Multicolinealidad

- Andrés Alejandro Guzmán González | A01633819
- Joel Isaías Solano Ocampo | A01639289
- Tania Sayuri Guizado Hernandez | A01640092
- Ernesto Reynoso Lizárraga | A01639915

In [ ]:
```
!pip install ucimlrepo
!pip install --upgrade scikit-learn
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-packages (0.0.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.3.
1)
Requirement already satisfied: numpy<2.0,>=1.17.3 in /usr/local/lib/python3.10/dist-packages
(from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from
scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from
scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-package
s (from scikit-learn) (3.2.0)
```

In [ ]:
```
from ucimlrepo import fetch_ucirepo
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats.mstats import winsorize
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
from sklearn.metrics import mean_squared_error
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [ ]:
```
abalone = fetch_ucirepo(id=1)
X = abalone.data.features
y = abalone.data.targets

print(abalone.metadata)
print(abalone.variables)
```

{'uci_id': 1, 'name': 'Abalone', 'repository_url': 'https://archive.ics.uci.edu/dataset/1/abalone', 'data_url': 'https://archive.ics.uci.edu/static/public/1/data.csv', 'abstract': 'Predict the age of abalone from physical measurements', 'area': 'Life Science', 'tasks': ['Classification', 'Regression'], 'characteristics': ['Tabular'], 'num_instances': 4177, 'num_features': 8, 'feature_types': ['Categorical', 'Integer', 'Real'], 'demographics': [], 'target_col': ['Rings'], 'index_col': None, 'has_missing_values': 'no', 'missing_values_symbol': None, 'year_of_dataset_creation': 1994, 'last_updated': 'Mon Aug 28 2023', 'dataset_doi': '10.24432/C55C7W', 'creators': ['Warwick Nash', 'Tracy Sellers', 'Simon Talbot', 'Andrew Cawthorn', 'Wes Ford'], 'intro_paper': None, 'additional_info': {'summary': 'Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.\r\n\r\nFrom the original data examples with missing values were removed (the majority having the predicted value missing), and the ranges of the continuous values have been scaled for use with an ANN (by dividing by 200).', 'purpose': None, 'funded_by': None, 'instances_represent': None, 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': 'Given is the attribute name, attribute type, the measurement unit and a brief description. The number of rings is the value to predict: either as a continuous value or as a classification problem.\r\n\r\nName / Data Type / Measurement Unit / Description\r\n-----------------------------\r\nSex / nominal / -- / M, F, and I (infant)\r\nLength / continuous / mm / Longest shell measurement\r\nDiameter\t/ continuous / mm / perpendicular to length\r\nHeight / continuous / mm / with meat in shell\r\nWhole weight / continuous / grams / whole abalone\r\nShucked weight / continuous\t / grams / weight of meat\r\nViscera weight / continuous / grams / gut weight (after bleeding)\r\nShell weight / continuous / grams / after being dried\r\nRings / integer / -- / +1.5 gives the age in years\r\n\r\nThe readme file contains attribute statistics.', 'citation': None}}

```
             name     role         type demographic  \
0             Sex  Feature  Categorical        None   
1          Length  Feature   Continuous        None   
2        Diameter  Feature   Continuous        None   
3          Height  Feature   Continuous        None   
4    Whole_weight  Feature   Continuous        None   
5  Shucked_weight  Feature   Continuous        None   
6  Viscera_weight  Feature   Continuous        None   
7    Shell_weight  Feature   Continuous        None   
8           Rings   Target      Integer        None   

                 description  units missing_values  
0        M, F, and I (infant)   None             no  
1    Longest shell measurement     mm             no  
2      perpendicular to length     mm             no  
3           with meat in shell     mm             no  
4               whole abalone  grams             no  
5              weight of meat  grams             no  
6  gut weight (after bleeding)  grams             no  
7           after being dried  grams             no  
8  +1.5 gives the age in years   None             no  
```

In [ ]:

```
target_url = 'https://archive.ics.uci.edu/static/public/1/data.csv'
abalone = pd.read_csv(target_url)
abalone.head()
```

Out[ ]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Rings
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

In [ ]:

```
abalone = abalone.drop('Sex', axis=1)
abalone.head()
```

Out[ ]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Rings
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [ ]:

```
X = abalone.loc[:,:'Shell_weight']
X
```

Out[ ]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550
...	...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950

4177 rows × 7 columns

In [ ]:

```
Y = abalone['Rings']
Y
```

Out[ ]:

```
0       15
1        7
2        9
3       10
4        7
        ..
4172    11
4173    10
4174     9
4175    10
4176    12
Name: Rings, Length: 4177, dtype: int64
```

In [ ]:

```
X_fit = sm.add_constant(X)
X_fit
```

Out[ ]:

	const	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight
0	1.0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	1.0	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	1.0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100
3	1.0	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550
4	1.0	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550
...	...	...	...	...	...	...	...	...
4172	1.0	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490
4173	1.0	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605
4174	1.0	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080
4175	1.0	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960
4176	1.0	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950

4177 rows × 8 columns

In [ ]:

```
X.shape
```

Out[ ]:

```
(4177, 7)
```

In [ ]:

```
model = sm.OLS(Y, X_fit)
fit_model = model.fit()
print("R^2 del modelo original:", fit_model.rsquared)
print("Parámetros del modelo original:")
print(fit_model.params)
```

