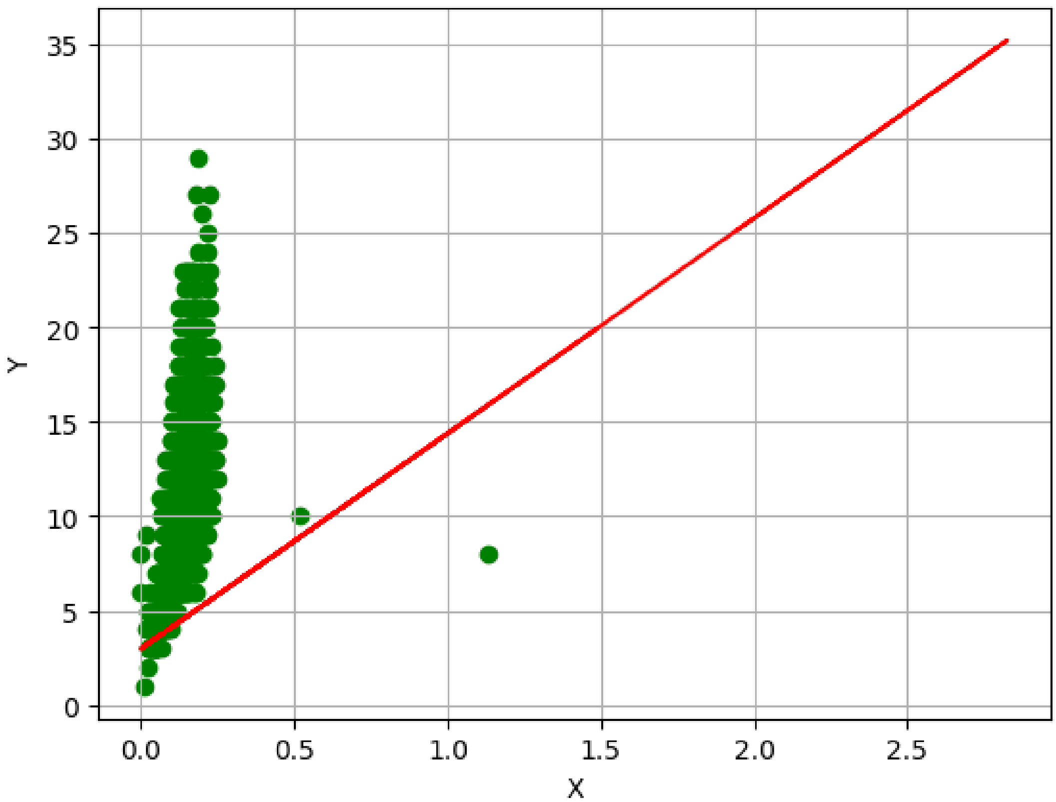
```
const             2.985154
Length           -1.571897
Diameter         13.360916
Height           11.826072
Whole_weight      9.247414
Shucked_weight  -20.213913
Viscera_weight   -9.829675
Shell_weight      8.576242
dtype: float64
```

[10]:

```python
#Generar las predicciones del modelo
ypred = model.predict(x)
```

[11]:

```python
#Graficar la exactitud del modelo de regresión
betas = model.params.to_numpy()
ecuacion = betas[0] + x*betas[1] + x*betas[2] + x*betas[3] + x*betas[4] +
  ↪x*betas[5] + x*betas[6] + x*betas[7]
plt.scatter(df['Height'],y,color="green")
plt.plot(x,ecuacion, color="red")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid()
```



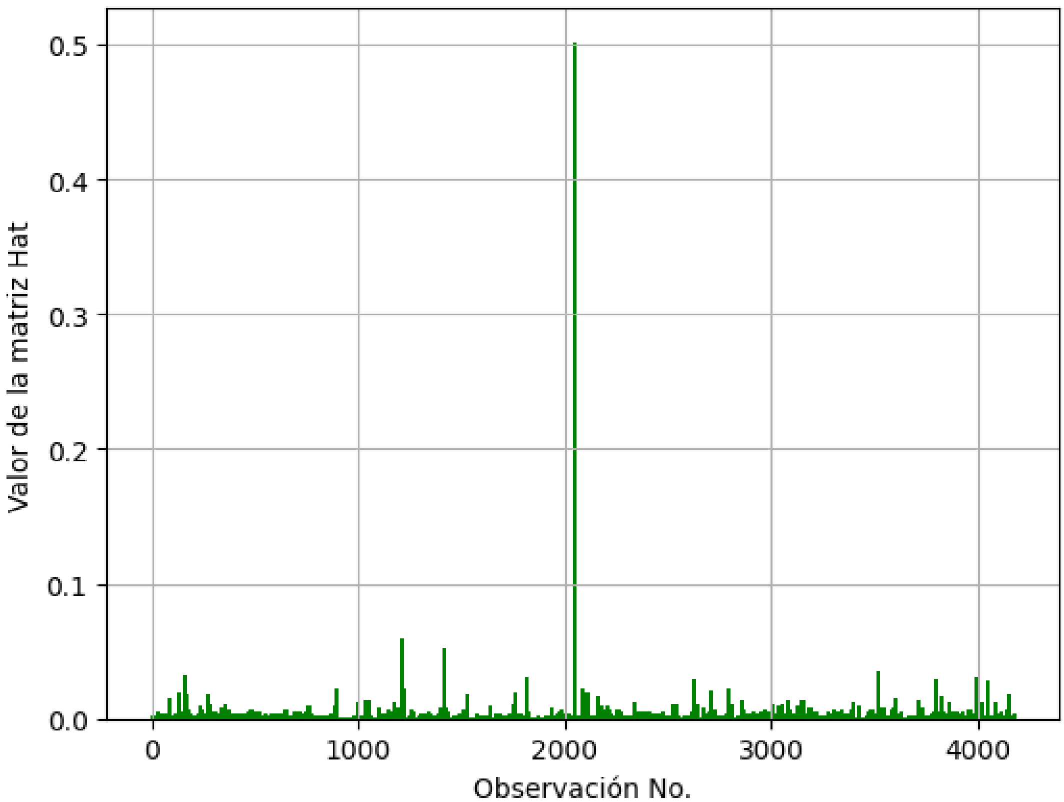
0.5 ## Identificación de Puntos influyentes y no influyentes

[12]:
```
#Obtener los scores de influencia y la matriz Hat
influence = model.get_influence()
h_mat = influence.hat_matrix_diag
print(h_mat)
```

```
[0.00089205 0.00076875 0.00072514 … 0.00160134 0.00103437 0.0033281 ]
```

[13]:
```
#Graficar los residuos de la matriz hat
plt.bar(df.index, h_mat, width=20, color="green")
plt.grid()
plt.xlabel("Observación No.")
plt.ylabel("Valor de la matriz Hat")
```

