Fernando Cerriteño Magaña - A01702790

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from numpy.random.mtrand import logistic
from numpy.random.mtrand import standard_cauchy
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
from scipy.stats import f_oneway, t
import scipy.stats as stats
import statsmodels.api as sm

df = pd.read_csv('/content/breast_cancer.csv')

df.head()
```

```
         id diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
0    842302         M        17.99         10.38          122.80
1001.0
1    842517         M        20.57         17.77          132.90
1326.0
2  84300903         M        19.69         21.25          130.00
1203.0
3  84348301         M        11.42         20.38           77.58
386.1
4  84358402         M        20.29         14.34          135.10
1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0          0.11840           0.27760          0.3001
0.14710
1          0.08474           0.07864          0.0869
0.07017
2          0.10960           0.15990          0.1974
0.12790
3          0.14250           0.28390          0.2414
0.10520
4          0.10030           0.13280          0.1980
0.10430

   ...  radius_worst  texture_worst  perimeter_worst  area_worst  \
0  ...         25.38          17.33           184.60      2019.0
1  ...         24.99          23.41           158.80      1956.0
2  ...         23.57          25.53           152.50      1709.0
```

```
3  ...         14.91          26.50            98.87       567.7
4  ...         22.54          16.67           152.20      1575.0

   smoothness_worst  compactness_worst  concavity_worst  concave
points_worst  \
0            0.1622             0.6656           0.7119
0.2654
1            0.1238             0.1866           0.2416
0.1860
2            0.1444             0.4245           0.4504
0.2430
3            0.2098             0.8663           0.6869
0.2575
4            0.1374             0.2050           0.4000
0.1625

   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
1          0.2750                  0.08902
2          0.3613                  0.08758
3          0.6638                  0.17300
4          0.2364                  0.07678

[5 rows x 32 columns]
```

```
df = df.rename(columns={'concave points_mean': "concave_points_mean"})

df = df.drop(['id'], axis = 1)
df = df.drop(['diagnosis'], axis = 1)
```

Verificación que los datos esten completos

```
df.isnull().sum()
```

```
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave_points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
```

```
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

Se revisa que no exista correlación entre los datos

```
correlacion = df.corr()

alta_corr = np.where((correlacion>0.95)&(correlacion<1))
alta_corr
```

```
(array([ 0,  0,  0,  0,  2,  2,  2,  2,  3,  3,  3,  3,  3, 10, 10,
12, 13,
        20, 20, 20, 20, 20, 22, 22, 22, 22, 22, 23, 23, 23]),
 array([ 2,  3, 20, 22,  0,  3, 20, 22,  0,  2, 20, 22, 23, 12, 13,
10, 10,
         0,  2,  3, 22, 23,  0,  2,  3, 20, 23,  3, 20, 22]))
```

```
baja_corr = np.where((correlacion<-0.95)&(correlacion>-1))
baja_corr
```

```
(array([], dtype=int64), array([], dtype=int64))
```

Se ajustan los datos para remover correlacion

```
scaler = StandardScaler()
df_estandar = scaler.fit_transform(df)

df_estandar = pd.DataFrame(df_estandar,columns=df.columns)
```

Para nuestro caso, la hipotesis nula para un coeficiente de regresión (β_i)es:

$\left(H_o\right): \beta_i = ¿ 0$

El estadístico de prueba para un coeficiente de regresión (t-valor). Este valor es calculado dividiendo el coeficiente de regresión estimado ($\widehatβ^i$) por su error estándar ($SE(\widehatβ^i)$):