```
R^2 del modelo original: 0.5276299399919839
Parámetros del modelo original:
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
```

In [ ]:

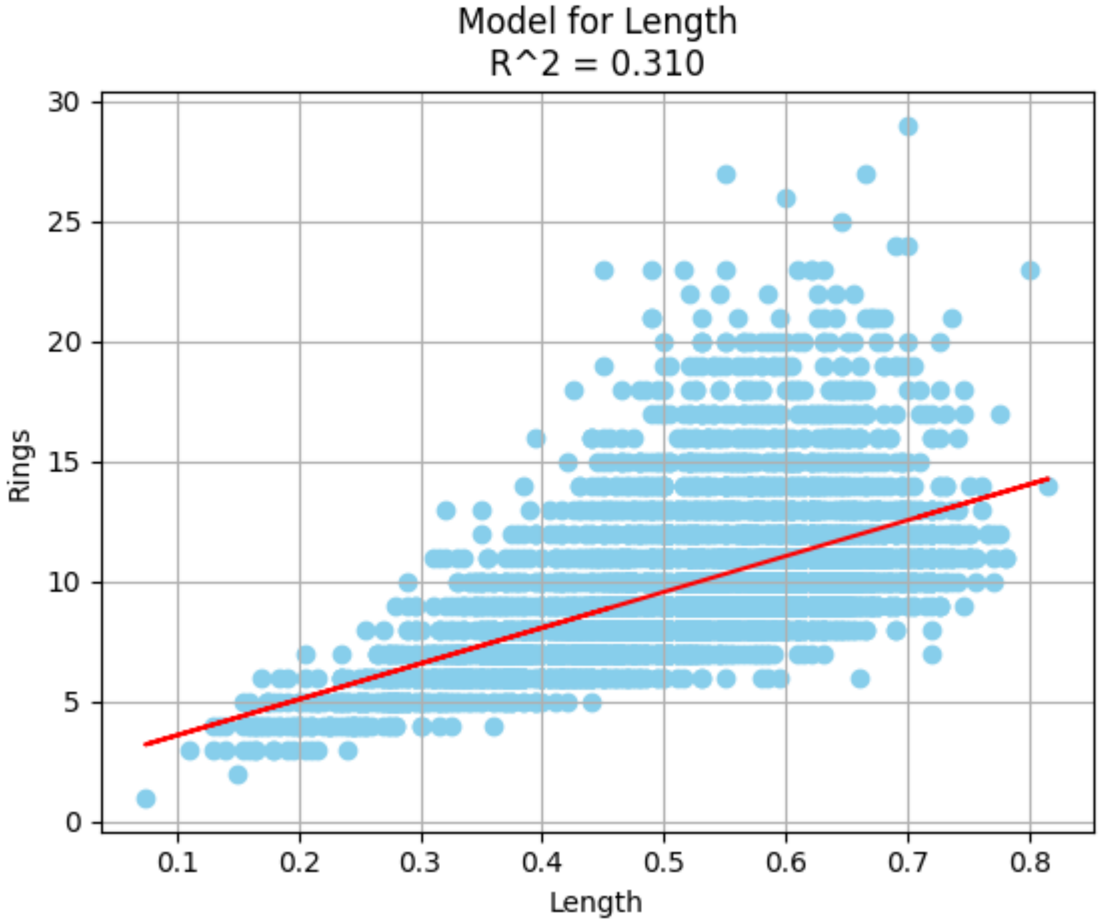
```
print(fit_model.summary())
```

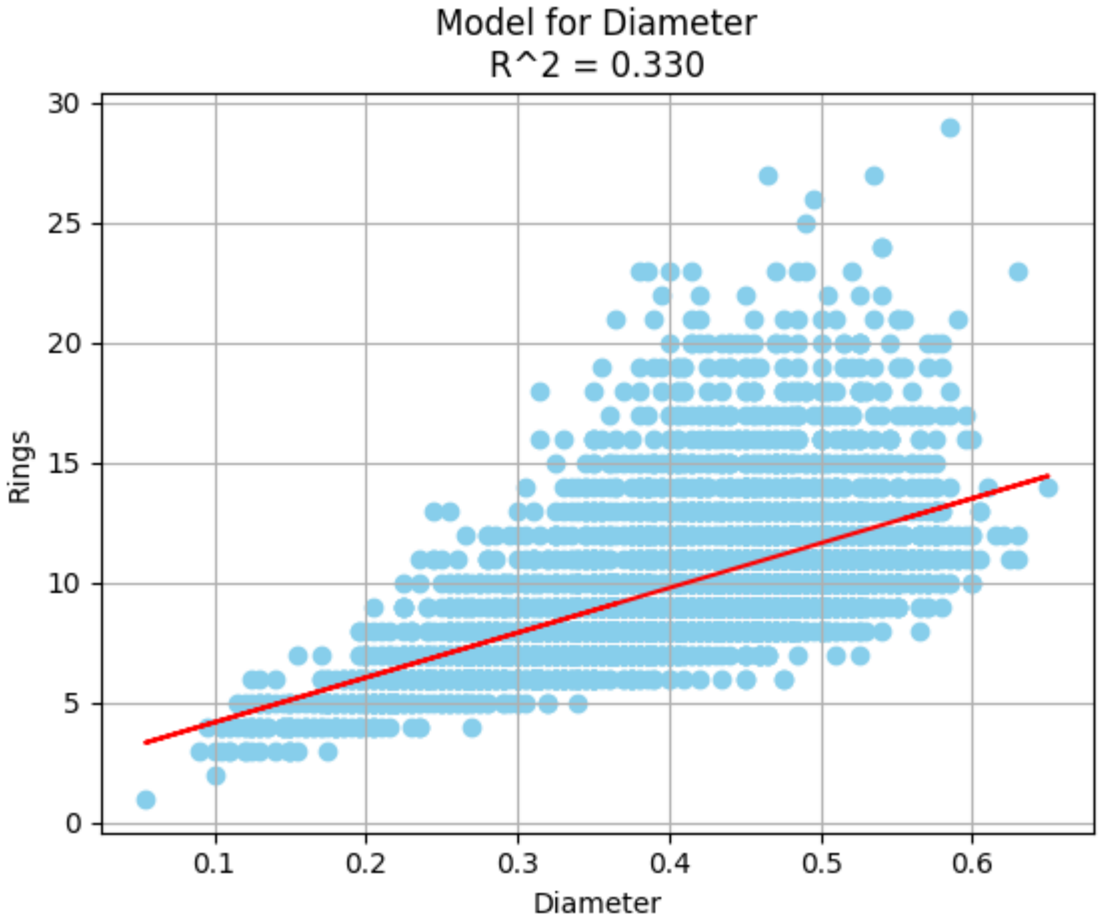
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.528
Model:                            OLS   Adj. R-squared:                  0.527
Method:                 Least Squares   F-statistic:                     665.2
Date:                Sat, 21 Oct 2023   Prob (F-statistic):               0.00
Time:                        00:24:05   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.
```

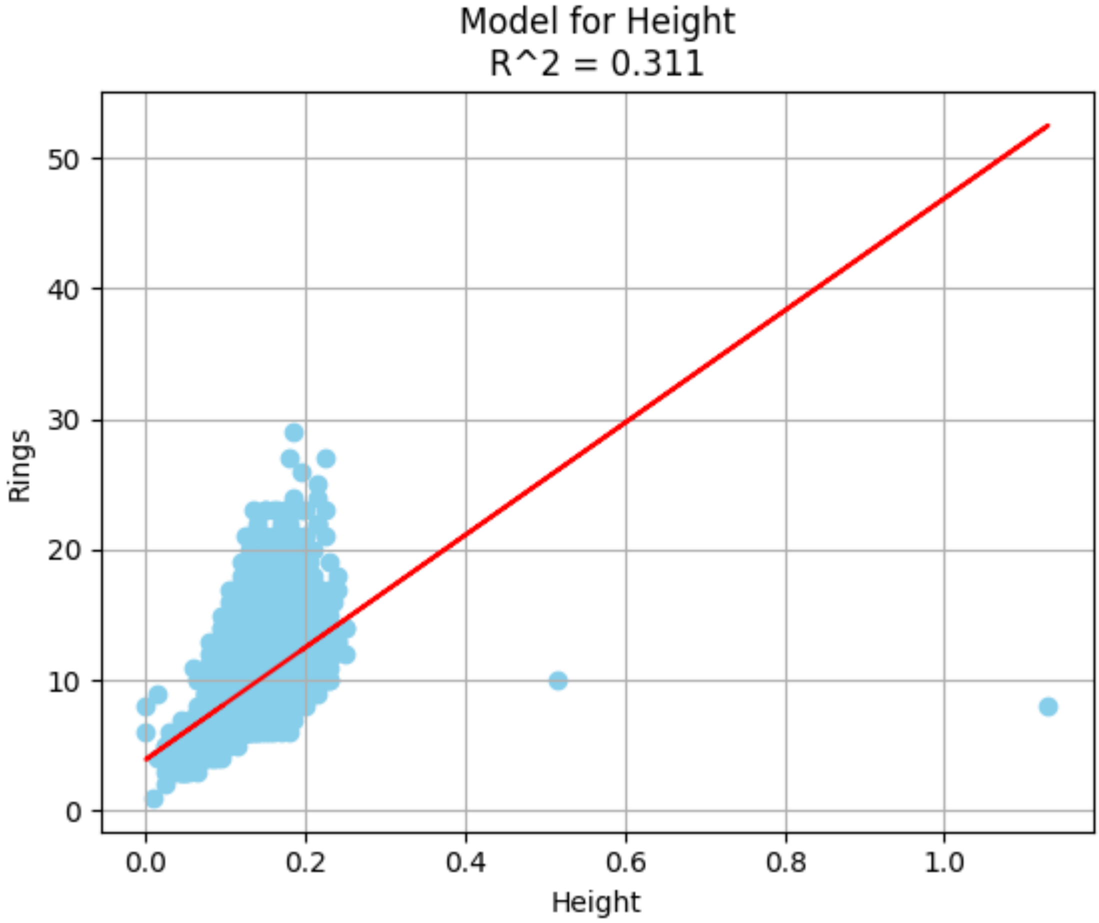
Busca una puntos "leverage"

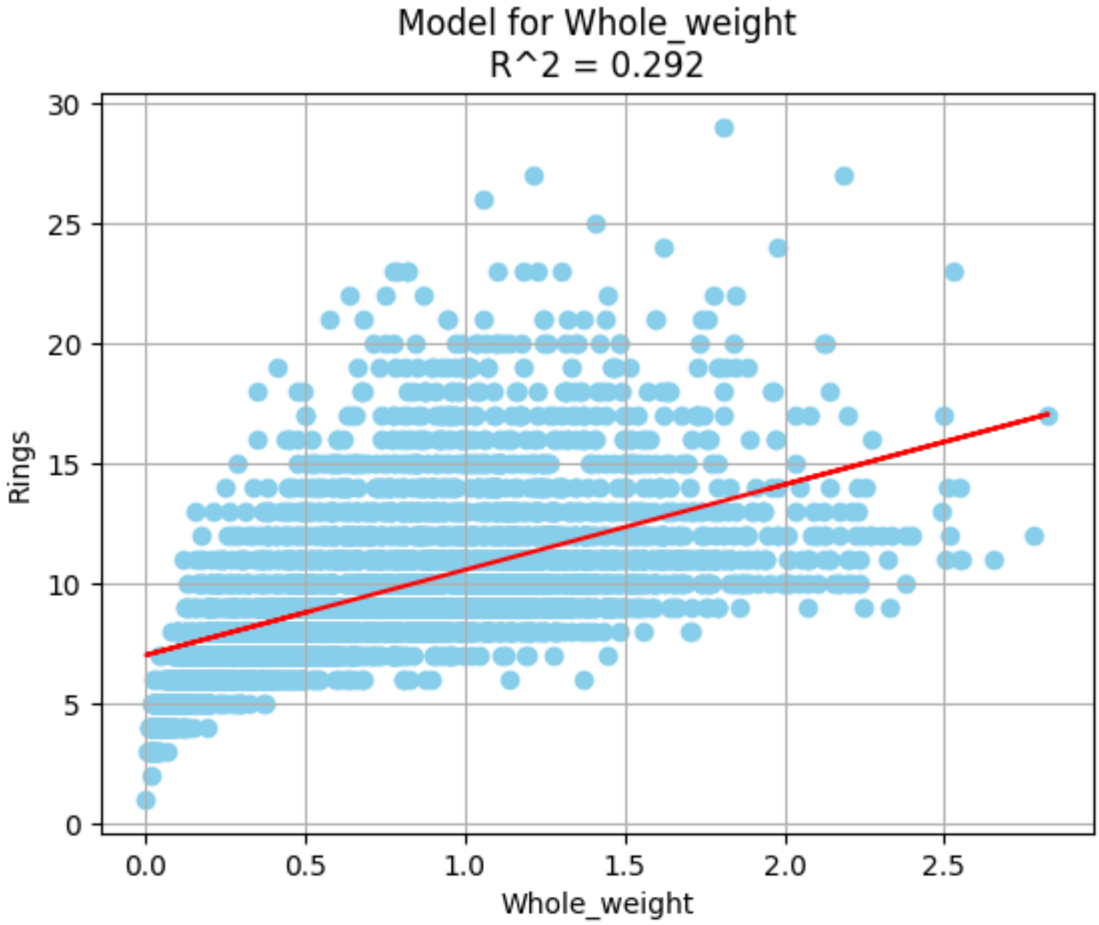
In [ ]:

