

## ✓ Tarea 1.2 Regresion Lineal Simple

1. En el siguiente bloque de codigo se importara la base de datos felicidad&GDP.csv con la funcion read\_csv() de la libreria pandas. Tambien se imprimiran los paises mas felices y los que tienen mayor GDP

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import statsmodels.api as sm
df = pd.read_csv('felicidad&GDP.csv')

tabla_felicidad = df.sort_values('Felicidad',ascending=False)
print('Top 10 paises mas felices de acuerdo a nuestra base de datos: ')
print(tabla_felicidad.head(10), '\n')
tabla_GDP = df.sort_values(by = 'GDP',ascending=False)
print('Top 10 paises con mayor GDP de acuerdo a nuestra base de datos: ')
print(tabla_GDP.head(10))
```

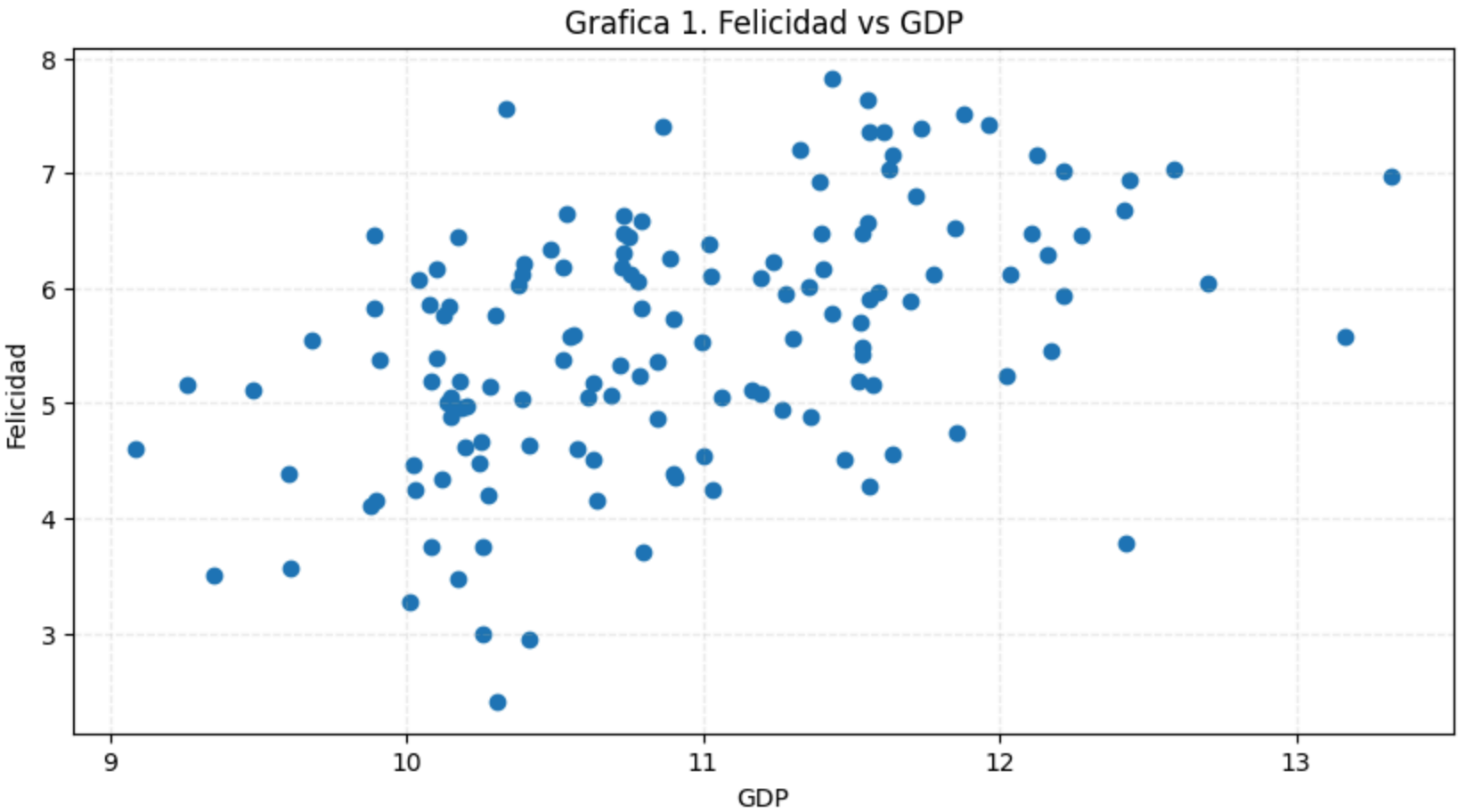
↗

Top 10 paises mas felices de acuerdo a nuestra base de datos:			
	Pais	Felicidad	GDP
0	Finland	7.8210	2.718370e+11
1	Denmark	7.6362	3.560850e+11
2	Iceland	7.5575	2.171808e+10
3	Switzerland	7.5116	7.522480e+11
4	Netherlands	7.4149	9.138650e+11
5	Luxembourg	7.4040	7.335313e+10
6	Sweden	7.3843	5.414870e+11
7	Norway	7.3651	3.621980e+11
8	Israel	7.3638	4.071010e+11
9	New Zealand	7.1998	2.117350e+11
Top 10 paises con mayor GDP de acuerdo a nuestra base de datos:			
	Pais	Felicidad	GDP
15	United States	6.9768	2.089370e+13
70	China	5.5853	1.468770e+13
52	Japan	6.0389	5.040110e+12
13	Germany	7.0341	3.846410e+12
16	United Kingdom	6.9425	2.756900e+12
130	India	3.7771	2.667690e+12
19	France	6.6867	2.630320e+12
29	Italy	6.4667	1.892570e+12
14	Canada	7.0251	1.645420e+12
57	South Korea	5.9351	1.637900e+12

2. En el siguiente bloque de codigo, se graficara el nivel de felicidad de acuerdo al nivel, en logaritmo base 10, del GDP, utilizando el comando log10() de la libreria numpy.