[13]:
```
Text(0, 0.5, 'Valor de la matriz Hat')
```



[14]:
```
#Encontrar los valores con más leverage
map = sorted(list(enumerate(h_mat)),key=lambda item: item[1], reverse=True)
max_value_indexes = [item[0] for item in map]
most10 = max_value_indexes[:10]
print("Valores con más leverage:")
print(df.iloc[most10])
```

```
Valores con más leverage:
     Sex  Length  Diameter  Height  Whole_weight  Shucked_weight  \
2051   F   0.455     0.355   1.130        0.5940          0.3320
1210   I   0.185     0.375   0.120        0.4645          0.1960
1417   M   0.705     0.565   0.515        2.2100          1.1075
3518   M   0.710     0.570   0.195        1.3480          0.8985
163    F   0.725     0.560   0.210        2.1410          0.6500
3996   I   0.315     0.230   0.000        0.1340          0.0575
1812   M   0.650     0.485   0.160        1.7395          0.5715
2627   I   0.275     0.205   0.070        0.1055          0.4950
3800   M   0.740     0.580   0.205        2.3810          0.8155
4052   M   0.625     0.470   0.145        1.7855          0.6750

      Viscera_weight  Shell_weight  Rings
```

```
2051          0.1160        0.1335      8
1210          0.1045        0.1500      6
1417          0.4865        0.5120     10
3518          0.4435        0.4535     11
163           0.3980        1.0050     18
3996          0.0285        0.3505      6
1812          0.2785        0.3075     10
2627          0.0190        0.0315      5
3800          0.4695        0.4880     12
4052          0.2470        0.3245     13
```

[15]:

```python
#Calcular la distancia Cook de cada uno de los registros (observaciones)
np.set_printoptions(suppress=True)
cooks = influence.cooks_distance[0]
print(cooks[:10])
```

```
[0.00087893 0.0000011  0.00006288 0.0000137  0.         0.00000042
 0.00223236 0.00049142 0.00000104 0.00135537]
```

[16]:

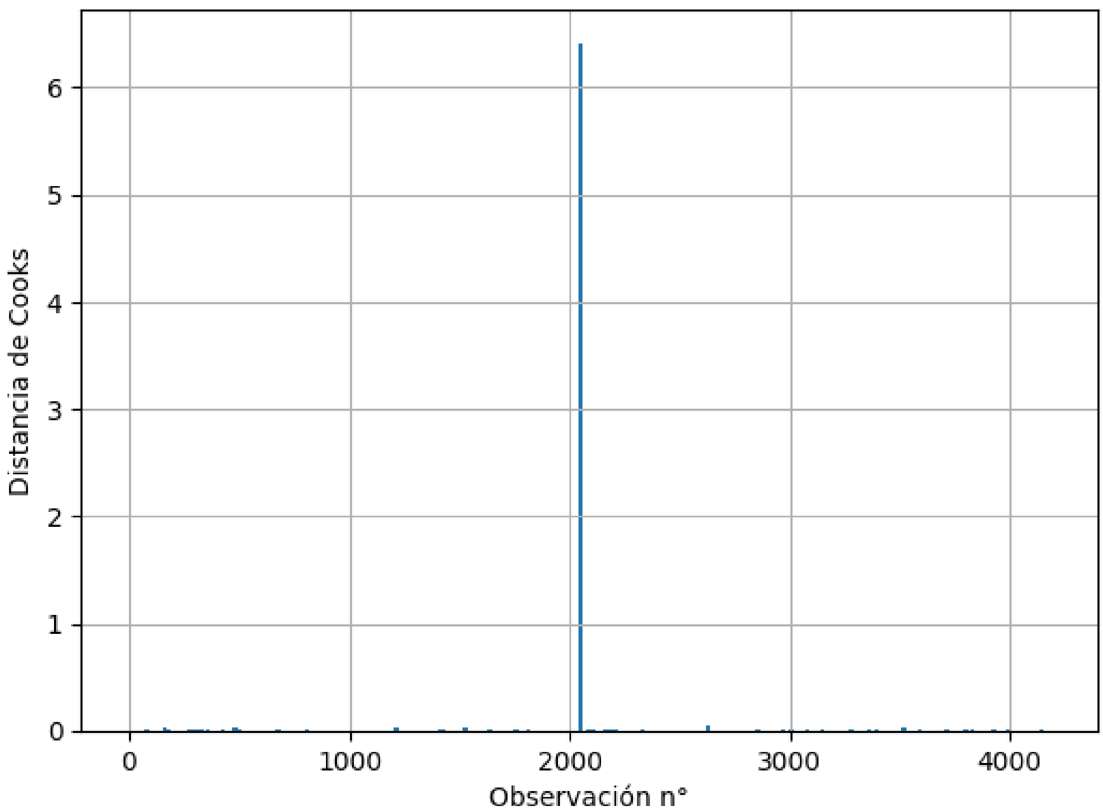
```python
print(influence.summary_frame())
```

```
      dfb_const  dfb_Length  dfb_Diameter  dfb_Height  dfb_Whole_weight  \
0      0.013647   -0.032927      0.045769   -0.048025          0.010240
1     -0.001956    0.000391      0.000189    0.000044          0.000010
2      0.009360    0.001597     -0.007949    0.001964         -0.006124
3      0.002439   -0.008192      0.008150    0.001118         -0.000097
4      0.000109   -0.000049      0.000021   -0.000022          0.000027
…           …           …             …           …                …
4172  -0.001359   -0.001517      0.002022    0.001833         -0.002014
4173  -0.001104    0.002819     -0.002169   -0.001326         -0.000042
4174   0.002614    0.006257     -0.002471   -0.024792          0.004487
4175  -0.001575    0.001016     -0.000027   -0.001537         -0.003201
4176   0.006267    0.000073     -0.003912   -0.000831          0.004942

      dfb_Shucked_weight  dfb_Viscera_weight  dfb_Shell_weight       cooks_d  \
0              -0.006130           -0.016609         -0.009500  8.789307e-04
1              -0.000171           -0.000053         -0.000109  1.104468e-06
2               0.010278            0.005830          0.006062  6.288230e-05
3              -0.000220            0.000446         -0.001137  1.370315e-05
4              -0.000009           -0.000013         -0.000019  2.800798e-09
…                    …                   …                …            …
4172           -0.000165            0.004425         -0.000339  5.445967e-06
4173            0.000044           -0.000011         -0.000184  1.825037e-06
4174           -0.002879           -0.012517          0.005699  1.444179e-04
4175            0.002913            0.002542          0.001413  3.855849e-06
4176            0.006734           -0.009569         -0.001904  7.827310e-05

      standard_resid  hat_diag  dffits_internal  student_resid    dffits
```

```
0        2.806306  0.000892       0.083854      2.808623  0.083923
1       -0.107167  0.000769      -0.002972     -0.107155 -0.002972
2       -0.832610  0.000725      -0.022429     -0.832579 -0.022428
3        0.328052  0.001018       0.010470      0.328017  0.010469
4        0.004717  0.001006       0.000150      0.004717  0.000150
…             …         …              …             …         …
4172     0.193886  0.001158       0.006601      0.193864  0.006600
4173     0.127301  0.000900       0.003821      0.127286  0.003821
4174    -0.848723  0.001601      -0.033990     -0.848695 -0.033989
4175     0.172601  0.001034       0.005554      0.172581  0.005553
4176     0.433041  0.003328       0.025024      0.432999  0.025021

[4177 rows x 14 columns]
```