$t = \frac{\widehatβ^i}{SE(\widehatβ^i)}$

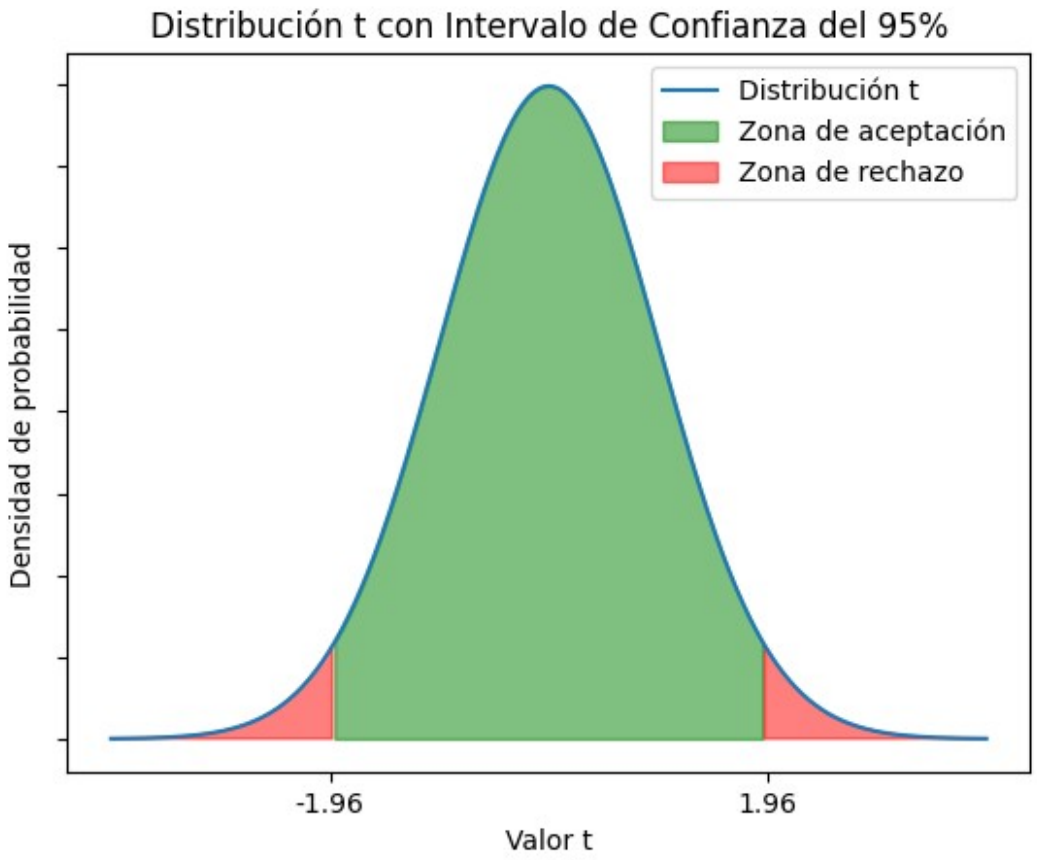
La distribución del estadístico de prueba depende del estadistico de prueba, no obstante bajo la asumnción de que el estadistico de prueba es t, la distribución va a seguir una distribución t student con $n - k - 1$ grados de libertad donde n es el número de observaciones y k es el número de variables independientes en el modelo.

Diagrama de confianza

```
grados_libertad = len(df.axes[0]) - len(df.axes[1])

grados_libertad
```

```
539
```

```
degrees_of_freedom = 539
confidence_level = 0.95
critical_value = stats.t.ppf(1 - (1 - confidence_level) / 2,
degrees_of_freedom)

t_values = np.linspace(-4, 4, 400)
t_distribution = stats.t.pdf(t_values, degrees_of_freedom)

fig,ax = plt.subplots(1)
ax.plot(t_values, t_distribution, label='Distribución t')

ax.fill_between(t_values, 0, t_distribution, where=np.abs(t_values) <=
critical_value, alpha=0.5, color='green', label='Zona de aceptación')
ax.fill_between(t_values, 0, t_distribution, where=np.abs(t_values) >
critical_value, alpha=0.5, color='red', label='Zona de rechazo')

plt.xlabel('Valor t')
plt.ylabel('Densidad de probabilidad')
plt.title('Distribución t con Intervalo de Confianza del 95%')
ax.set_yticklabels([])
ax.set_xticks([-2, 2])
ax.set_xticklabels([-1.960,1.960])
ax.legend()
plt.show()
```



