Actividad

Utiliza un modelo de regresión lineal múltiple para predecir el salario en dolares (salary_in_usd) de cada empleado. Las variables regresoras de tu modelo deben de ser las siguientes: nivel de experiencia (experience_level), tipo de empleo (employment_type), salario (salary) y radio remoto (remote_ratio).

Entrega un documento en formato PDF donde se observe la siguiente información.

1.-Ecuación matemática que describe el modelo de regresión lineal a ejecutar. Se debe especificar el nombre de las variables.

2.- Base de datos completa. No se observan valores faltantes. En caso de haberlos se realiza imputación simple.

3.-Mostrar que las variables regresoras son independientes. En caso de no serlo realizar el procedimiento correspondiente.

4.- Calculo de R^2, calculo de los coeficientes de regresión y p-valor; interpretación de resultados.

5.-Comparación entre datos reales y predicción. Análisis de los resultados.

6.-Análisis de los errores mediante diferentes medios (QQ-plot, histograma, test Kolmogorov etc.). Mostrar las gráficas correspondientes y el análisis de resultados

El trabajo se realizará de forma individual. La forma de entrega será mediante un documento PDF en canvas.

El documento debe de tener como pie de página tu nombre con tu matrícula

Es una actividad de puntos extra, si deseas que te cuente para la calificación la debes entregar antes de la fecha solicitada.

Importamos las librerias necesarias

```
import pandas as pd
import numpy as np
df = pd.read_csv('/content/drive/MyDrive/7mo Semestre/Colab Notebooks/DataSources/ds_salaries.csv')
```

Analizamos los datos

- El tamaño de la tabla
- Los nombres de las columnas
- Deteccion de valores nulos
- Deteccion y eliminacion de columnas no necesarias para la regresion lineal

```
df.head()
```

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_resic
0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	
1	1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	
2	2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	
3	3	2020	MI	FT	Product Data Analyst	20000	USD	20000	
4	4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	

```
df.shape
```

```
(607, 12)
```

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

```
Unnamed: 0          0
work_year           0
experience_level    0
employment_type     0
job_title           0
salary              0
```

```
salary_currency       0
salary_in_usd         0
employee_residence    0
remote_ratio          0
company_location      0
company_size          0
dtype: int64
```

```
df.drop('Unnamed: 0', axis = 1, inplace = True)
df.drop('job_title', axis = 1, inplace = True)
df.drop('salary_currency', axis = 1, inplace = True)
df.drop('employee_residence', axis = 1, inplace = True)
df.drop('company_location', axis = 1, inplace = True)
df.drop('company_size', axis = 1, inplace = True)
df.drop('work_year', axis = 1, inplace = True)
```

```
df.head()
```

	experience_level	employment_type	salary	salary_in_usd	remote_ratio
0	MI	FT	70000	79833	0
1	SE	FT	260000	260000	0
2	SE	FT	85000	109024	50
3	MI	FT	20000	20000	0
4	SE	FT	150000	150000	50

Creamos variables dummies

- Encontramos las variables categoricas y las sustituimos con valores numericos equivalentes.

```
df['employment_type'].unique()
```

```
array(['FT', 'CT', 'PT', 'FL'], dtype=object)
```

```
df['experience_level'].unique()
```

```
array(['MI', 'SE', 'EN', 'EX'], dtype=object)
```

- Generamos los arreglos de las variables dummies.

```
dummies_employ = pd.get_dummies(df['employment_type'], prefix = 'employment_type')
```

```
dummies_employ
```

	employment_type_CT	employment_type_FL	employment_type_FT	employment_type_PT
0	0	0	1	0
1	0	0	1	0
2	0	0	1	0
3	0	0	1	0
4	0	0	1	0
...	...	...	...	...
602	0	0	1	0
603	0	0	1	0
604	0	0	1	0
605	0	0	1	0
606	0	0	1	0

607 rows × 4 columns

```
dummies_exp = pd.get_dummies(df['experience_level'], prefix = 'experience_level')
```

```
dummies_exp
```

	experience_level_EN	experience_level_EX	experience_level_MI	experience_level_SE
0	0	0	1	0
1	0	0	0	1
2	0	0	0	1
3	0	0	1	0
4	0	0	0	1
...	...	...	...	...
602	0	0	0	1
603	0	0	0	1
604	0	0	0	1
605	0	0	0	1
606	0	0	1	0