[17]:

```python
#Visualizar los puntos influyentes
plt.figure(figsize=(7,5))
plt.bar(df.index, cooks, width=20)
plt.xlabel("Observación n°")
plt.ylabel("Distancia de Cooks")
plt.grid()
```



[18]:
```python
#Encontrar los valores con más distancia de Cooks
map = sorted(list(enumerate(cooks)), key=lambda item:item[1], reverse=True)
max_value_indexes = [item[0] for item in map]
print("Observaciones con la mayor distancia de Cooks")
print(df.iloc[max_value_indexes[:10]])
```

```
Observaciones con la mayor distancia de Cooks
     Sex  Length  Diameter  Height  Whole_weight  Shucked_weight  \
2051   F   0.455     0.355   1.130        0.5940          0.3320
2627   I   0.275     0.205   0.070        0.1055          0.4950
480    F   0.700     0.585   0.185        1.8075          0.7055
3518   M   0.710     0.570   0.195        1.3480          0.8985
1528   M   0.725     0.575   0.240        2.2100          1.3510
1210   I   0.185     0.375   0.120        0.4645          0.1960
1216   I   0.310     0.225   0.070        0.1055          0.4350
163    F   0.725     0.560   0.210        2.1410          0.6500
2108   M   0.665     0.535   0.225        2.1835          0.7535
81     M   0.620     0.510   0.175        1.6150          0.5105

      Viscera_weight  Shell_weight  Rings
2051          0.1160        0.1335      8
2627          0.0190        0.0315      5
480           0.3215        0.4750     29
3518          0.4435        0.4535     11
1528          0.4130        0.5015     13
1210          0.1045        0.1500      6
1216          0.0150        0.0400      5
163           0.3980        1.0050     18
2108          0.3910        0.8850     27
81            0.1920        0.6750     12
```

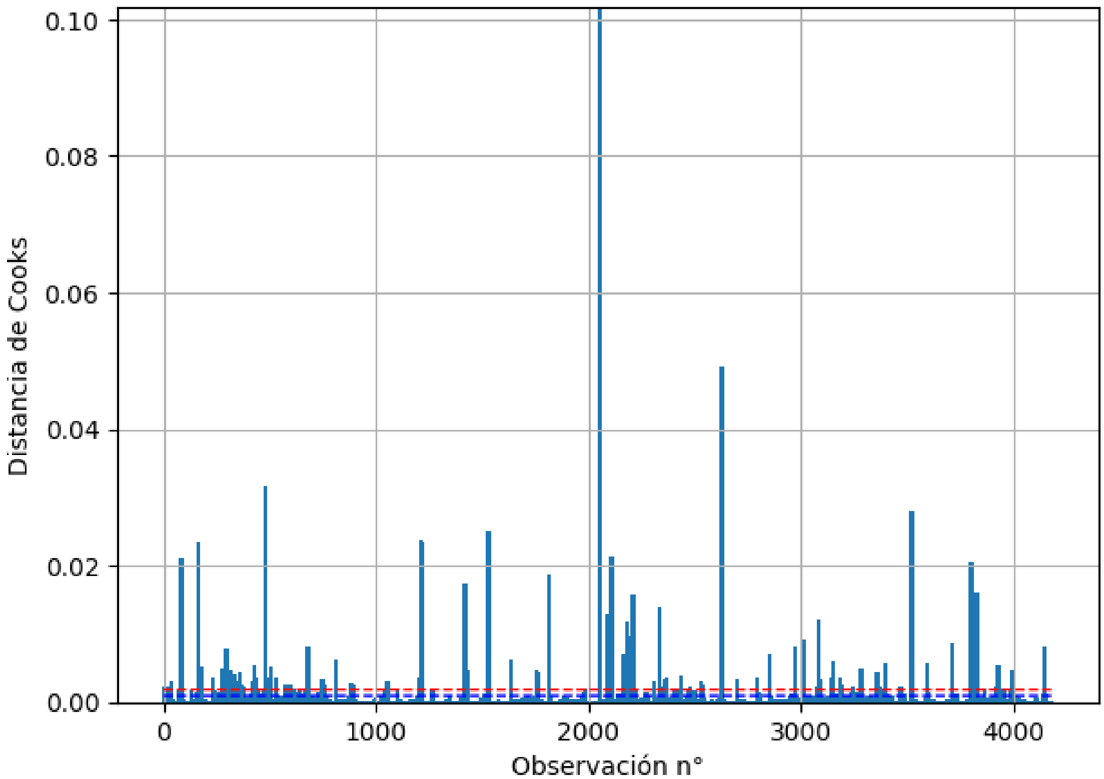
[19]:
```python
prom_cooks = np.mean(cooks)
print(prom_cooks)
```

```
0.0018730877579285728
```

[20]:
```python
#Graficar los valores adecuados
mean_cooks_list = [prom_cooks for i in df.index]
cooks_limites = [4/len(cooks) for i in df.index]
```

[21]:
```python
plt.figure(figsize=(7,5))
plt.bar(df.index, cooks, width=20)
plt.plot(df.index, mean_cooks_list, color="red", linestyle="--", linewidth=1)
plt.plot(df.index, cooks_limites, color="blue", linestyle="--", linewidth=1)
plt.xlabel("Observación n°")
plt.ylabel("Distancia de Cooks")
plt.ylim(top=max(mean_cooks_list + cooks_limites)+ 1e-1)
```

```
plt.grid()
```



[22]:
```
#Encontrar los puntos influyentes del dataset
puntos_influyentes = df.index[cooks > 4/len(cooks)]
print(puntos_influyentes[:10])
```

```
Int64Index([6, 9, 32, 33, 36, 67, 72, 81, 83, 85], dtype='int64')
```

[23]:
```
df.iloc[puntos_influyentes,:].head(10)
```

[23]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	\
6	F	0.530	0.415	0.150	0.7775	0.2370	
9	F	0.550	0.440	0.150	0.8945	0.3145	
32	M	0.665	0.525	0.165	1.3380	0.5515	
33	F	0.680	0.550	0.175	1.7980	0.8150	
36	F	0.540	0.475	0.155	1.2170	0.5305	
67	F	0.595	0.495	0.185	1.2850	0.4160	
72	F	0.595	0.475	0.170	1.2470	0.4800	
81	M	0.620	0.510	0.175	1.6150	0.5105	
83	M	0.595	0.475	0.160	1.3175	0.4080	
85	F	0.570	0.465	0.180	1.2950	0.3390	

```
    Viscera_weight  Shell_weight  Rings
6           0.1415         0.330     20
9           0.1510         0.320     19
32          0.3575         0.350     18
33          0.3925         0.455     19
36          0.3075         0.340     16
67          0.2240         0.485     13
72          0.2250         0.425     20
81          0.1920         0.675     12
83          0.2340         0.580     21
85          0.2225         0.440     12
```