Modelo de regresión multiple

```
train, test = train_test_split(df_estandar, test_size = 0.2,
random_state = 42)

modelo =
smf.ols(formula='radius_mean~texture_mean+perimeter_mean+area_mean+smo
othness_mean+compactness_mean+concavity_mean+concave_points_mean+symme
try_mean+fractal_dimension_mean+radius_se+texture_se', data=train)

modelo = modelo.fit()

print(modelo.summary())
```

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:            radius_mean   R-squared:
0.999
Model:                            OLS   Adj. R-squared:
0.999
Method:                 Least Squares   F-statistic:
6.678e+04
```

```
Date:                Mon, 04 Sep 2023   Prob (F-statistic):
0.00
Time:                        23:51:18   Log-Likelihood:
1039.7
No. Observations:                 455   AIC:
-2055.
Df Residuals:                     443   BIC:
-2006.
Df Model:                          11

Covariance Type:            nonrobust

==============================================================================
                     coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
Intercept              0.0002      0.001      0.210      0.834      -0.002
0.003
texture_mean           0.0013      0.001      0.904      0.366      -0.001
0.004
perimeter_mean         1.0850      0.011    101.344      0.000       1.064
1.106
area_mean             -0.0266      0.010     -2.749      0.006      -0.046
-0.008
smoothness_mean        0.0052      0.002      2.614      0.009       0.001
0.009
compactness_mean      -0.0757      0.004    -17.113      0.000      -0.084
-0.067
concavity_mean        -0.0169      0.004     -4.448      0.000      -0.024
-0.009
concave_points_mean   -0.0047      0.006     -0.847      0.397      -0.016
0.006
symmetry_mean          0.0037      0.002      2.338      0.020       0.001
0.007
fractal_dimension_mean     0.0110      0.003      3.700      0.000       0.005
0.017
radius_se             -0.0032      0.002     -1.553      0.121      -0.007
0.001
texture_se            -0.0021      0.001     -1.459      0.145      -0.005
0.001
==============================================================================
Omnibus:                       71.350   Durbin-Watson:
1.938
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
734.308
Skew:                           0.216   Prob(JB):
```

```
3.52e-160
Kurtosis:                       9.209   Cond. No.
28.5
==============================================================================
========

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

```
modelo =
smf.ols(formula='radius_mean~texture_mean+area_mean+smoothness_mean+co
mpactness_mean+concavity_mean+concave_points_mean+symmetry_mean+fracta
l_dimension_mean+radius_se+texture_se', data=train)

modelo = modelo.fit()

print(modelo.summary())
```

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:            radius_mean   R-squared:
0.985
Model:                            OLS   Adj. R-squared:
0.985
Method:                 Least Squares   F-statistic:
3002.
Date:                Mon, 04 Sep 2023   Prob (F-statistic):
0.00
Time:                        23:51:19   Log-Likelihood:
314.92
No. Observations:                 455   AIC:
-607.8
Df Residuals:                     444   BIC:
-562.5
Df Model:                          10

Covariance Type:            nonrobust

==============================================================================
====================
                             coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
--------------------
Intercept                  0.0005      0.006      0.089      0.929
-0.011       0.012
texture_mean               0.0029      0.007      0.420      0.675
```