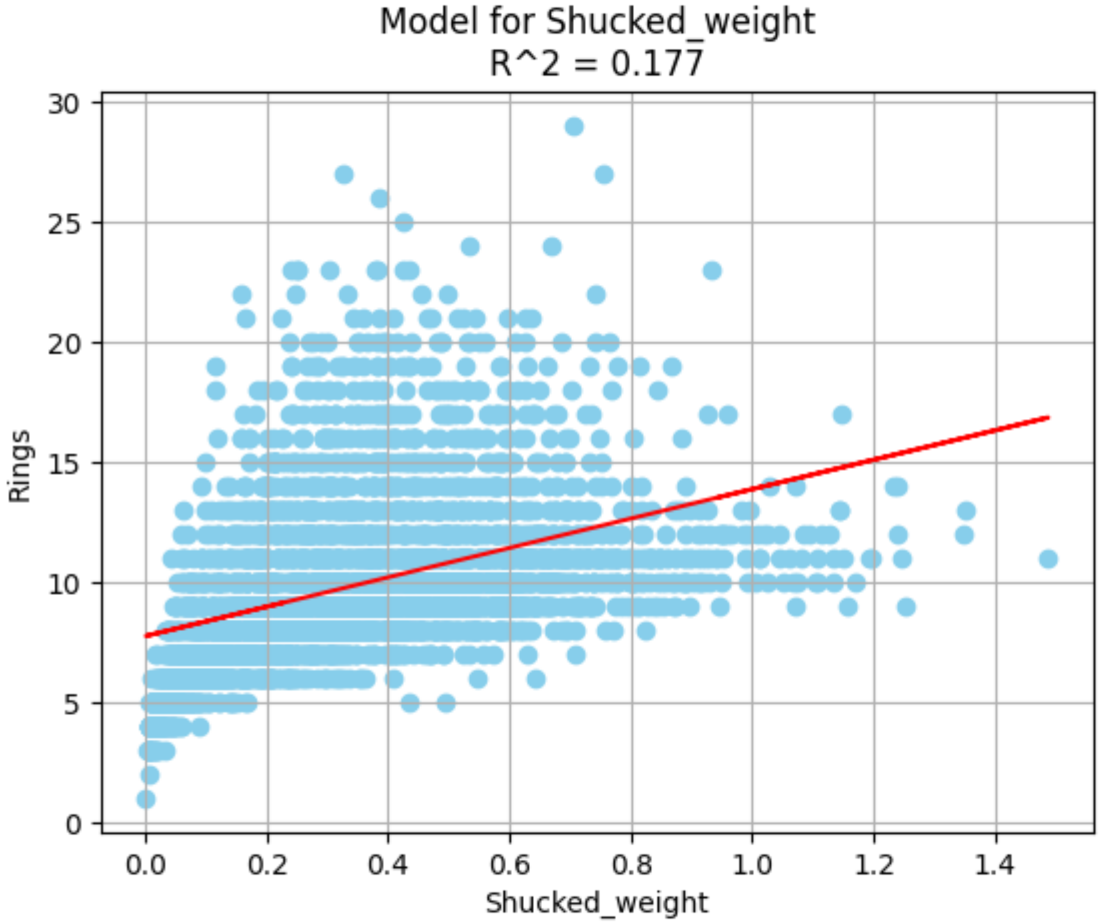
```
# A continuación, ajustar el modelo de mínimos cuadrados ordinarios (OLS) para cada variable i
for column in X.columns:
    X_temp = X[column]
    X_temp_fit = sm.add_constant(X_temp)
    model_temp = sm.OLS(Y, X_temp_fit)
    fit_model_temp = model_temp.fit()
    r_squared_temp = fit_model_temp.rsquared
    parameters_temp = fit_model_temp.params

    # Visualizar el ajuste del modelo
    plt.scatter(X_temp, Y, color='skyblue')
    plt.plot(X_temp, parameters_temp[1] * X_temp + parameters_temp[0], color='red')
    plt.title(f'Model for {column}\nR^2 = {r_squared_temp:.3f}')
    plt.xlabel(column)
    plt.ylabel('Rings')
    plt.grid()
    plt.show()
```

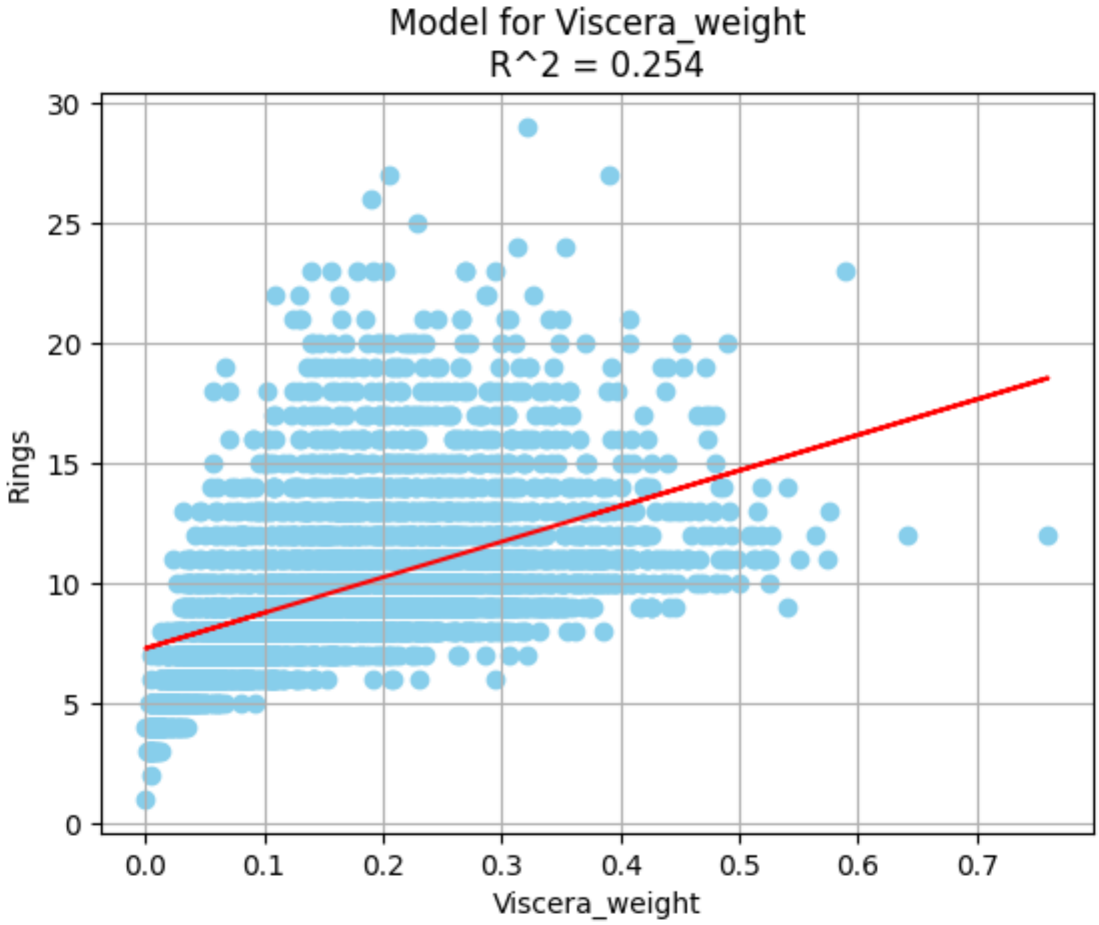

Model for Length
R^2 = 0.310
Rings
Length
30
25
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