```
x = np.log10(df['GDP'])
y = df['Felicidad']

plt.figure(figsize=(10,5))
plt.title('Grafica 1. Felicidad vs GDP',fontsize = 12)
plt.grid(True, alpha = 0.25,linestyle = '--')
plt.scatter(x,y)
plt.xlabel('GDP')
plt.ylabel('Felicidad')
plt.show()
```



**3.** A continuacion, se calcularan los dos coeficientes de nuestro modelo de regresion lineal simple, utilizando las formulas vistas en clase, y las funciones de numpy, sum y mean.

```
mean_x = np.mean(x)
mean_y = np.mean(y)
beta_1 = np.sum((x - mean_x)*(y - mean_y))/np.sum((x - mean_x)**2)
beta_0 = mean_y - beta_1*mean_x

print('Coeficiente beta 1: ',beta_1)
print('Coeficiente beta 0: ',beta_0)
```



Coeficiente beta 1: 0.628128465881041  
Coeficiente beta 0: -1.3023500570747295

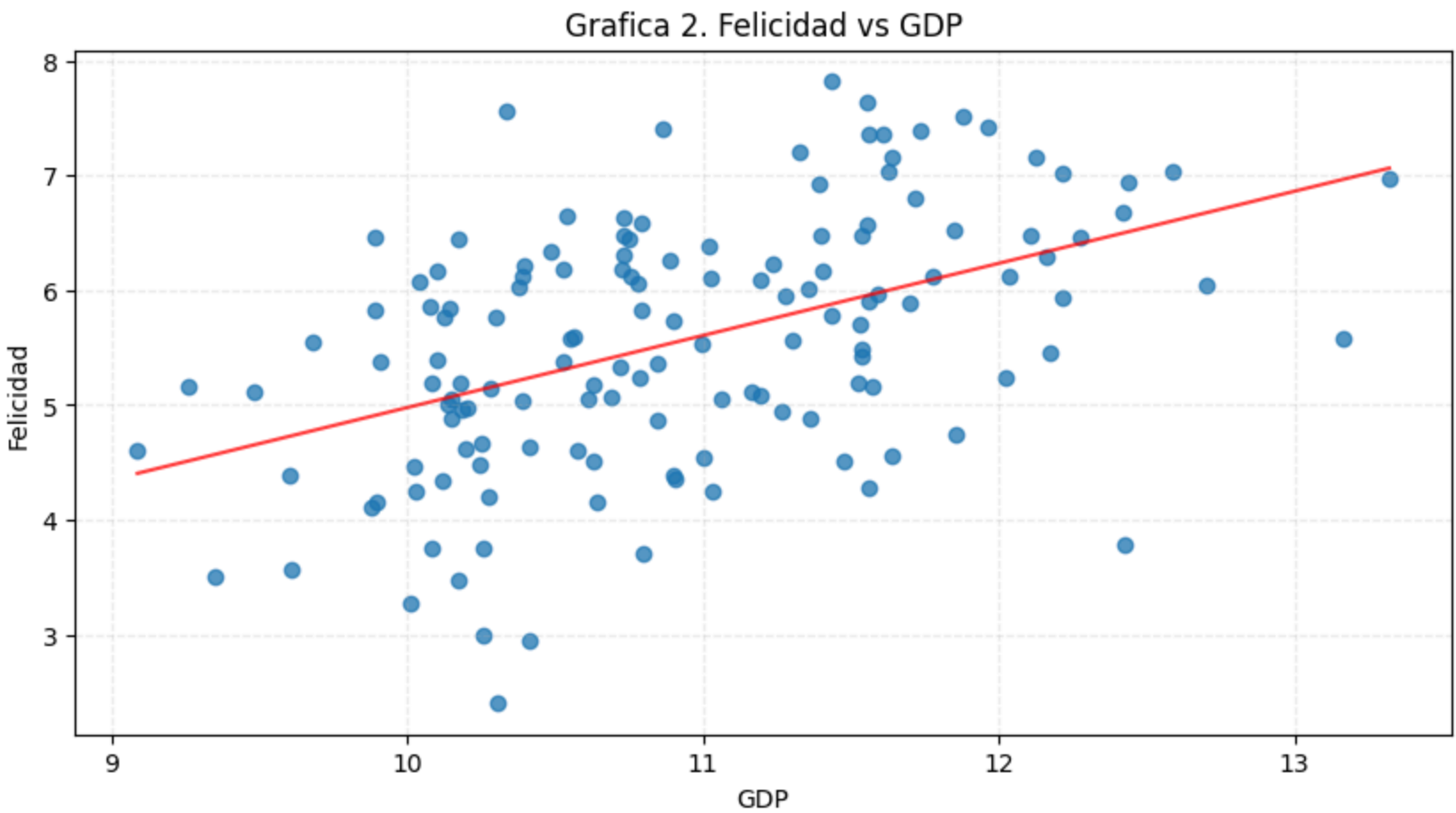
**4.** En el siguiente bloque, se graficarán de nuevo, junto con el nuevo modelo de regresion lineal simple.

```
modelo = beta_0 + beta_1*x

plt.figure(figsize=(10,5))
plt.title('Grafica 2. Felicidad vs GDP',fontsize = 12)
plt.grid(True, alpha = 0.25,linestyle = '--')
plt.scatter(x,y, alpha = 0.75)
plt.xlabel('GDP')
plt.ylabel('Felicidad')

x_linea = [min(x),max(x)]
y_linea = [beta_0 + beta_1*x_linea[0],beta_0 + beta_1*x_linea[1]]
plt.plot(x_linea,y_linea,'r', alpha = 0.75)

plt.show()
```



Observando la grafica anterior, podemos darnos cuenta que la linea de tendencia esta pasando por el centro del area observada en los puntos de nuestra base de datos.

5. A continuacion se calculara el RSS (Residual Sum of Squares) de nuestro modelo, para poderlo utilizar despues en los caluclos de nuestro error estandar de los coeficientes y en el RSE (Residual Standard Error), que sera el error promedio tipico de nuestro modelo