```
-0.011       0.016
area_mean                  0.8819      0.018     49.624      0.000
0.847       0.917
smoothness_mean           -0.0103      0.010     -1.054      0.293
-0.030       0.009
compactness_mean           0.1727      0.018      9.543      0.000
0.137       0.208
concavity_mean            -0.0749      0.018     -4.062      0.000
-0.111      -0.039
concave_points_mean        0.1000      0.027      3.728      0.000
0.047       0.153
symmetry_mean             -0.0012      0.008     -0.155      0.877
-0.016       0.014
fractal_dimension_mean    -0.1429      0.013    -11.395      0.000
-0.168      -0.118
radius_se                 -0.0741      0.009     -7.868      0.000
-0.093      -0.056
texture_se                -0.0029      0.007     -0.406      0.685
-0.017       0.011
==============================================================================
========
Omnibus:                       172.917   Durbin-Watson:
2.070
Prob(Omnibus):                   0.000   Jarque-Bera (JB):
1176.239
Skew:                           -1.475   Prob(JB):
3.83e-256
Kurtosis:                       10.304   Cond. No.
12.0
==============================================================================
========

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

En forma matricial, la fórmula de regresión lineal múltiple se puede expresar como:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon$$

Donde:

y es la variable de respuesta.

β_0 es el intercepto que representa el valor de y cuando tudas las variables predictoras son cero.

$\beta_1, \beta_2, \ldots, \beta_p$ son los coeficientes de regresión que representan el cambio en y asociado con un cambio unitario en las varibales predictoras correspondientes.

$x_1, x_2, \ldots, x_p$ son las variables predictoras.

ε es el término de error que representa la variablilidad no explicada del modelo.

```
coefs = modelo.HC0_se.to_numpy()

print('Lista de coeficientes del modelo ', coefs)
```

```
Lista de coeficientes del modelo  [0.00561086 0.00717197 0.03028286
0.01311619 0.02493743 0.03253191
 0.04290769 0.00758578 0.0169003  0.02362515 0.00895742]
```

```
y_pred = modelo.predict(test)

E = test.radius_mean-y_pred
```

Comparación entre datos reales y predicción

```
tabla = pd.DataFrame(data={'Valor real':
test.radius_mean,'Predicción':y_pred,'Error':E})
tabla.head()
```

```
     Valor real  Predicción     Error
204   -0.470694   -0.466938 -0.003756
70     1.366877    1.320826  0.046052
131    0.378508    0.395180 -0.016672
431   -0.490575   -0.537279  0.046704
540   -0.734828   -0.722212 -0.012616
```

En la prueba F-Fisher, la hipótesis nula $H\,0$ y la distribución del estadístico de prueba varían según el contexto.

La hipótesis nula para la prueba F en la significancia del modelo es:

$H\,0$: Todos los coeficientes de regresión son iguales a cero, lo que significa que el modelo no tiene capacidad para explicar la variabilidad en la variable objetivo. En otras palabras, las variables predictoras no tienen un efecto conjunto significativo en la variable de respuesta.

La hipótesis alternativa $H\,1$ es que al menos uno de los coeficientes de regresión es diferente de cero, lo que indica que el modelo tiene algún valor predictivo significativo.

El estadístico de prueba F sigue una distribución F de Fisher. Si el modelo completo tiene p variables predictoras y el modelo reducido no tiene predictoras (solo constante), entonces:

Grados de libertad del numerador $d\,f\,n$: p (número de variables predictoras en el modelo completo).

Grados de libertad del denominador $d\,f\,d$: N - p - 1 (número total de observaciones menos el número de variables predictoras menos 1 para el intercepto).

```
# Realizar la prueba ANOVA
resultado_anova = f_oneway(df['texture_mean'], df['area_mean'],
df['smoothness_mean'], df['compactness_mean'], df['concavity_mean'],
```

```
df['concave_points_mean'], df['symmetry_mean'],
df['fractal_dimension_mean'], df['radius_se'], df['texture_se'])

# Obtener los grados de libertad del numerador y del denominador
dfn = len(df.columns) - 1
dfd = len(df) - len(df.columns)

print("Estadística F:", resultado_anova.statistic)
print("Valor p:", resultado_anova.pvalue)
print("Grados de libertad del numerador:", dfn)
print("Grados de libertad del denominador:", dfd)
```