- Concatenamos los arreglos de las variables dummies con el dataframe principal.

```
df = pd.concat([df, dummies_employ], axis = 1)

df = pd.concat([df, dummies_exp], axis = 1)

df
```

	experience_level	employment_type	salary	salary_in_usd	remote_ratio	employment_type_CT	employment_type_FL	employ
0	MI	FT	70000	79833	0	0	0	
1	SE	FT	260000	260000	0	0	0	
2	SE	FT	85000	109024	50	0	0	
3	MI	FT	20000	20000	0	0	0	
4	SE	FT	150000	150000	50	0	0	
...	...	...	...	...	...	...	...	
602	SE	FT	154000	154000	100	0	0	
603	SE	FT	126000	126000	100	0	0	
604	SE	FT	129000	129000	0	0	0	
605	SE	FT	150000	150000	100	0	0	
606	MI	FT	200000	200000	100	0	0	

607 rows × 13 columns

- Eliminamos las columnas en las que se basan las variables dummies.

```
df.drop('experience_level', axis = 1, inplace = True)

df.drop('employment_type', axis = 1, inplace = True)
```

Empezamos con el modelo de regresion lineal

- Primero buscamos la correlación entre variables.

```
corr = df.corr()

corr
```

	salary	salary_in_usd	remote_ratio	employment_type_CT	employment_type_FL	employment_type_FT	emp
salary	1.000000	-0.083906	-0.014608	-0.008268	-0.014568	0.025685	
salary_in_usd	-0.083906	1.000000	0.132122	0.092907	-0.073863	0.091819	
remote_ratio	-0.014608	0.132122	1.000000	0.065149	-0.016865	-0.023834	
employment_type_CT	-0.008268	0.092907	0.065149	1.000000	-0.007423	-0.506989	
employment_type_FL	-0.014568	-0.073863	-0.016865	-0.007423	1.000000	-0.453089	
experience_level_EN	-0.015845	-0.294196	-0.010490	0.066013	-0.033537	-0.167828	

- La siguiente linea nos generará un arreglo con las variables que se encuentran altamente correlacionadas, especificamente, con una correlación mayor a 0.95 y menor a 1.

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

alta_corr
```

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

- La siguiente linea nos generará un arreglo con las variables que se encuentran altamente correlacionadas de manera negativa, especificamente, con una correlación menor a -0.95 y mayor a -1.

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

baja_corr
```

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

- Como podemos observar, no tenemos variables altamente correlacionadas, ya sea positivamente o negativamente. Lo cual es bueno, significa que no tenemos que eliminar variables.

- A continuación estandarizamos el dataframe.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df_estandar = scaler.fit_transform(df)

df_estandar
```

```
array([[-0.16460538, -0.45790445, -1.74361532, ..., -0.21154298,
         1.36006076, -0.9253481 ],
       [-0.0414754 ,  2.08328151, -1.74361532, ..., -0.21154298,
        -0.73526127,  1.08067439],
       [-0.15488459, -0.04617667, -0.51437665, ..., -0.21154298,
        -0.73526127,  1.08067439],
       ...,
       [-0.12637028,  0.2355771 , -1.74361532, ..., -0.21154298,
        -0.73526127,  1.08067439],
       [-0.11276118,  0.53177399,  0.71486203, ..., -0.21154298,
        -0.73526127,  1.08067439],
       [-0.08035855,  1.23700468,  0.71486203, ..., -0.21154298,
         1.36006076, -0.9253481 ]])
```

- Pero aun no es un dataframe, es un arreglo. Así que lo modelamos como dataframe.

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

df_estandar
```

	salary	salary_in_usd	remote_ratio	employment_type_CT	employment_type_FL	employment_type_FT	employment_type_PT
0	-0.164605	-0.457904	-1.743615	-0.091135	-0.081446	0.179758	-0.129423
1	-0.041475	2.083282	-1.743615	-0.091135	-0.081446	0.179758	-0.129423
2	-0.154885	-0.046177	-0.514377	-0.091135	-0.081446	0.179758	-0.129423
3	-0.197008	-1.301826	-1.743615	-0.091135	-0.081446	0.179758	-0.129423
603	-0.128314	0.193263	0.714862	-0.091135	-0.081446	0.179758	-0.129423