```
e = np.zeros(len(y))
for i in range(len(y)):
    e[i] = y[i] - modelo[i]

RSS = np.sum(e**2)
print('RSS: ',RSS)
```

RSS: 131.3738317732635

6. En el siguiente bloque, calcularemos el error estandar de los coeficientes y el intervalo de confianza del 95% de beta 1.

```
SE_beta1 = np.sqrt(RSS/(len(y) - 2))*1/np.sqrt(np.sum((x - mean_x)**2))
print('Error estandar de beta 1: ',SE_beta1)

t = beta_1/SE_beta1
print('Estadistico t: ',t)
gdl = len(y) - 2
print('Grados de libertad: ',gdl)
p_value = 2*(1 - stats.t.cdf(t,gdl))
print('Valor p: ',p_value)

print('Intervalos de confianza: ', [beta_1 - p_value*SE_beta1,
                                   beta_1 + p_value*SE_beta1])

print('\nDebido a que el valor p es menor que nuestro nivel de significancia ')
print('de 5%, podemos decir que sí existe una relacion significativa entre el ')
print('nivel de felicidad y el valor de GDP en logaritmo base 10 ')
```

Error estandar de beta 1: 0.09983378435340727  
Estadistico t: 6.291742519320849  
Grados de libertad: 139  
Valor p: 3.825717964645037e-09  
Intervalos de confianza: [0.6281284654991051, 0.6281284662629769]

Debido a que el valor p es menor que nuestro nivel de significancia de 5%, podemos decir que sí existe una relacion significativa entre el nivel de felicidad y el valor de GDP en logaritmo base 10

7. A continuacion, se calculara el RSE y la R^2, para saber el error promedio de nuestro modelo y que tan adecuado es para el sistema para los datos utilizados.