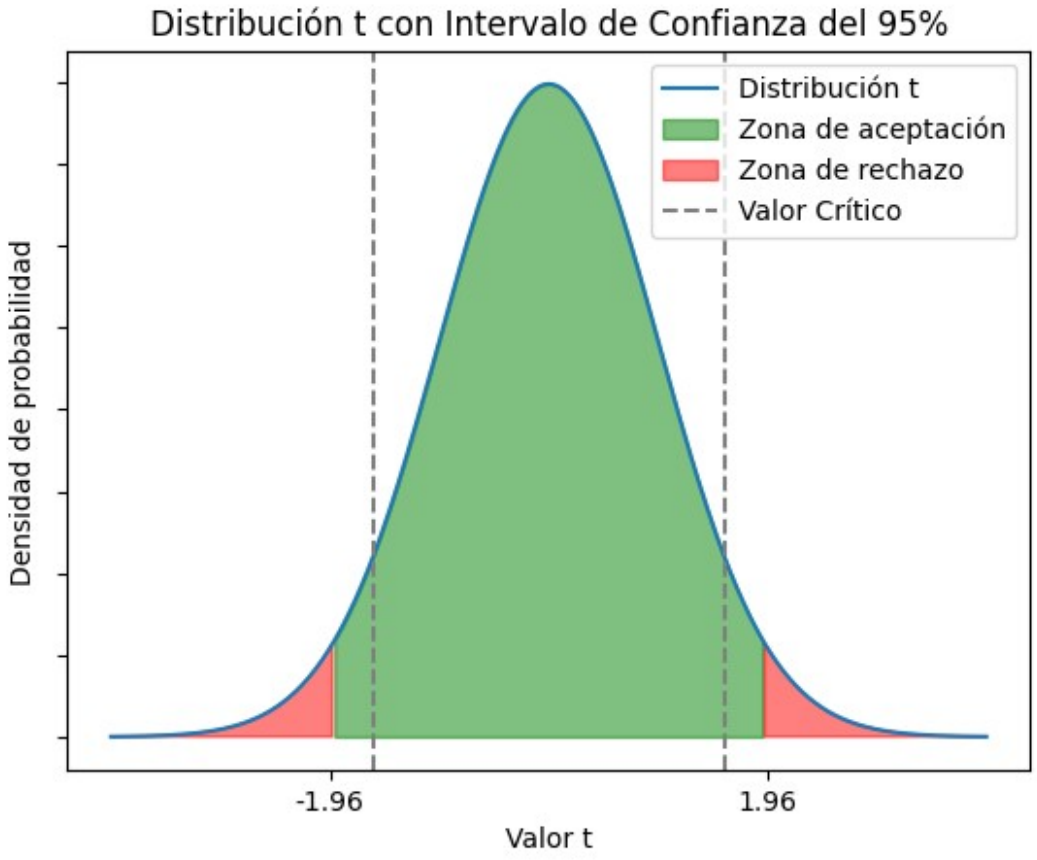
```
Estadística F: 1957.4926598413363
Valor p: 0.0
Grados de libertad del numerador: 29
Grados de libertad del denominador: 539
```

```
degrees_of_freedom = 539
confidence_level = 0.95

# Calcular el valor crítico
valor_critico = stats.f.ppf(1 - (1 - confidence_level) / 2, dfn, dfd)

print("Valor crítico de F:", valor_critico)
```

```
Valor crítico de F: 1.6040149414026632
```

```
critical_value = stats.t.ppf(1 - (1 - confidence_level) / 2,
degrees_of_freedom)

t_values = np.linspace(-4, 4, 400)
t_distribution = stats.t.pdf(t_values, degrees_of_freedom)

fig,ax = plt.subplots(1)
ax.plot(t_values, t_distribution, label='Distribución t')

ax.fill_between(t_values, 0, t_distribution, where=np.abs(t_values) <=
critical_value, alpha=0.5, color='green', label='Zona de aceptación')
ax.fill_between(t_values, 0, t_distribution, where=np.abs(t_values) >
critical_value, alpha=0.5, color='red', label='Zona de rechazo')

ax.axvline(x=valor_critico, color='grey', linestyle='--', label='Valor
Crítico')
ax.axvline(x=-valor_critico, color='grey', linestyle='--')

plt.xlabel('Valor t')
plt.ylabel('Densidad de probabilidad')
plt.title('Distribución t con Intervalo de Confianza del 95%')
ax.set_yticklabels([])
ax.set_xticks([-2, 2])
ax.set_xticklabels([-1.960,1.960])
```

```
ax.legend()
plt.show()
```