Entrenamiento

- A continuación, empezamos a entrenar el modelo.

```
from sklearn.model_selection import train_test_split

entrenamiento, prueba = train_test_split(df_estandar, test_size=0.20, random_state=42)

entrenamiento
```

	salary	salary_in_usd	remote_ratio	employment_type_CT	employment_type_FL	employment_type_FT	employment_type_PT
9	-0.128962	0.179159	-0.514377	-0.091135	-0.081446	0.179758	-0.129423
227	-0.161365	-0.333488	-0.514377	-0.091135	-0.081446	0.179758	-0.129423
591	-0.116096	0.459192	0.714862	-0.091135	-0.081446	0.179758	-0.129423
516	-0.111141	0.567036	0.714862	-0.091135	-0.081446	0.179758	-0.129423
132	-0.185084	-1.042301	0.714862	-0.091135	-0.081446	0.179758	-0.129423
...	...	...	...	...	...	...	...
71	-0.185991	-0.988746	-0.514377	-0.091135	-0.081446	0.179758	-0.129423
106	-0.057677	1.059879	0.714862	-0.091135	-0.081446	0.179758	-0.129423
270	-0.162985	-0.561334	0.714862	-0.091135	-0.081446	0.179758	-0.129423
435	-0.164605	-0.291738	0.714862	-0.091135	-0.081446	0.179758	-0.129423
102	6.918609	-1.072499	-0.514377	-0.091135	-0.081446	0.179758	-0.129423

485 rows × 11 columns

```
df.columns
```

```
Index(['salary', 'salary_in_usd', 'remote_ratio', 'employment_type_CT',
       'employment_type_FL', 'employment_type_FT', 'employment_type_PT',
       'experience_level_EN', 'experience_level_EX', 'experience_level_MI',
       'experience_level_SE'],
      dtype='object')
```

- Calculamos el modelo de regresion lineal con los datos de entrenamiento.

```
import statsmodels.formula.api as smf

modelo = smf.ols(formula = 'salary_in_usd~salary+remote_ratio+employment_type_CT+employment_type_FL+employment_type_FT+experience_level_E

modelo = modelo.fit()

print(modelo.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          salary_in_usd   R-squared:                       0.264
Model:                            OLS   Adj. R-squared:                  0.252
Method:                 Least Squares   F-statistic:                     21.37
Date:                Fri, 18 Aug 2023   Prob (F-statistic):           8.41e-28
Time:                        03:15:41   Log-Likelihood:                -627.06
No. Observations:                 485   AIC:                             1272.
Df Residuals:                     476   BIC:                             1310.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept              0.0167      0.040      0.413      0.680      -0.063       0.096
salary                -0.1455      0.065     -2.251      0.025      -0.272      -0.018
remote_ratio           0.0592      0.040      1.463      0.144      -0.020       0.139
employment_type_CT     0.0980      0.050      1.974      0.049       0.000       0.196
employment_type_FL    -0.0042      0.058     -0.073      0.941      -0.117       0.109
```

```
employment_type_FT      0.0879      0.057      1.531      0.126      -0.025       0.201
experience_level_EN    -0.3717      0.044     -8.423      0.000      -0.458      -0.285
experience_level_EX     0.1902      0.040      4.698      0.000       0.111       0.270
experience_level_MI    -0.3225      0.044     -7.387      0.000      -0.408      -0.237
==============================================================================
Omnibus:                      242.000   Durbin-Watson:                   1.979
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2005.484
Skew:                           2.000   Prob(JB):                         0.00
Kurtosis:                      12.123   Cond. No.                         2.46
==============================================================================

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

Si vemos nuestro valor de r^2 *(R-squared)*, nos daremos cuenta que el valor no es muy bueno, ya que tiene que estar lo mas cercano a 1, por lo tanto, es posible que podamos mejorar ese valor al quitar las variables que su valor β *(coef)* es igual a cero, y esto se sabe por medio del p-valor, que en los datos se expresa como $P > |t|$, si el p-valor es menor o igual a 0.05, se rechaza la hipotesis nula, que dice que $\beta = 0$, así que conservamos el coeficiente.