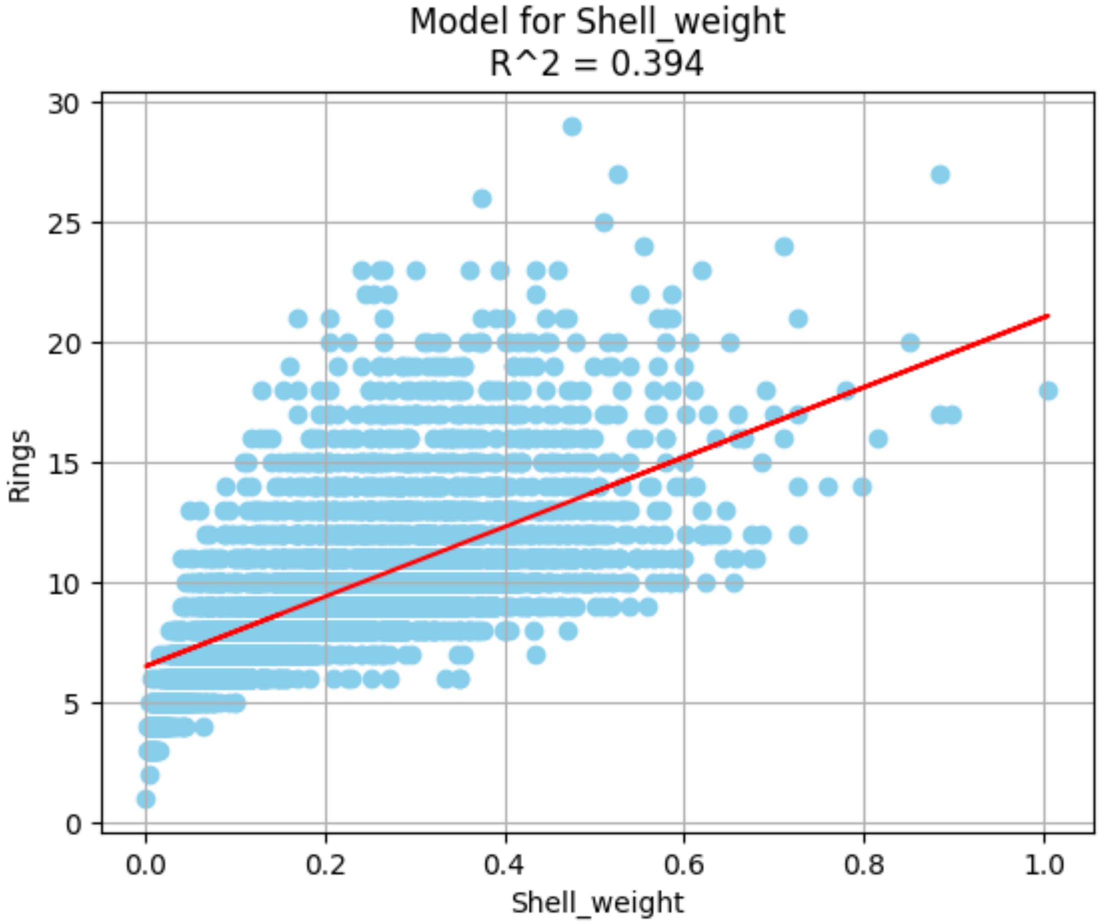

Model for Diameter
R^2 = 0.330
Rings
Diameter
30
25
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6


Model for Height
R^2 = 0.311
Rings
Height
0
10
20
30
40
50
0.0
0.2
0.4
0.6
0.8
1.0


Model for Whole_weight
R^2 = 0.292
Rings
Whole_weight
0
5
10
15
20
25
30
0.0
0.5
1.0
1.5
2.0
2.5

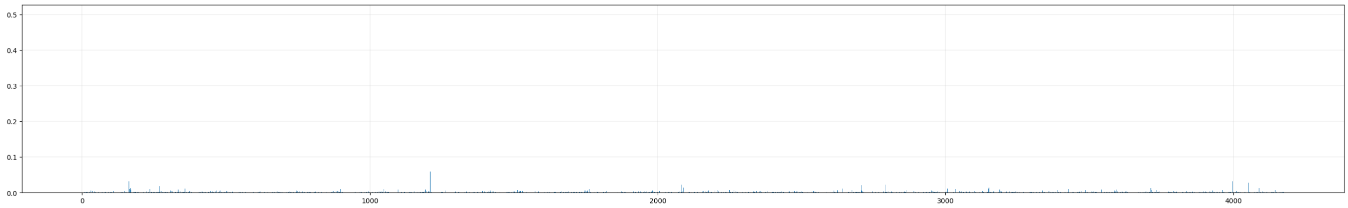

Model for Shucked_weight
R^2 = 0.177
Rings
Shucked_weight
0
5
10
15
20
25
30
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Model for Viscera_weight
R^2 = 0.254
Rings
Viscera_weight
0
5
10
15
20
25
30
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7



In [ ]:
```
influence = fit_model.get_influence()
H_diag = influence.hat_matrix_diag
print(H_diag)

plt.figure(figsize =(35, 5))
plt.bar(abalone.index, H_diag, width = 0.5)
#plt.xticks(abalone.index)
plt.grid(linewidth = 0.2)
```

```
[0.001 0.001 0.001 ... 0.002 0.001 0.003]
```



In [ ]:
```
mapping = sorted(list(enumerate(H_diag)), key=lambda item: item[1],reverse=True)
max_value_idxs = [item[0] for item in mapping]

print('Top leverage values')
print([item[1] for item in mapping][:3])

print('\nSample indexes with more leverage')
print(abalone.iloc[max_value_idxs])
```

```
Top leverage values
[0.5019723528421322, 0.05960859244317343, 0.05295671927323318]

Sample indexes with more leverage
      Length  Diameter  Height  Whole_weight  Shucked_weight  Viscera_weight  \
2051   0.455     0.355   1.130        0.5940          0.3320          0.1160   
1210   0.185     0.375   0.120        0.4645          0.1960          0.1045   
1417   0.705     0.565   0.515        2.2100          1.1075          0.4865   
3518   0.710     0.570   0.195        1.3480          0.8985          0.4435   
163    0.725     0.560   0.210        2.1410          0.6500          0.3980   
...      ...       ...     ...           ...             ...             ...   
837    0.475     0.365   0.125        0.5465          0.2290          0.1185   
600    0.535     0.420   0.145        0.9260          0.3980          0.1965   
3555   0.535     0.415   0.135        0.7800          0.3165          0.1690   
2744   0.480     0.375   0.120        0.5895          0.2535          0.1280   
488    0.540     0.420   0.135        0.8075          0.3485          0.1795   

      Shell_weight  Rings  
2051        0.1335      8  
1210        0.1500      6  
1417        0.5120     10  
3518        0.4535     11  
163         1.0050     18  
...            ...    ...  
837         0.1720      9  
600         0.2500     17  
3555        0.2365      8  
2744        0.1720     11  
488         0.2350     11  

[4177 rows x 8 columns]
```

Distancia Cook para detección de puntos influyentes

In [ ]:
```
np.set_printoptions(suppress=True)
cooks_dist = influence.cooks_distance[0]
print(cooks_dist[:10])
```

```
[0.001 0.    0.    0.    0.    0.    0.002 0.    0.    0.001]
```

In [ ]:
```
summary_cooks = influence.summary_frame()
print(summary_cooks[:10])
```