Modelo de regresión hacia atrás

```
# Función para ajustar un modelo y calcular el valor p para cada característica
def fit_and_get_p_values(X, y):
    model = sm.OLS(y, X).fit()
    p_values = model.pvalues
    return p_values

df = sm.add_constant(df)

df.head()
```

	const	radius_mean	texture_mean	perimeter_mean	area_mean	\
0	1.0	17.99	10.38	122.80	1001.0	
1	1.0	20.57	17.77	132.90	1326.0	
2	1.0	19.69	21.25	130.00	1203.0	
3	1.0	11.42	20.38	77.58	386.1	
4	1.0	20.29	14.34	135.10	1297.0	

```
   smoothness_mean  compactness_mean  concavity_mean
concave_points_mean  \
0          0.11840           0.27760          0.3001
0.14710
1          0.08474           0.07864          0.0869
0.07017
2          0.10960           0.15990          0.1974
0.12790
3          0.14250           0.28390          0.2414
0.10520
4          0.10030           0.13280          0.1980
0.10430

   symmetry_mean  ...  radius_worst  texture_worst  perimeter_worst  \
0         0.2419  ...         25.38          17.33           184.60
1         0.1812  ...         24.99          23.41           158.80
2         0.2069  ...         23.57          25.53           152.50
3         0.2597  ...         14.91          26.50            98.87
4         0.1809  ...         22.54          16.67           152.20

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   concave points_worst  symmetry_worst  fractal_dimension_worst
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 31 columns]
```

El objetivo principal en regresión hacia atras es obtener un modelo más simple y más interpretable sin sacrificar la precisión de la predicción. El criterio general para eliminar variables en el modelo de regresión hacia atrás es:

1. En un modelo completo que cuente con un modelo de regresión ajustado, se selecciona un criterio de eliminación, como un nivel de significancia o un cambio en el estadistico F.

2. Se elimina la variable predictora menos significativa segun el criterio seleccionado en el paso anterior, en este caso se revisa el p value.

3. Se realiza un reajuste del modelo con las variables restantes.

4. Se evalua el nuevo modelo y se compara con el modelo anterior, esto con el proposito de verificar si el nuevo modelo es mejor, y si su capacidad predictiva sigue siendo igual o mejor.

5. Se repite el proceso hasta que las variables restantes en el modelo sean significativas o hasta que se este satisfecho con la precisión y simplicidad del modelo.

```
max_p_value = 0.05
while True:
    p_values =
fit_and_get_p_values(train.drop(columns=['radius_mean']),
train['radius_mean'])
    max_p_value_index = p_values.idxmax()
    max_p_value_value = p_values[max_p_value_index]

    if max_p_value_value > max_p_value:
        print(f"Eliminando '{max_p_value_index}' con valor
p={max_p_value_value:.4f}")
        train = train.drop(columns=max_p_value_index)
        test = test.drop(columns=max_p_value_index)
    else:
        break
```

```
Eliminando 'texture_worst' con valor p=0.9662
Eliminando 'texture_se' con valor p=0.8981
Eliminando 'concavity_worst' con valor p=0.8738
Eliminando 'concave_points_mean' con valor p=0.6400
Eliminando 'concave points_worst' con valor p=0.7375
Eliminando 'smoothness_se' con valor p=0.5272
Eliminando 'compactness_se' con valor p=0.5052
Eliminando 'area_se' con valor p=0.4497
Eliminando 'fractal_dimension_worst' con valor p=0.4317
Eliminando 'radius_se' con valor p=0.0949
Eliminando 'texture_mean' con valor p=0.0787
```