[24]:
```python
#Encontrar los puntos no nfluyentes del dataset
puntos_noinfluyentes = df.index[cooks < 4/len(cooks)]
print(puntos_noinfluyentes[:10])
```

```
Int64Index([0, 1, 2, 3, 4, 5, 7, 8, 10, 11], dtype='int64')
```

[25]:
```python
df.iloc[puntos_noinfluyentes,:].head(10)
```

[25]:
```
   Sex  Length  Diameter  Height  Whole_weight  Shucked_weight  \
0    M   0.455     0.365   0.095        0.5140          0.2245
1    M   0.350     0.265   0.090        0.2255          0.0995
2    F   0.530     0.420   0.135        0.6770          0.2565
3    M   0.440     0.365   0.125        0.5160          0.2155
4    I   0.330     0.255   0.080        0.2050          0.0895
5    I   0.425     0.300   0.095        0.3515          0.1410
7    F   0.545     0.425   0.125        0.7680          0.2940
8    M   0.475     0.370   0.125        0.5095          0.2165
10   F   0.525     0.380   0.140        0.6065          0.1940
11   M   0.430     0.350   0.110        0.4060          0.1675

    Viscera_weight  Shell_weight  Rings
0           0.1010         0.150     15
1           0.0485         0.070      7
2           0.1415         0.210      9
3           0.1140         0.155     10
4           0.0395         0.055      7
5           0.0775         0.120      8
7           0.1495         0.260     16
8           0.1125         0.165      9
10          0.1475         0.210     14
11          0.0810         0.135     10
```

0.6 ## Identificación de outliers

[26]:
```
#Encontrar los outliers del dataset
lim1 = x.mean() + 3*x.std()
lim2 = x.mean() - 3*x.std()
print("Limite superior:",lim1)
print("\nLimite inferior:",lim2)
print("\nNúmero de outliers:",x[(x>lim1)|(x<lim2)].shape)
```

```
Limite superior: const             1.000000
Length            0.884271
Diameter          0.705601
Height            0.264998
Whole_weight      2.299909
Shucked_weight    1.025256
Viscera_weight    0.509436
Shell_weight      0.656439
dtype: float64

Limite inferior: const             1.000000
Length            0.163713
Diameter          0.110162
Height            0.014035
Whole_weight     -0.642425
Shucked_weight   -0.306521
Viscera_weight   -0.148249
Shell_weight     -0.178777
dtype: float64

Número de outliers: (4177, 8)
```

[27]:
```
#Identificar los outliers mediante el rango intercuartílico
q1 = x.quantile(0.25)
q3 = x.quantile(0.75)
rango = q3-q1

print("Q1:",q1)
print("\nQ3:",q3)
print("\nRango intercuartílico:",rango)

outliers_riq = (x < q1-1.5*rango) | (x>q3+1.5*rango)
print("\nNúmero de outliers:",x[outliers_riq].shape)
```

```
Q1: const             1.0000
Length            0.4500
Diameter          0.3500
Height            0.1150
Whole_weight      0.4415
```

```
Shucked_weight    0.1860
Viscera_weight    0.0935
Shell_weight      0.1300
Name: 0.25, dtype: float64

Q3: const             1.000
Length            0.615
Diameter          0.480
Height            0.165
Whole_weight      1.153
Shucked_weight    0.502
Viscera_weight    0.253
Shell_weight      0.329
Name: 0.75, dtype: float64

Rango intercuartílico: const             0.0000
Length            0.1650
Diameter          0.1300
Height            0.0500
Whole_weight      0.7115
Shucked_weight    0.3160
Viscera_weight    0.1595
Shell_weight      0.1990
dtype: float64

Número de outliers: (4177, 8)
```

0.7 ## Estandarización de los datos

[28]:
```python
#Estandarizar los datos para eliminar los outliers
x = df[['Length', 'Diameter', 'Height', 'Whole_weight',␣
 ↪'Shucked_weight','Viscera_weight', 'Shell_weight']]
scaler = MinMaxScaler(feature_range=(-1,1))
scaler.fit(x)
print("Valores máximos:",scaler.data_max_)
print("\nTransformaciones:")
print(scaler.transform(x),"\n")
x_transformed1 = scaler.transform(x)
```

```
Valores máximos: [0.815  0.65   1.13   2.8255 1.488  0.76   1.005 ]

Transformaciones:
[[ 0.02702703  0.04201681 -0.83185841 … -0.69939475 -0.73535221
  -0.70403587]
 [-0.25675676 -0.29411765 -0.84070796 … -0.86751849 -0.87360105
  -0.86347783]
 [ 0.22972973  0.22689076 -0.76106195 … -0.65635508 -0.62870309
  -0.58445441]
```

```
…
 [ 0.41891892  0.41176471 -0.63716814 … -0.29455279 -0.24423963
  -0.38913802]
 [ 0.48648649  0.44537815 -0.73451327 … -0.28715535 -0.31402238
  -0.41305431]
 [ 0.71621622  0.68067227 -0.65486726 …  0.27034297 -0.00987492
  -0.01644245]]
```

[29]:

```python
scaler2 = StandardScaler()
scaler2.fit(x)
print("Promedios:",scaler2.mean_)
print("\nTransformaciones:")
print(scaler2.transform(x),"\n")
x_transformed2 = scaler2.transform(x)
```

```
Promedios: [0.5239921  0.40788125 0.1395164  0.82874216 0.35936749 0.18059361
 0.23883086]

Transformaciones:
[[-0.57455813 -0.43214879 -1.06442415 … -0.60768536 -0.72621157
  -0.63821689]
 [-1.44898585 -1.439929   -1.18397831 … -1.17090984 -1.20522124
  -1.21298732]
 [ 0.05003309  0.12213032 -0.10799087 … -0.4634999  -0.35668983
  -0.20713907]
 …
 [ 0.6329849   0.67640943  1.56576738 …  0.74855917  0.97541324
   0.49695471]
 [ 0.84118198  0.77718745  0.25067161 …  0.77334105  0.73362741
   0.41073914]
 [ 1.54905203  1.48263359  1.32665906 …  2.64099341  1.78744868
   1.84048058]]
```