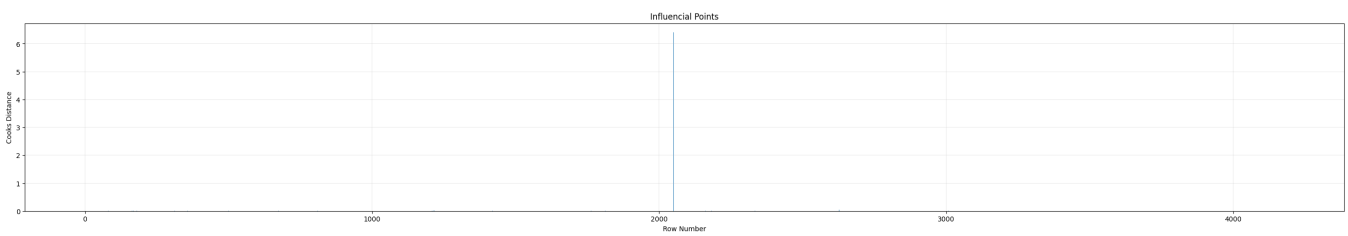
```
   dfb_const  dfb_Length  dfb_Diameter  dfb_Height  dfb_Whole_weight  \
0   0.013647   -0.032927      0.045769   -0.048025          0.010240   
1  -0.001956    0.000391      0.000189    0.000044          0.000010   
2   0.009360    0.001597     -0.007949    0.001964         -0.006124   
3   0.002439   -0.008192      0.008150    0.001118         -0.000097   
4   0.000109   -0.000049      0.000021   -0.000022          0.000027   
5  -0.000208   -0.001309      0.001323    0.000343          0.000119   
6  -0.006927    0.015866     -0.013475    0.004010         -0.007380   
7  -0.015747    0.012440      0.002541   -0.028781          0.005996   
8   0.000420    0.000070     -0.000485   -0.000221          0.000825   
9  -0.010038   -0.009157      0.017295   -0.004262          0.034385   

   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   
5            0.000060            0.000003         -0.000339  4.193532e-07   
6           -0.025396           -0.029626          0.068886  2.232357e-03   
7           -0.015638           -0.020015          0.011382  4.914173e-04   
8           -0.000304           -0.000163         -0.000499  1.039345e-06   
9           -0.039006           -0.061996          0.012198  1.355366e-03   

   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  
5       -0.052563  0.001213        -0.001832      -0.052556 -0.001831  
6        3.019824  0.001955         0.133637       3.022770  0.133767  
7        2.163169  0.000839         0.062700       2.164124  0.062728  
8       -0.137458  0.000440        -0.002884      -0.137442 -0.002883  
9        2.730818  0.001452         0.104129       2.732936  0.104210  
```

In [ ]:

```
summary_cooks = influence.summary_frame()
print(summary_cooks)
```

```
      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]
```

In [ ]:
```python
plt.figure(figsize = (35,5))
plt.bar(abalone.index, cooks_dist, width = 0.5)
#plt.xticks(abalone.index);
plt.xlabel('Row Number')
plt.ylabel('Cooks Distance')
plt.title('Influencial Points')
plt.grid(linewidth = 0.2)
```



In [ ]:
```python
mapping = sorted(list(enumerate(cooks_dist)), key=lambda item: item[1], reverse=True)
max_value_idxs = [item[0] for item in mapping]

print("Top cook's distance values:")
print([item[1] for item in mapping][:3])

print('Top Sample indexes with more distance values:')
print(abalone.iloc[max_value_idxs])
```

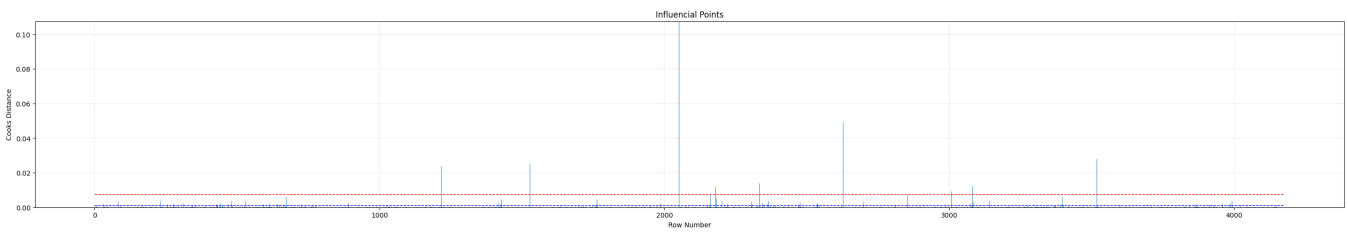
```
Top cook's distance values:
[6.409299513058319, 0.04919146658760428, 0.031621158221939866]
Top Sample indexes with more distance values:
      Length  Diameter  Height  Whole_weight  Shucked_weight  Viscera_weight  \
2051   0.455     0.355   1.130        0.5940          0.3320          0.1160
2627   0.275     0.205   0.070        0.1055          0.4950          0.0190
480    0.700     0.585   0.185        1.8075          0.7055          0.3215
3518   0.710     0.570   0.195        1.3480          0.8985          0.4435
1528   0.725     0.575   0.240        2.2100          1.3510          0.4130
...      ...       ...     ...           ...             ...             ...
2522   0.545     0.450   0.150        0.8795          0.3870          0.1500
2369   0.560     0.440   0.170        0.9445          0.3545          0.2175
1272   0.475     0.355   0.100        0.5035          0.2535          0.0910
1022   0.640     0.500   0.170        1.5175          0.6930          0.3260
897    0.265     0.195   0.060        0.0920          0.0345          0.0250

      Shell_weight  Rings
2051        0.1335      8
2627        0.0315      5
480         0.4750     29
3518        0.4535     11
1528        0.5015     13
...            ...    ...
2522        0.2625     11
2369        0.3000     12
1272        0.1400      8
1022        0.4090     11
897         0.0245      6

[4177 rows x 8 columns]
```

In [ ]:
```python
mean_cooks = np.mean(cooks_dist)
mean_cooks
```

Out[ ]:
```
0.0018730877579285728
```

In [ ]:
```python
mean_cooks_list = [4*mean_cooks for _ in abalone.index]
cooks_threshold = [4/len(cooks_dist) for _ in abalone.index]
```

In [ ]:
```python
plt.figure(figsize = (35,5))
plt.bar(abalone.index, cooks_dist, width=0.5)
plt.plot(abalone.index, mean_cooks_list, color="red", linestyle='--', linewidth=1)
plt.plot(abalone.index, cooks_threshold, color="blue", linestyle='--', linewidth=1)

#plt.xticks(abalone.index);
plt.xlabel('Row Number')
plt.ylabel('Cooks Distance')
plt.title('Influencial Points')
plt.ylim(top=max(mean_cooks_list + cooks_threshold) + 1e-1)
plt.grid(linewidth=0.2)
```



In [ ]:
```python
influencial_points = abalone.index[cooks_dist > 4/len(cooks_dist)]
print(influencial_points[:10])
abalone.iloc[influencial_points,:].head(10)
```

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

Out[ ]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Rings
6	0.530	0.415	0.150	0.7775	0.2370	0.1415	0.330	20
9	0.550	0.440	0.150	0.8945	0.3145	0.1510	0.320	19
32	0.665	0.525	0.165	1.3380	0.5515	0.3575	0.350	18
33	0.680	0.550	0.175	1.7980	0.8150	0.3925	0.455	19
36	0.540	0.475	0.155	1.2170	0.5305	0.3075	0.340	16
67	0.595	0.495	0.185	1.2850	0.4160	0.2240	0.485	13
72	0.595	0.475	0.170	1.2470	0.4800	0.2250	0.425	20
81	0.620	0.510	0.175	1.6150	0.5105	0.1920	0.675	12
83	0.595	0.475	0.160	1.3175	0.4080	0.2340	0.580	21
85	0.570	0.465	0.180	1.2950	0.3390	0.2225	0.440	12

In [ ]:

```
noninfluencial_point = abalone.index[cooks_dist < 4/len(cooks_dist)]
print(noninfluencial_point[:10])
abalone.iloc[noninfluencial_point,:].head(10)
```

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

Out[ ]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Rings
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7
5	0.425	0.300	0.095	0.3515	0.1410	0.0775	0.120	8
7	0.545	0.425	0.125	0.7680	0.2940	0.1495	0.260	16
8	0.475	0.370	0.125	0.5095	0.2165	0.1125	0.165	9
10	0.525	0.380	0.140	0.6065	0.1940	0.1475	0.210	14
11	0.430	0.350	0.110	0.4060	0.1675	0.0810	0.135	10

In [ ]:

```
for i in X_fit:
  winsorize(X_fit[i],limits=[0.25,0.25])
```

Calcula el nuevo valor de $\boldsymbol{R^2}$ y los parámetros obtenidos, comparar.