Análisis de los resultados

```
final_model = sm.OLS(train['radius_mean'],
train.drop(columns=['radius_mean'])).fit()
print(final_model.summary())
```

```
                                 OLS Regression Results

=============================================================================
=================
Dep. Variable:            radius_mean   R-squared (uncentered):
1.000
Model:                            OLS   Adj. R-squared (uncentered):
```

```
1.000
Method:                 Least Squares   F-statistic:
9.723e+04
Date:                Mon, 04 Sep 2023   Prob (F-statistic):
0.00
Time:                        23:51:19   Log-Likelihood:
1240.2
No. Observations:                 455   AIC:
-2444.
Df Residuals:                     437   BIC:
-2370.
Df Model:                          18

Covariance Type:            nonrobust

==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
perimeter_mean               0.9424      0.016     58.384      0.000
0.911       0.974
area_mean                    0.0775      0.013      5.895      0.000
0.052       0.103
smoothness_mean              0.0072      0.002      3.796      0.000
0.003       0.011
compactness_mean            -0.0576      0.004    -14.084      0.000
-0.066      -0.050
concavity_mean              -0.0336      0.003    -10.836      0.000
-0.040      -0.027
symmetry_mean                0.0034      0.001      2.382      0.018
0.001       0.006
fractal_dimension_mean       0.0060      0.002      2.589      0.010
0.001       0.010
perimeter_se                -0.0126      0.002     -6.936      0.000
-0.016      -0.009
concavity_se                 0.0119      0.002      5.826      0.000
0.008       0.016
concave points_se            0.0038      0.002      2.308      0.021
0.001       0.007
symmetry_se                  0.0044      0.001      3.255      0.001
0.002       0.007
fractal_dimension_se        -0.0045      0.002     -2.896      0.004
-0.008      -0.001
radius_worst                 0.2380      0.013     18.663      0.000
0.213       0.263
perimeter_worst             -0.1206      0.011    -10.680      0.000
-0.143      -0.098
```

```
area_worst                -0.0834      0.011     -7.851      0.000
-0.104      -0.063
smoothness_worst          -0.0048      0.002     -2.688      0.007
-0.008      -0.001
compactness_worst          0.0126      0.003      5.031      0.000
0.008       0.018
symmetry_worst            -0.0044      0.002     -2.352      0.019
-0.008      -0.001
==============================================================================
========
Omnibus:                       51.429   Durbin-Watson:
2.108
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
276.939
Skew:                           0.268   Prob(JB):
7.30e-61
Kurtosis:                       6.784   Cond. No.
93.1
==============================================================================
========

Notes:
[1] R² is computed without centering (uncentered) since the model does
not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

```
y_pred = final_model.predict(test.drop(columns=['radius_mean']))
tabla = pd.DataFrame(data={'Valor real':
test.radius_mean,'Predicción':y_pred,'Error':test.radius_mean-y_pred})
```

Analisis de resultados

```
tabla
```

```
     Valor real  Predicción     Error
204   -0.470694   -0.454358 -0.016337
70     1.366877    1.366228  0.000650
131    0.378508    0.392072 -0.013564
431   -0.490575   -0.504292  0.013716
540   -0.734828   -0.739252  0.004424
..          ...         ...       ...
486    0.145616    0.152166 -0.006550
75     0.551757    0.546082  0.005674
249   -0.740508   -0.739309 -0.001199
238    0.026330    0.014841  0.011489
265    1.875263    1.832638  0.042625

[114 rows x 3 columns]
```

```
print('MAE =',np.sum(np.abs(tabla.Error)/len(tabla))**2)
```

```
MAE = 0.0001617317428659931
```