```
modelo = smf.ols(formula = 'salary_in_usd~salary+remote_ratio+employment_type_CT+employment_type_FT+experience_level_EN+experience_level_

modelo = modelo.fit()

print(modelo.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          salary_in_usd   R-squared:                       0.264
Model:                            OLS   Adj. R-squared:                  0.253
Method:                 Least Squares   F-statistic:                     24.47
Date:                Fri, 18 Aug 2023   Prob (F-statistic):           1.63e-28
Time:                        03:15:41   Log-Likelihood:                -627.06
No. Observations:                 485   AIC:                             1270.
Df Residuals:                     477   BIC:                             1304.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0168      0.040      0.416      0.677      -0.063       0.096
salary        -0.1455      0.065     -2.253      0.025      -0.272      -0.019
remote_ratio   0.0591      0.040      1.463      0.144      -0.020       0.139
employment_type_CT      0.0990      0.048      2.070      0.039       0.005       0.193
employment_type_FT      0.0898      0.051      1.765      0.078      -0.010       0.190
experience_level_EN    -0.3713      0.044     -8.503      0.000      -0.457      -0.285
experience_level_EX     0.1903      0.040      4.704      0.000       0.111       0.270
experience_level_MI    -0.3225      0.044     -7.395      0.000      -0.408      -0.237
==============================================================================
Omnibus:                      242.008   Durbin-Watson:                   1.979
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2005.729
Skew:                           2.000   Prob(JB):                         0.00
Kurtosis:                      12.124   Cond. No.                         2.08
==============================================================================

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

Como podemos observar, no cambió nada el valor de r^2, pero el p-valor de una de las variables ahora es mayor a 0.05, por lo que volvemos a hacer el modelo sin esa variable.

```
modelo = smf.ols(formula = 'salary_in_usd~salary+remote_ratio+employment_type_CT+experience_level_EN+experience_level_EX+experience_level

modelo = modelo.fit()

print(modelo.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          salary_in_usd   R-squared:                       0.259
Model:                            OLS   Adj. R-squared:                  0.250
Method:                 Least Squares   F-statistic:                     27.91
Date:                Fri, 18 Aug 2023   Prob (F-statistic):           1.33e-28
Time:                        03:15:41   Log-Likelihood:                -628.64
No. Observations:                 485   AIC:                             1271.
Df Residuals:                     478   BIC:                             1301.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0183      0.040      0.452      0.652      -0.061       0.098
salary        -0.1419      0.065     -2.194      0.029      -0.269      -0.015
remote_ratio   0.0564      0.040      1.394      0.164      -0.023       0.136
```

```
employment_type_CT      0.0549      0.041      1.344      0.180      -0.025       0.135
experience_level_EN    -0.3884      0.043     -9.106      0.000      -0.472      -0.305
experience_level_EX     0.1905      0.041      4.699      0.000       0.111       0.270
experience_level_MI    -0.3246      0.044     -7.431      0.000      -0.410      -0.239
==============================================================================
Omnibus:                      240.517   Durbin-Watson:                   1.983
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1966.204
Skew:                           1.990   Prob(JB):                         0.00
Kurtosis:                      12.026   Cond. No.                         1.89
==============================================================================

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

Como podemos observar, ahora sí que cambió el valor de r^2, pero disminuyó, así que nos quedamos con el segundo modelo porque siempre preferimos tener un modelo con el mayor valor de r^2 posible y la ecuacion más simple (con menos variables), que uno más complejo.

Así que nuestra ecuación de predicción de y sería así:
$y = 0 - 0.1455x1 + 0.0591x2 + 0.0990x3 + 0x4 - 0.3713x5 + 0.1903x6 - 0.3225x7$

```
y_aprox = 0 - 0.1455*prueba['salary'] + 0.0591*prueba['remote_ratio'] + 0.0990*prueba['employment_type_CT'] + 0 - 0.3713*prueba['experier
```

```
y_aprox
```

```
563    0.400308
289    0.400803
76    -0.271638
78     0.807652
182   -0.409579
         ...
249    0.397503
365    0.400464
453   -0.273524
548    0.404193
235   -0.417877
Length: 122, dtype: float64
```

- Creamos una tabla para comparar las predicciones con los datos reales y así medir tambien su nivel de error.