In [ ]:

```
model_transformed = sm.OLS(Y, X_fit)
fit_model_transformed = model_transformed.fit()
print(fit_model_transformed.summary())
```

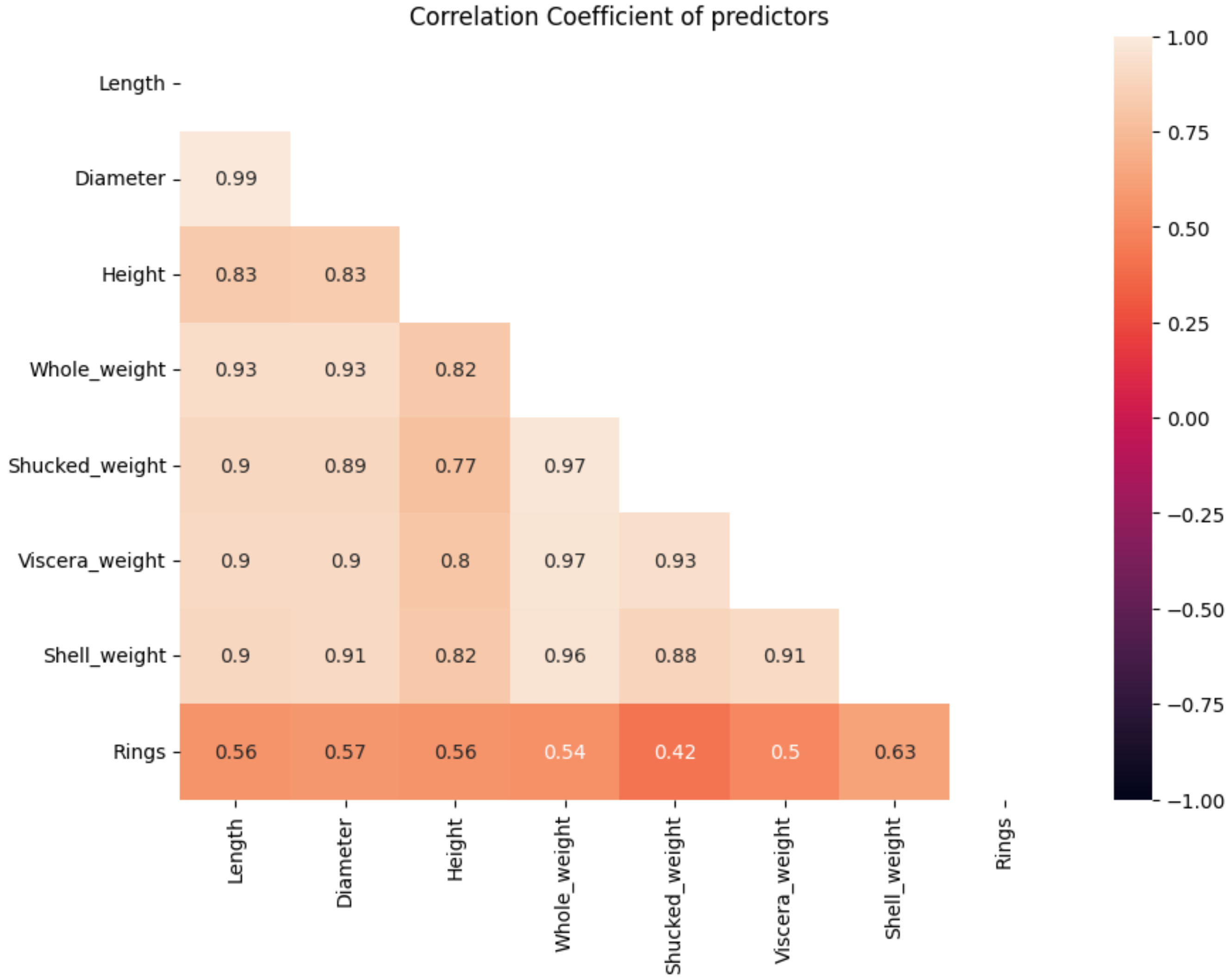
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.528
Model:                            OLS   Adj. R-squared:                  0.527
Method:                 Least Squares   F-statistic:                     665.2
Date:                Sat, 21 Oct 2023   Prob (F-statistic):               0.00
Time:                        00:24:43   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.
```

In [ ]:
```
r_squared_transformed = fit_model_transformed.rsquared
parameters_transformed = fit_model_transformed.params

print("\nR^2 del modelo transformado:", r_squared_transformed)
print("Parámetros del modelo transformado:")
print(parameters_transformed)
```

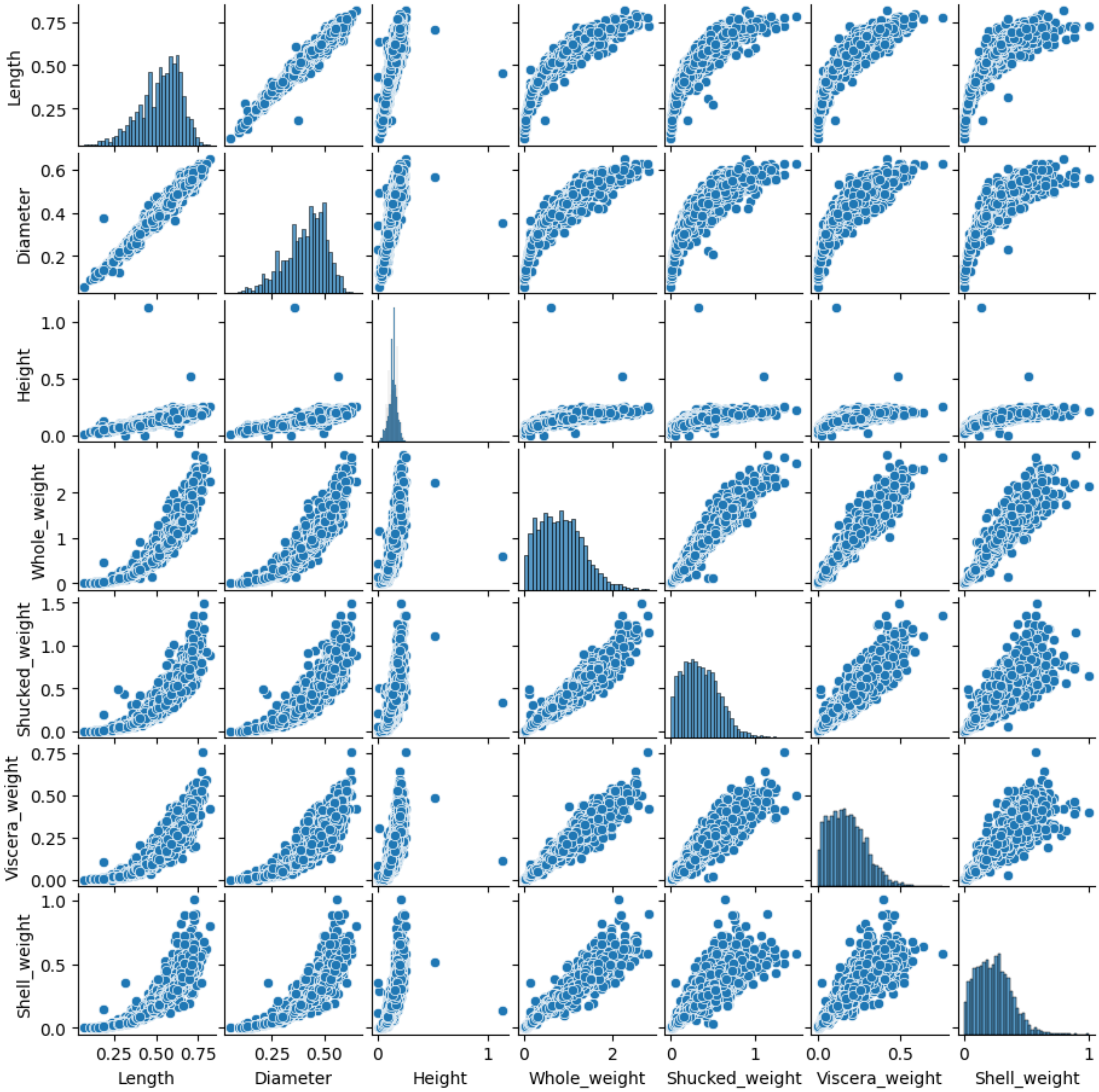
```
R^2 del modelo transformado: 0.5276299399919839
Parámetros del modelo transformado:
const              2.985154
Length            -1.571897
Diameter          13.360916
Height            11.826072
Whole_weight       9.247414
Shucked_weight   -20.213913
Viscera_weight    -9.829675
Shell_weight       8.576242
dtype: float64
```

Busca multicolinealidad en los datos usando VIF

In [ ]:
```
plt.figure(figsize=(10,7))
mask = np.triu(np.ones_like(abalone.corr(),dtype=bool))

sns.heatmap(abalone.corr(), annot=True, mask=mask,vmin=-1,vmax=1)
plt.title('Correlation Coefficient of predictors')
plt.show()
```



In [ ]:
```
Fig = sns.pairplot(X)
Fig.fig.set_size_inches(9,9)
```



In [ ]:
```python
def compute_vif(considered_features):
  X = abalone[considered_features].copy()
  X['intercept'] = 1
  vif_data = pd.DataFrame()
  vif_data['feature'] = X.columns
  vif_data['VIF'] = [variance_inflation_factor(X.values,i)
                     for i in range(len(X.columns))]
  vif_data = vif_data[vif_data['feature'] != 'intercept']
  return vif_data
```

In [ ]:
```python
considered_features = ['Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight','Visce
print(compute_vif(considered_features).sort_values('VIF',ascending=False))
```