0.8 ## Evaluar el modelo con los datos transformados

[30]:

```python
#Evaluar el modelo con los datos estandarizados con el MinMaxScaler
x_transformed1 = sm.add_constant(x_transformed1)
model = sm.OLS(y,x_transformed1)
model = model.fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.528
Model:                            OLS   Adj. R-squared:                  0.527
Method:                 Least Squares   F-statistic:                     665.2
```

```
Date:                Fri, 20 Oct 2023   Prob (F-statistic):               0.00
Time:                        23:23:21   Log-Likelihood:                -9250.0
No. Observations:                4177   AIC:                         1.852e+04
Df Residuals:                    4169   BIC:                         1.857e+04
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         12.2796      0.717     17.117      0.000      10.873      13.686
x1            -0.5816      0.675     -0.861      0.389      -1.905       0.742
x2             3.9749      0.666      5.972      0.000       2.670       5.280
x3             6.6817      0.875      7.639      0.000       4.967       8.397
x4            13.0550      1.034     12.622      0.000      11.027      15.083
x5           -15.0290      0.612    -24.552      0.000     -16.229     -13.829
x6            -3.7328      0.495     -7.538      0.000      -4.704      -2.762
x7             4.3031      0.570      7.545      0.000       3.185       5.421
==============================================================================
Omnibus:                      933.799   Durbin-Watson:                   1.387
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2602.745
Skew:                           1.174   Prob(JB):                         0.00
Kurtosis:                       6.072   Cond. No.                         64.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[31]:
```python
#Evaluar el modelo con los datos estandarizados con el StandardScaler
x_transformed2 = sm.add_constant(x_transformed2)
model = sm.OLS(y,x_transformed2)
model = model.fit()
print(model.summary())
```

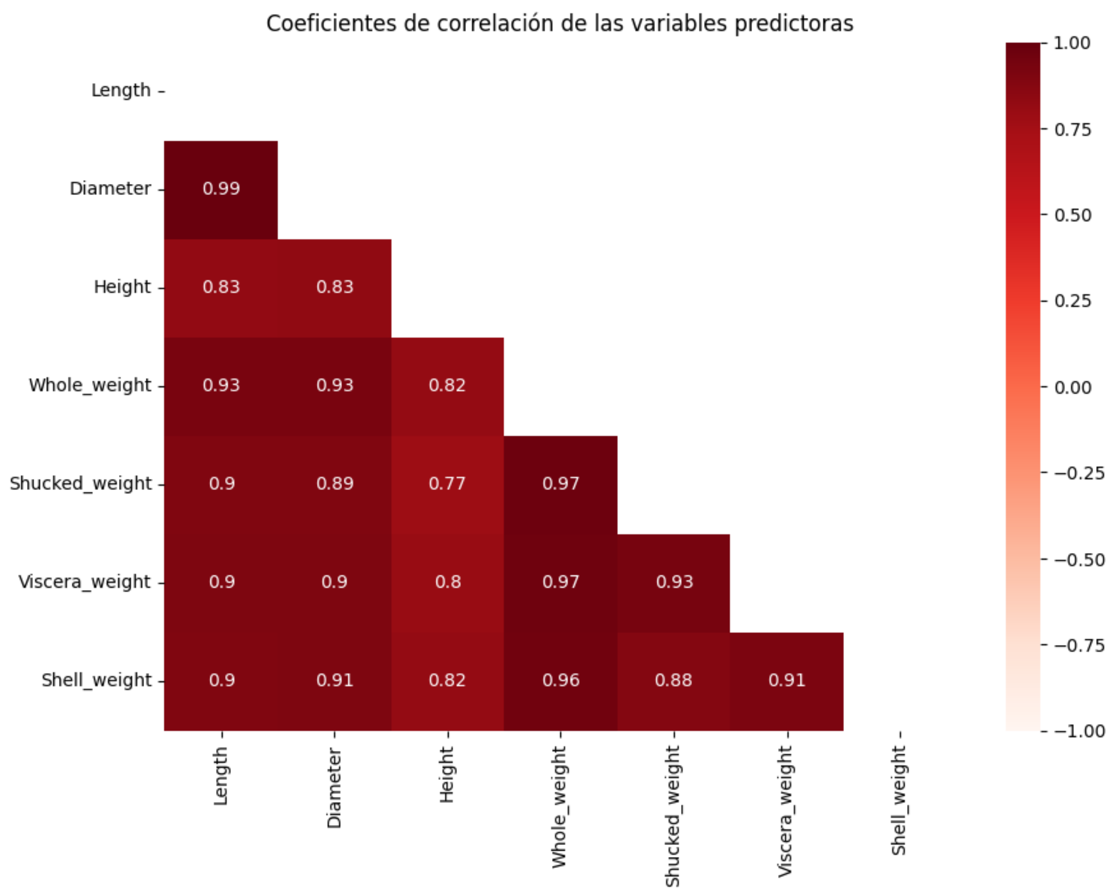
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.528
Model:                            OLS   Adj. R-squared:                  0.527
Method:                 Least Squares   F-statistic:                     665.2
Date:                Fri, 20 Oct 2023   Prob (F-statistic):               0.00
Time:                        23:23:21   Log-Likelihood:                -9250.0
No. Observations:                4177   AIC:                         1.852e+04
Df Residuals:                    4169   BIC:                         1.857e+04
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
const          9.9337      0.034    289.481      0.000       9.866      10.001
x1            -0.1888      0.219     -0.861      0.389      -0.618       0.241
x2             1.3258      0.222      5.972      0.000       0.891       1.761
x3             0.4946      0.065      7.639      0.000       0.368       0.622
x4             4.5343      0.359     12.622      0.000       3.830       5.239
x5            -4.4862      0.183    -24.552      0.000      -4.844      -4.128
x6            -1.0773      0.143     -7.538      0.000      -1.358      -0.797
x7             1.1937      0.158      7.545      0.000       0.884       1.504
==============================================================================
Omnibus:                      933.799   Durbin-Watson:                   1.387
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2602.745
Skew:                           1.174   Prob(JB):                         0.00
Kurtosis:                       6.072   Cond. No.                         30.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

0.9 ## Multicolinealidad con VIF

[32]:
```
#Encontrar las variables con coeficientes de correlación altos
plt.figure(figsize=(10,7))
x = df[['Length', 'Diameter', 'Height', 'Whole_weight',␣
 ↪'Shucked_weight','Viscera_weight', 'Shell_weight']]
triangulo = np.triu(np.ones_like(x.corr(numeric_only=True), dtype="bool"))
sns.heatmap(x.corr(numeric_only=True), annot=True, mask=triangulo, vmin=-1,␣
 ↪vmax=1, cmap="Reds")
plt.title("Coeficientes de correlación de las variables predictoras")
```

[32]:
```
Text(0.5, 1.0, 'Coeficientes de correlación de las variables predictoras')
```



[33]:
```
#Calcular el vif de cada una de las variables
def vif(variables):
  x = df[variables].copy()
  x["intercepto"] = 1

  vif_df = pd.DataFrame()
  vif_df["variable"] = x.columns

  vif_df["VIF"] = [variance_inflation_factor(x.values, i) for i in range(len(x.
↪columns))]
  vif_df = vif_df[vif_df["variable"] != "intercepto"]

  return vif_df
```

[34]:
```
#Crear una función para evaluar el modelo con las variables seleccionadas
def test_model(variables):
  x = df[variables]
  y = df["Rings"]
  #Estandarizar los datos para eliminar outliers
```