```
# Otra forma de calcular ESS y TSS
#z = np.zeros(len(y))
#g = np.zeros(len(y))
```

```
#for i in range(len(y)):
#  z[i] = (y[i] - mean_y)**2
#for i in range(len(y)):
#  g[i] = (modelo[i] - mean_y)**2
#ESS = np.sum(g)

RSE = np.sqrt(RSS/(gdl))

ESS = np.sum((modelo - mean_y)**2)
TSS = np.sum((y - mean_y)**2)

R2 = ESS/TSS

print('RSE: ',RSE)
print('TSS: ',TSS)
print('ESS: ',ESS)
print('R^2: ',R2)

print('\nObservando R^2, podemos concluir que este modelo de regresion lineal')
print('simple, no es el adecuado, ya que el valor de R^2 es muy bajo. ')
print('Lo cual me hace sentido, al observar la grafica 2, se puede ver que')
print('los datos estan muy dispersos en comparacion con la linea de tendencia. ')
```

RSE: 0.9721807858537376  
TSS: 168.78798751626567  
ESS: 37.41415574300217  
R^2: 0.22166361654970654

Observando R^2, podemos concluir que este modelo de regresion lineal simple, no es el adecuado, ya que el valor de R^2 es muy bajo. Lo cual me hace sentido, al observar la grafica 2, se puede ver que los datos estan muy dispersos en comparacion con la linea de tendencia.

8. Para concluir, se utilizara la funcion OLS() de la libreria statsmodels.api, para verificar que los resultados anteriores son correctos.

```
modelo_2 = sm.OLS(y,sm.add_constant(x))
modelo_2 = modelo_2.fit()
print(modelo_2.summary())
pvalues = modelo_2.pvalues

print('\nValor de p: ', f"{pvalues[0]:.15f}",'\n          ', f"{pvalues[1]:.15f}")
print('RSE:', modelo_2.scale**.5,'\n')
```

OLS Regression Results

Dep. Variable:	Felicidad	R-squared:	0.222
Model:	OLS	Adj. R-squared:	0.216
Method:	Least Squares	F-statistic:	39.59
Date:	Mon, 27 Jan 2025	Prob (F-statistic):	3.83e-09
Time:	02:27:37	Log-Likelihood:	-195.09
No. Observations:	141	AIC:	394.2
Df Residuals:	139	BIC:	400.1
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	-1.3024	1.094	-1.191	0.236	-3.465	0.860
GDP	0.6281	0.100	6.292	0.000	0.431	0.826

Omnibus:	2.648	Durbin-Watson:	0.455
Prob(Omnibus):	0.266	Jarque-Bera (JB):	2.523
Skew:	-0.326	Prob(JB):	0.283
Kurtosis:	2.944	Cond. No.	148.

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

Valor de p: 0.235797533420326  
0.000000003825718  
RSE: 0.9721807858537376

<ipython-input-19-6b279390fdc8>:6: FutureWarning: Series.\_\_getitem\_\_ treating keys as positions is deprecated. In a fut  
print('\nValor de p: ', f"{pvalues[0]:.15f}",'\n ', f"{pvalues[1]:.15f}")

**Conclusion** Al obtener los datos de ajuste de este modelo, por medio de la funcion OLS(), se puede observar que el valor de RSE de la funcion es de 0.9736302908855606 , es muy parecido al calculado por el codigo es de 0.9721807858537376. Tambien, observamos que el valor de p de beta 1, calculado por la funcion, es de 3.82e-9, que es menor al nivel de significancia de 0.05, y con esto podemos confirmar que sí hay una relacion significativa entre la felicidad y el GDP de la nacion. Y por ultimo, se puede observar el resultado de

la  $R^2$  de la funcion de la libreria statsmodels.api es de 0.22, que es aproximadamente igual al que fue calculado en el codigo, por lo tanto, esto quiere decir que este modelo no es ideal para nuestra base de datos.