```
          feature         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
```

Analiza y elimina variables independientes que indiquen que hay multicolinealidad

In [ ]:
```
considered_features.remove('Whole_weight')
print(compute_vif(considered_features).sort_values('VIF',ascending=False))
```

```
          feature        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
```

In [ ]:
```
X = abalone[considered_features]
model = sm.OLS(Y,sm.add_constant(X))
fit_model = model.fit()
print(fit_model.summary())
```

```
                            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:                Sat, 21 Oct 2023   Prob (F-statistic):               0.00
Time:                        00:25:02   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              2.8131      0.274     10.273      0.000       2.276       3.350
Length            -1.8916      1.859     -1.018      0.309      -5.536       1.753
Diameter          14.0606      2.278      6.171      0.000       9.594      18.528
Height            12.2266      1.577      7.753      0.000       9.135      15.318
Shucked_weight   -11.5957      0.469    -24.741      0.000     -12.515     -10.677
Viscera_weight     0.3601      1.043      0.345      0.730      -1.685       2.406
Shell_weight      19.9848      0.702     28.456      0.000      18.608      21.362
==============================================================================
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.                         108.
==============================================================================

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

In [ ]:
```
features_alt = [i for i in considered_features if i != 'Viscera_weight']
print(compute_vif(features_alt).sort_values('VIF',ascending=False))
```

```
          feature        VIF
1        Diameter  41.791313
0          Length  40.320617
4    Shell_weight   6.930345
3  Shucked_weight   6.115478
2          Height   3.536331
```

In [ ]:
```
X = abalone[features_alt]
model = sm.OLS(Y,sm.add_constant(X))
fit_model = model.fit()
print(fit_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.510
Model:                            OLS   Adj. R-squared:                  0.509
Method:                 Least Squares   F-statistic:                     866.7
Date:                Sat, 21 Oct 2023   Prob (F-statistic):               0.00
Time:                        00:25:02   Log-Likelihood:                -9328.3
No. Observations:                4177   AIC:                         1.867e+04
Df Residuals:                    4171   BIC:                         1.871e+04
Df Model:                           5
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const              2.7936      0.268     10.427      0.000       2.268       3.319
Length            -1.8247      1.849     -0.987      0.324      -5.449       1.799
Diameter          14.0401      2.277      6.165      0.000       9.575      18.505
Height            12.2695      1.572      7.806      0.000       9.188      15.351
Shucked_weight   -11.5057      0.390    -29.539      0.000     -12.269     -10.742
Shell_weight      20.0665      0.661     30.350      0.000      18.770      21.363
==============================================================================
Omnibus:                     1036.225   Durbin-Watson:                   1.366
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3175.198
Skew:                           1.265   Prob(JB):                         0.00
Kurtosis:                       6.442   Cond. No.                         106.
==============================================================================

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

In [ ]:

```
features_alt = [i for i in considered_features if i != 'Length']
print(compute_vif(features_alt).sort_values('VIF',ascending=False))
```

```
          feature        VIF
3  Viscera_weight  10.581434
2  Shucked_weight   8.716730
0        Diameter   8.221429
4    Shell_weight   7.780973
1          Height   3.555879
```

In [ ]:

```
X = abalone[features_alt]
model = sm.OLS(Y,sm.add_constant(X))
fit_model = model.fit()
print(fit_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.509
Model:                            OLS   Adj. R-squared:                  0.509
Method:                 Least Squares   F-statistic:                     866.4
Date:                Sat, 21 Oct 2023   Prob (F-statistic):               0.00
Time:                        00:25:02   Log-Likelihood:                -9328.8
No. Observations:                4177   AIC:                         1.867e+04
Df Residuals:                    4171   BIC:                         1.871e+04
Df Model:                           5
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const              2.7051      0.252     10.717      0.000       2.210       3.200
Diameter          11.9824      1.010     11.861      0.000      10.002      13.963
Height            12.1835      1.576      7.729      0.000       9.093      15.274
Shucked_weight   -11.6546      0.465    -25.059      0.000     -12.566     -10.743
Viscera_weight     0.2494      1.038      0.240      0.810      -1.785       2.284
Shell_weight      20.0338      0.701     28.593      0.000      18.660      21.407
==============================================================================
Omnibus:                     1040.749   Durbin-Watson:                   1.365
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3190.463
Skew:                           1.270   Prob(JB):                         0.00
Kurtosis:                       6.447   Cond. No.                         55.7
==============================================================================

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

Calcular el valor de MSE.

In [ ]:

```python
predictions = fit_model.predict(sm.add_constant(X))
mse = mean_squared_error(Y, predictions)

print("Mean Squared Error (MSE):", mse)
```

```
Mean Squared Error (MSE): 5.098104020582973
```

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

La disminución en el valor de R^2 en comparación con el modelo inicial podría deberse a la eliminación de ciertas características que poseen una mayor influencia en el modelo. Esta eliminación, a su vez, afecta negativamente el rendimiento del modelo.

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

De las características que se mantuvieron, todas experimentaron modificaciones en sus coeficientes, con variaciones más pronunciadas en algunas que en otras. Esta variación puede explicarse por el hecho de que al eliminar ciertas variables, se redistribuyen los pesos entre las variables restantes con el propósito de lograr un equilibrio en el modelo.

Analiza y determina el numero de componentes principales suficientes para mantener la cantidad de información justa necesaria.

In [ ]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
pca = PCA(n_components=2).fit(X)

print('Components with maximum variance')
for i, c in enumerate(pca.components_):
  print(f'Component {i} = {c}')

print('\nPercentage of variance explained by each of the selected components:')
for i, m in enumerate(pca.explained_variance_):
  print(f'Component magnitude {i} = {m}')
```

```
Components with maximum variance
Component 0 = [0.317 0.117 0.746 0.36  0.447]
Component 1 = [ 0.28   0.151 -0.616  0.091  0.715]

Percentage of variance explained by each of the selected components:
Component magnitude 0 = 0.08607826421650328
Component magnitude 1 = 0.003551610346868044
```

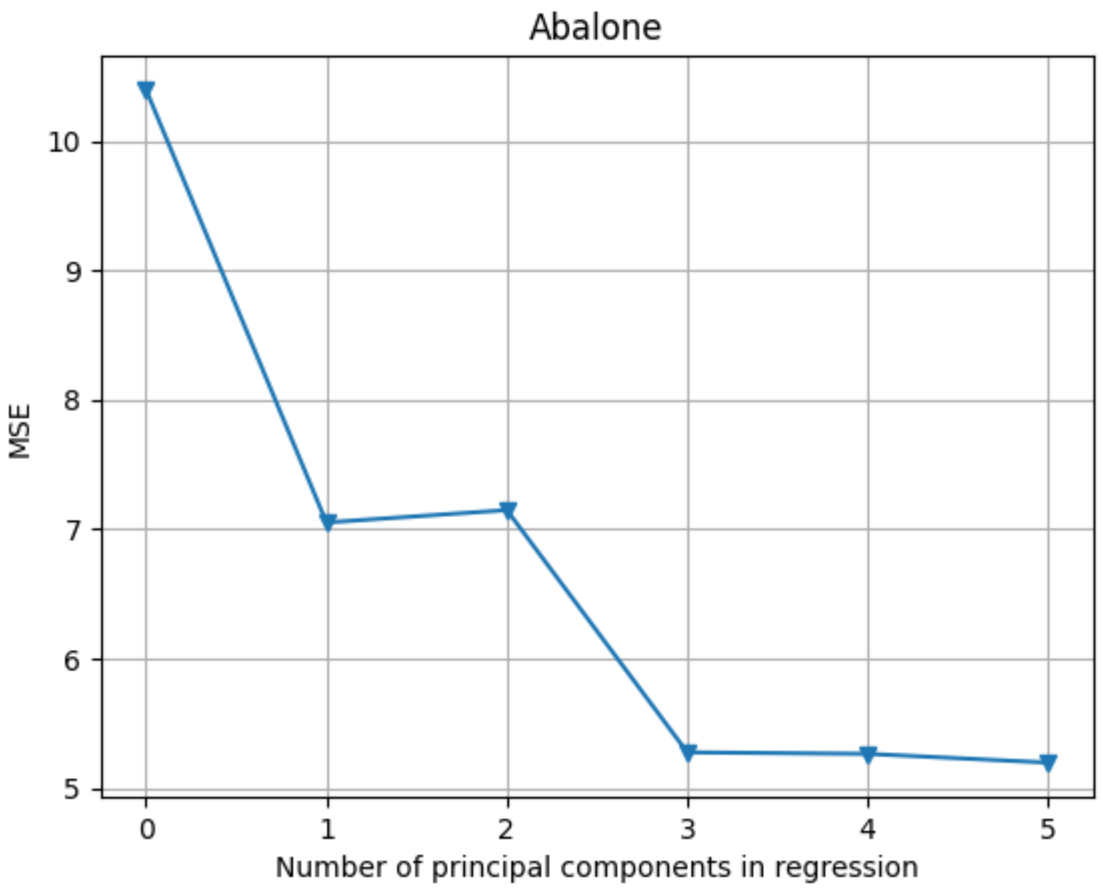
In [ ]:
```
np.set_printoptions(suppress=True, precision=3)
pca = PCA()