```
    scaler = StandardScaler()
    scaler.fit(x)
    x = scaler.transform(x)
    x = sm.add_constant(x)
    #Generar y ajustar el modelo, para luego evaluarlo
    model = sm.OLS(y,x)
    model = model.fit()
    ypred = model.predict(x)

    print(model.summary())
    print("\nMSE:", mean_squared_error(y,ypred))
```

[35]:

```
#Evaluar las variables para ver cuáles eliminar
x_vars = ['Length', 'Diameter', 'Height', 'Whole_weight',␣
 ↪'Shucked_weight','Viscera_weight', 'Shell_weight']
print(vif(x_vars).sort_values("VIF", ascending=False))
```

```
         variable         VIF
3    Whole_weight  109.592750
1        Diameter   41.845452
0          Length   40.771813
4  Shucked_weight   28.353191
6    Shell_weight   21.258289
5  Viscera_weight   17.346276
2          Height    3.559939
```

[36]:

```
test_model(x_vars)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.528
Model:                            OLS   Adj. R-squared:                  0.527
Method:                 Least Squares   F-statistic:                     665.2
Date:                Fri, 20 Oct 2023   Prob (F-statistic):               0.00
Time:                        23:23:21   Log-Likelihood:                -9250.0
No. Observations:                4177   AIC:                         1.852e+04
Df Residuals:                    4169   BIC:                         1.857e+04
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.9337      0.034    289.481      0.000       9.866      10.001
x1            -0.1888      0.219     -0.861      0.389      -0.618       0.241
x2             1.3258      0.222      5.972      0.000       0.891       1.761
x3             0.4946      0.065      7.639      0.000       0.368       0.622
x4             4.5343      0.359     12.622      0.000       3.830       5.239
```

```
x5            -4.4862      0.183    -24.552      0.000      -4.844      -4.128
x6            -1.0773      0.143     -7.538      0.000      -1.358      -0.797
x7             1.1937      0.158      7.545      0.000       0.884       1.504
==============================================================================
Omnibus:                      933.799   Durbin-Watson:                   1.387
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2602.745
Skew:                           1.174   Prob(JB):                         0.00
Kurtosis:                       6.072   Cond. No.                         30.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

MSE: 4.909236815818961
```

[37]:
```python
#Eliminar la variable con el mayor VIF
x_vars.remove("Whole_weight")
print(vif(x_vars).sort_values("VIF", ascending=False))
```

```
         variable        VIF
1        Diameter  41.819755
0          Length  40.763955
4  Viscera_weight  10.697780
3  Shucked_weight   8.852112
5    Shell_weight   7.817781
2          Height   3.558443
```

[38]:
```python
test_model(x_vars)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.510
Model:                            OLS   Adj. R-squared:                  0.509
Method:                 Least Squares   F-statistic:                     722.1
Date:                Fri, 20 Oct 2023   Prob (F-statistic):               0.00
Time:                        23:23:21   Log-Likelihood:                -9328.3
No. Observations:                4177   AIC:                         1.867e+04
Df Residuals:                    4170   BIC:                         1.871e+04
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.9337      0.035    284.137      0.000       9.865      10.002
x1            -0.2271      0.223     -1.018      0.309      -0.665       0.210
x2             1.3952      0.226      6.171      0.000       0.952       1.838
x3             0.5113      0.066      7.753      0.000       0.382       0.641
```

```
x4            -2.5735      0.104    -24.741      0.000      -2.777      -2.370
x5             0.0395      0.114      0.345      0.730      -0.185       0.264
x6             2.7816      0.098     28.456      0.000       2.590       2.973
==============================================================================
Omnibus:                     1037.149   Durbin-Watson:                   1.367
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3176.648
Skew:                           1.266   Prob(JB):                         0.00
Kurtosis:                       6.441   Cond. No.                         20.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

MSE: 5.0968383150592524
```

[39]:

```python
#Eliminar la variable con el mayor VIF
x_vars.remove("Diameter")
print(vif(x_vars).sort_values("VIF", ascending=False))
```

```
         variable        VIF
3  Viscera_weight  10.690504
2  Shucked_weight   8.851834
0          Length   8.013867
4    Shell_weight   7.457755
1          Height   3.509983
```

[40]:

```python
test_model(x_vars)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.505
Model:                            OLS   Adj. R-squared:                  0.505
Method:                 Least Squares   F-statistic:                     851.4
Date:                Fri, 20 Oct 2023   Prob (F-statistic):               0.00
Time:                        23:23:21   Log-Likelihood:                -9347.3
No. Observations:                4177   AIC:                         1.871e+04
Df Residuals:                    4171   BIC:                         1.874e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.9337      0.035    282.882      0.000       9.865      10.003
x1             1.0075      0.099     10.135      0.000       0.813       1.202
x2             0.5588      0.066      8.494      0.000       0.430       0.688
x3            -2.5699      0.104    -24.598      0.000      -2.775      -2.365
x4             0.0211      0.115      0.183      0.854      -0.204       0.246
```

```
x5             2.9111      0.096     30.356      0.000       2.723       3.099
==============================================================================
Omnibus:                     1040.962   Durbin-Watson:                   1.358
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3288.926
Skew:                           1.259   Prob(JB):                         0.00
Kurtosis:                       6.544   Cond. No.                         8.38
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

MSE: 5.143385827045999
```

[41]:
```
#Eliminar la variable con el mayor VIF
x_vars.remove("Viscera_weight")
print(vif(x_vars).sort_values("VIF", ascending=False))
```

```
         variable       VIF
0          Length  7.746817
3    Shell_weight  6.598972
2  Shucked_weight  6.114777
1          Height  3.489531
```

[42]:
```
test_model(x_vars)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.505
Model:                            OLS   Adj. R-squared:                  0.505
Method:                 Least Squares   F-statistic:                     1064.
Date:                Fri, 20 Oct 2023   Prob (F-statistic):               0.00
Time:                        23:23:22   Log-Likelihood:                -9347.3
No. Observations:                4177   AIC:                         1.870e+04
Df Residuals:                    4172   BIC:                         1.874e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.9337      0.035    282.915      0.000       9.865      10.003
x1             1.0109      0.098     10.344      0.000       0.819       1.202
x2             0.5598      0.066      8.534      0.000       0.431       0.688
x3            -2.5592      0.087    -29.476      0.000      -2.729      -2.389
x4             2.9170      0.090     32.341      0.000       2.740       3.094
==============================================================================
Omnibus:                     1040.521   Durbin-Watson:                   1.358
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3288.766
```

```
Skew:                           1.258   Prob(JB):                         0.00
Kurtosis:                       6.545   Cond. No.                         6.16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