X_reduced = pca.fit_transform(scale(X))

print('Returs a vector of the variance explained by each dimension')
print(pca.explained_variance_)
print('\nGives the variance explained solely by the i+1st dimension')
print(pca.explained_variance_ratio_)
print('\nReturn a vector x such that x[i] return the cumulative variance explained by the firs
print(pca.explained_variance_ratio_.cumsum())
```

```
Returs a vector of the variance explained by each dimension
[4.462 0.263 0.117 0.096 0.063]

Gives the variance explained solely by the i+1st dimension
[0.892 0.053 0.023 0.019 0.013]

Return a vector x such that x[i] return the cumulative variance explained by the first i+1 di
mensions
[0.892 0.945 0.968 0.987 1.   ]
```

Obtenga de nuevo los valores de R^2 y MSE de esta aproximación.

In [ ]:
```
n, pc =  X_reduced.shape

kf_10 = model_selection.KFold(n_splits=10,shuffle=True, random_state=1)
model = LinearRegression()
mse = []

score = -1*model_selection.cross_val_score(model, np.ones((n, 1)), Y.ravel(),
                                           cv=kf_10,
                                           scoring='neg_mean_squared_error').mean()

mse.append(score)

for i in np.arange(1,pc+1):
    score = -1*model_selection.cross_val_score(model, X_reduced[:,:i], Y.ravel(),
                                           cv=kf_10,
                                           scoring='neg_mean_squared_error').mean()
    mse.append(score)

x_axis = np.arange(0,len(mse))
```

In [ ]:
```
plt.plot(x_axis,mse,'-v')
plt.xlabel('Number of principal components in regression')
plt.ylabel('MSE')
plt.title('Abalone')
plt.grid()
# plt.xticks(x_axis);
```

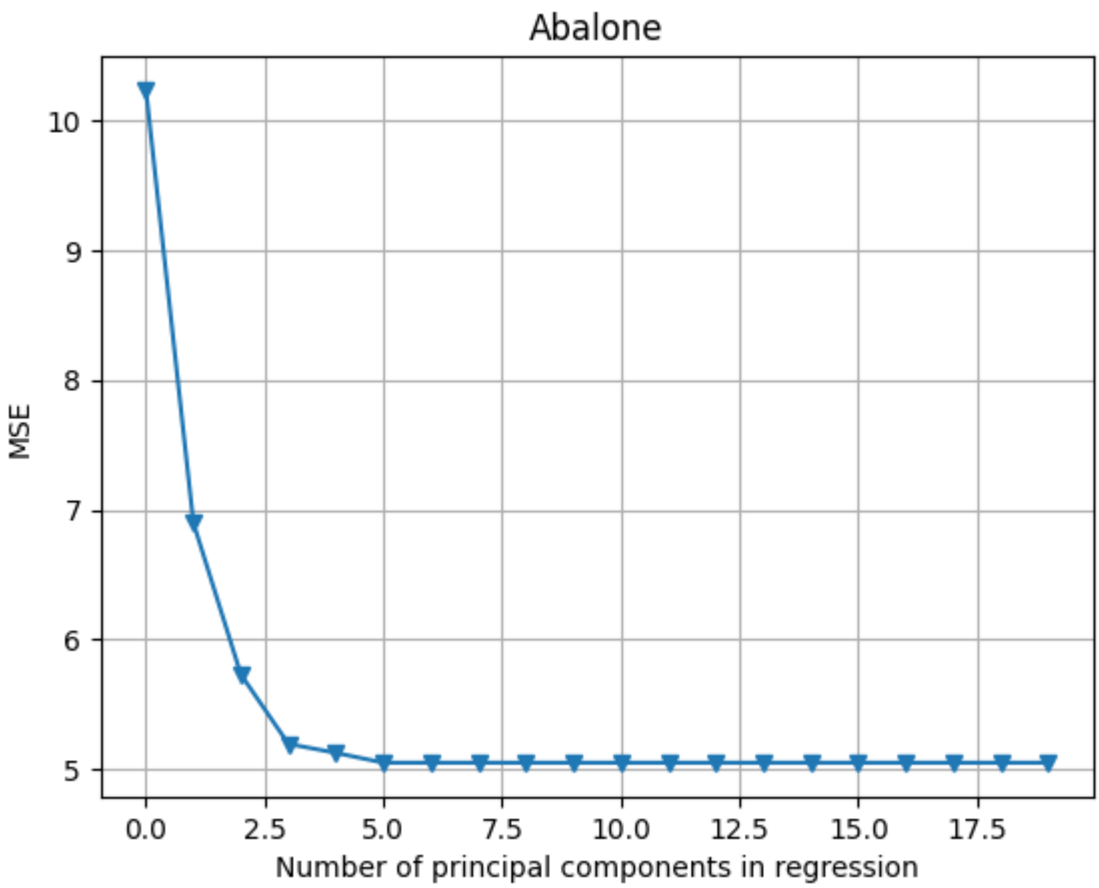


In [ ]:

```
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
pca2 = PCA()

X_train, X_test, y_train, y_test = model_selection.train_test_split(X,Y, test_size=0.5,random_
X_reduced_train = pca2.fit_transform(scale(X_train))
n, pc =  X_reduced_train.shape

kf_10 = model_selection.KFold(n_splits=10,shuffle=True, random_state=1)
model = LinearRegression()
mse = []

score = -1*model_selection.cross_val_score(model, np.ones((n,1)),y_train.ravel(),
                                           cv=kf_10,
                                           scoring='neg_mean_squared_error').mean()

mse.append(score)

for i in np.arange(1,20):
    score = -1*model_selection.cross_val_score(model,X_reduced_train[:,:i],y_train.ravel(),
                                               cv=kf_10,
                                               scoring='neg_mean_squared_error').mean()
    mse.append(score)

x_axis = np.arange(0,len(mse))
```

In [ ]:

```
plt.plot(x_axis,mse,'-v')
plt.xlabel('Number of principal components in regression')
plt.ylabel('MSE')
plt.title('Abalone')
plt.grid()
# plt.xticks(x_axis);
```



In [ ]:
```
X_reduced_TEST = pca2.transform(scale(X_test))[:,:5]

model =sm.OLS(y_train, sm.add_constant(X_reduced_train[:,:5]))
fit_model = model.fit()

pred = fit_model.predict(sm.add_constant(X_reduced_TEST))
mse = mean_squared_error(y_test,pred)

print('Mean squared error: {}'.format(np.round(mse,2)))
print(fit_model.summary())
```

```
Mean squared error: 5.27
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Rings   R-squared:                       0.510
Model:                            OLS   Adj. R-squared:                  0.508
Method:                 Least Squares   F-statistic:                     432.6
Date:                Sat, 21 Oct 2023   Prob (F-statistic):          7.70e-319
Time:                        00:29:41   Log-Likelihood:                -4646.4
No. Observations:                2088   AIC:                             9305.
Df Residuals:                    2082   BIC:                             9339.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.8793      0.049    201.267      0.000       9.783       9.976
x1             0.8539      0.023     37.239      0.000       0.809       0.899
x2            -2.5560      0.116    -22.045      0.000      -2.783      -2.329
x3             2.3572      0.157     15.032      0.000       2.050       2.665
x4             0.9380      0.172      5.449      0.000       0.600       1.276
x5            -1.1848      0.202     -5.880      0.000      -1.580      -0.790
==============================================================================
Omnibus:                      520.481   Durbin-Watson:                   2.023
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1422.169
Skew:                           1.301   Prob(JB):                    1.51e-309
Kurtosis:                       6.094   Cond. No.                         8.79
==============================================================================

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

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

Al comparar ambos modelos, podemos concluir que el método de PCR no muestra una mejora con respecto al enfoque de VIF. Por el contrario, el modelo PCR, a pesar de tener un valor de R^2 similar, presenta un ligero empeoramiento en el valor del error cuadrático medio. Esta situación puede atribuirse al hecho de que el modelo PCR mantiene características que no son esenciales, y al evaluar el error cuadrático medio, estas características destacan al no ajustarse de manera óptima al modelo.