MSE: 5.143427334731926
```

- **¿Cómo cambio el valor de R^2 del modelo? ¿A que se lo adjudica?**

El coeficiente de R^2 fue disminuyendo al reducir el número de variables que se tomaron en cuenta en el modelo. Esto se debe a que, al tener un número muy pequeño de variables, quitar una de ellas elimina una gran cantidad de información que le permite al modelo encontrar relaciones numéricas entre variables. Aunque la variable removida tenía un VIF mayor, y en teoría el modelo debería mejorar, cuando se tiene una cantidad pequeña de variables el tradeoff es peor para el coeficiente de R^2, ya que se eliminan relaciones que modelan a la variable predictora.

- **¿Como cambiaron los coeficientes?¿Qué se interpretación se puede obtener con los nuevos valores de coeficientes?**

Los coeficientes se fueron haciendo menores en las variables con valores grandes, y los valores pequeños incrementaron al eliminar las variables. Esto se debe a que eliminar variables reduce las relaciones entre ellas, lo que hace que haya números más pequeños que puedan representar las relaciones que hay entre las variables restantes sin necesidad de incrementar la magnitud de cada una de ellas. Otra cosa que se puede interpretar de esto, es que las variables que tenían mayor intervención en predecir, redujeron su impacto y las que tenían poca significancia tuvieron que incrementar para lograr predecir lo más adecuadamente posible la variable de respuesta.

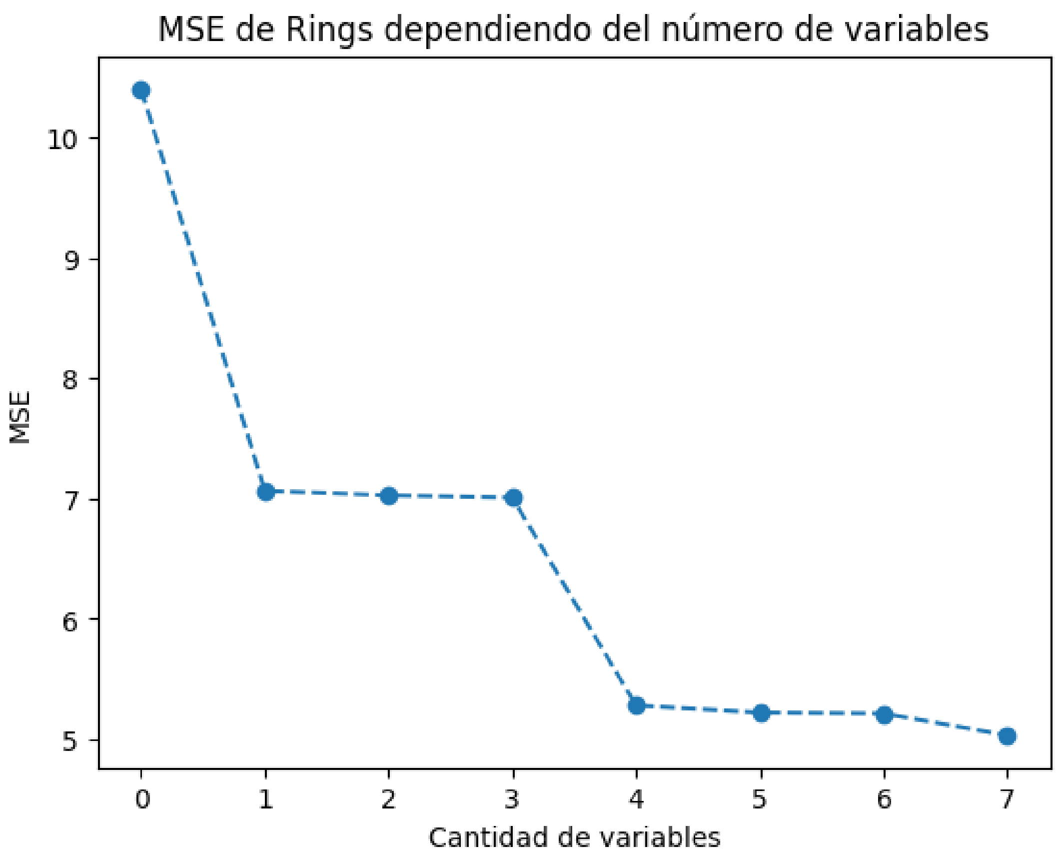
0.10 ## Análisis de Componentes Principales (PCA)

[43]:
```python
#Realizar el PCA
np.set_printoptions(suppress=True, precision=3)
pca = PCA()

x_reduced = pca.fit_transform(scale(x))
print("Varianza de cada dimensión:")
print(pca.explained_variance_)
print("Varianza de cada dimensión + la primera:")
print(pca.explained_variance_ratio_)
print("Varianza acumulada:")
print(pca.explained_variance_ratio_.cumsum())
```

```
Varianza de cada dimensión:
[6.357 0.279 0.167 0.114 0.065 0.013 0.007]
Varianza de cada dimensión + la primera:
[0.908 0.04  0.024 0.016 0.009 0.002 0.001]
Varianza acumulada:
[0.908 0.948 0.972 0.988 0.997 0.999 1.   ]
```

[44]:
```python
from sklearn import model_selection
#Evaluar el MSE con cada número de variables en base al PCA
n, c = x_reduced.shape
y_values = df["Rings"].values

#Implementar validación cruzada para evaluar el MSE
kf = KFold(n_splits=10, shuffle=True, random_state=15)
modelo = LinearRegression()
mse = []

puntaje = -1 * model_selection.cross_val_score(modelo, np.ones((n,1)), y_values.
 ↪ravel(), cv=kf, scoring="neg_mean_squared_error").mean()
mse.append(puntaje)

for i in np.arange(1, c+1):
  puntaje = -1 * model_selection.cross_val_score(modelo, x_reduced[:,:i],␣
 ↪y_values.ravel(), cv=kf, scoring="neg_mean_squared_error").mean()
  mse.append(puntaje)

ejex = np.arange(0,len(mse))
plt.plot(ejex,mse,"--o")
plt.xlabel("Cantidad de variables")
plt.ylabel("MSE")
plt.title("MSE de Rings dependiendo del número de variables")
plt.xticks(ejex)
```

[44]:
```
([<matplotlib.axis.XTick at 0x7c4c2e878a30>,
  <matplotlib.axis.XTick at 0x7c4c2e878a60>,
  <matplotlib.axis.XTick at 0x7c4c2e878be0>,
  <matplotlib.axis.XTick at 0x7c4c2e8a3010>,
  <matplotlib.axis.XTick at 0x7c4c2e878250>,
  <matplotlib.axis.XTick at 0x7c4c2e90b580>,
  <matplotlib.axis.XTick at 0x7c4c2e9087c0>,
  <matplotlib.axis.XTick at 0x7c4c2e908910>],
 [Text(0, 0, '0'),
  Text(1, 0, '1'),
  Text(2, 0, '2'),
  Text(3, 0, '3'),
  Text(4, 0, '4'),
  Text(5, 0, '5'),
  Text(6, 0, '6'),
  Text(7, 0, '7')])
```



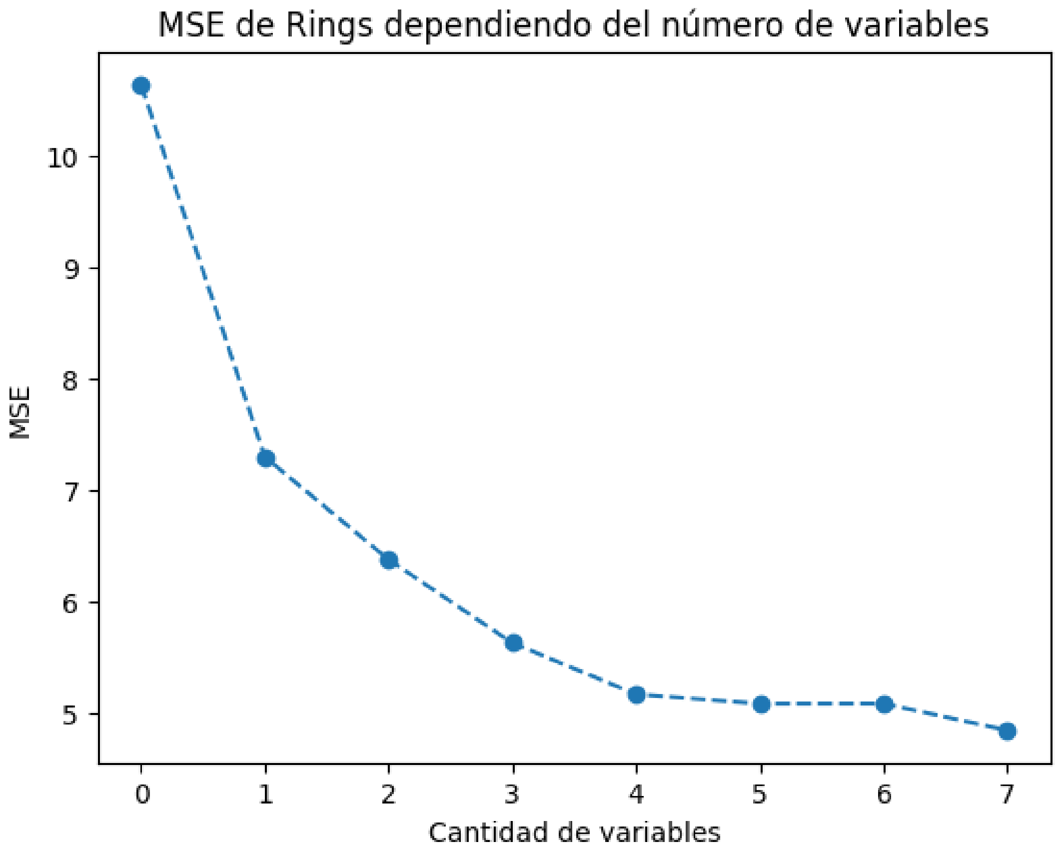
[45]:
```
#Obtener la varianza explicada de N número de variables usadas
print(np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100))
```

```
[ 90.79  94.78  97.17  98.8   99.72  99.9  100.  ]
```

[46]:
```
#Realizar el PCA con un set de datos de entrenamiento
pca2 = PCA()

xtrain, xtest, ytrain, ytest = model_selection.train_test_split(x,y, test_size␣
 ↪= 0.5, random_state=15)

x_reduced_train = pca2.fit_transform(scale(xtrain))
n, c = x_reduced_train.shape

kf2 = KFold(n_splits=10, shuffle=True, random_state=15)
model = LinearRegression()
mse = []

puntaje = -1 * model_selection.cross_val_score(modelo, np.ones((n,1)), ytrain.
 ↪ravel(), cv=kf2, scoring="neg_mean_squared_error").mean()
```

```python
mse.append(puntaje)

for i in np.arange(1, c+1):
  puntaje = -1 * model_selection.cross_val_score(modelo, x_reduced_train[:,:i],␣
↪ytrain.ravel(), cv=kf, scoring="neg_mean_squared_error").mean()
  mse.append(puntaje)

ejex = np.arange(0,len(mse))
plt.plot(ejex,mse,"--o")
plt.xlabel("Cantidad de variables")
plt.ylabel("MSE")
plt.title("MSE de Rings dependiendo del número de variables")
plt.xticks(ejex)
```

[46]:
```
([<matplotlib.axis.XTick at 0x7c4c2e9157b0>,
  <matplotlib.axis.XTick at 0x7c4c2e9143a0>,
  <matplotlib.axis.XTick at 0x7c4c2e9167d0>,
  <matplotlib.axis.XTick at 0x7c4c2e45f190>,
  <matplotlib.axis.XTick at 0x7c4c2e45fc40>,
  <matplotlib.axis.XTick at 0x7c4c2e879bd0>,
  <matplotlib.axis.XTick at 0x7c4c2e488c10>,
  <matplotlib.axis.XTick at 0x7c4c2e4896c0>],
 [Text(0, 0, '0'),
  Text(1, 0, '1'),
  Text(2, 0, '2'),
  Text(3, 0, '3'),
  Text(4, 0, '4'),
  Text(5, 0, '5'),
  Text(6, 0, '6'),
  Text(7, 0, '7')])
```



[47]:
```
#Obtener el MSE con los datos del conjunto de prueba

x_reduced_test = pca.transform(scale(xtest))[:,:5]
modelo = sm.OLS(ytrain, sm.add_constant(x_reduced_train[:,:5]))
modelo = modelo.fit()

ypred = modelo.predict(sm.add_constant(x_reduced_test))
mse = mean_squared_error(ytest,ypred)

print("MSE: {}".format(np.round(mse,3)))
print("R^2: {}".format(modelo.rsquared))
```

```
MSE: 12.843
R^2: 0.5245968633518544
```

- **¿Mejoró el valor de R^2 y MSE del modelo PCR respecto al metodo de VIF?¿A que se lo adjudica?**

El Mean Squared Error del modelo con el PCA incrementó con respecto al VIF, mientras que el coeficiente de R^2 aumentó. Esto se debe a que al reducir las variables predictoras, se pudo modelar de forma más precisa las relaciones que tienen entre ellas para modelar la variable de respuesta.

Sin embargo, al tomar en cuenta más variables predictoras, también hace que haya escenarios en donde las predicciones realizadas no puedan ser tan exactas. Otra ventaja que tiene el PCA con respecto al VIF es que determina automáticamente las N variables más significativas, haciendo que no se tenga que decidir de forma manual en base al puntaje del VIF.

En conclusión, se puede decir que el PCA es una mejor forma de encontrar el número de variables significativas para este conjunto de datos, y el aplicar este método de selección de características, junto con la división de los datos en conjuntos de entrenamiento y prueba generaron un modelo más adecuado que los que se habían conseguido anteriormente