```
tabla = pd.DataFrame({'Real' : prueba['salary_in_usd'], 'Prediccion' : y_aprox, 'Errores' : prueba['salary_in_usd']-y_aprox})
```

```
tabla
```

	Real	Prediccion	Errores
563	0.394254	0.400308	-0.006054
289	0.320205	0.400803	-0.080599
76	-0.173457	-0.271638	0.098181
78	2.224328	0.807652	1.416676
182	-1.217128	-0.409579	-0.807549
...	...	...	...
249	0.813866	0.397503	0.416363
365	0.370981	0.400464	-0.029483
453	0.108636	-0.273524	0.382159
548	-0.186856	0.404193	-0.591049
235	-0.032411	-0.417877	0.385466

122 rows × 3 columns

Graficación

```
import matplotlib.pyplot as plt
```

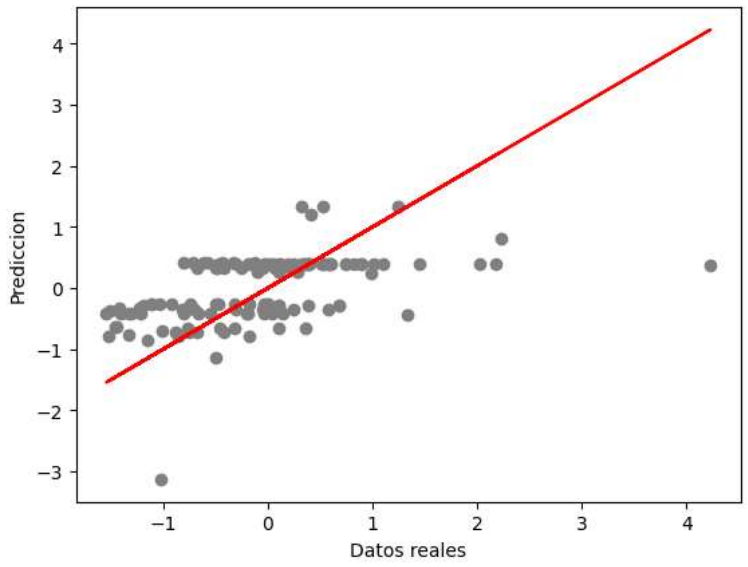
```
plt.scatter(prueba['salary_in_usd'], y_aprox, color = 'gray')
```

```
plt.plot(prueba['salary_in_usd'], prueba['salary_in_usd'], color = 'red')
```

```
plt.xlabel("Datos reales")
```

```
plt.ylabel("Prediccion")
```

```
Text(0, 0.5, 'Prediccion')
```

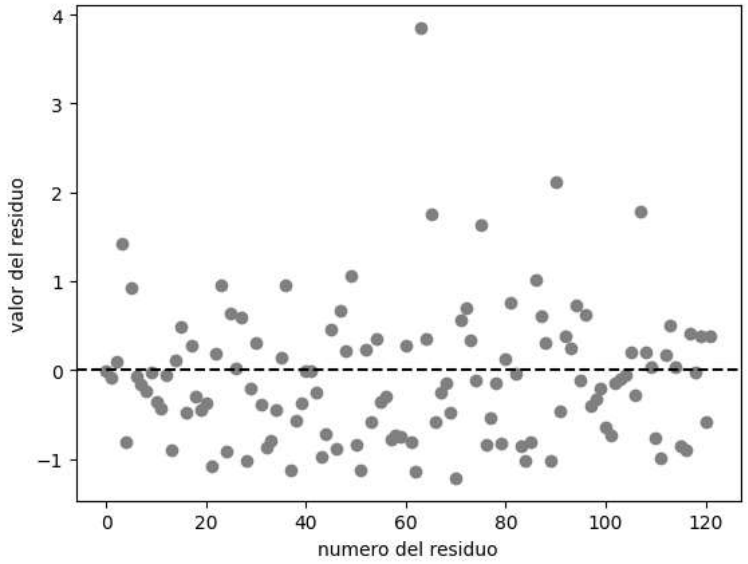


En base a esta grafica de puntos y comparandola con la linea recta, nos podemos dar cuenta que nuestra prediccion y los datos reales no son muy lineales. Lo cual es una señal de que nuestro modelo no es muy bueno.

```
l_residuos = len(tabla['Errores'])
```

```
plt.scatter(range(l_residuos), tabla['Errores'], color = 'gray')
plt.axhline(y = 0, linestyle = '--', color = 'black')
plt.xlabel("numero del residuo")
plt.ylabel("valor del residuo")
```

```
Text(0, 0.5, 'valor del residuo')
```



Podemos notar que nuestros valores de error se encuentran algo dispersos y algunos bastante lejos del cero.

```
plt.hist(x = tabla['Errores'], color = 'orange')
plt.title("Histograma residuos")
plt.xlabel("Residuos")
plt.ylabel("Frecuencia (proba)")
```

```
Text(0, 0.5, 'Frecuencia (proba)')
```

En este histograma podemos observar que no es exactamente una distribucion normal estandar. De hecho, se parece más a una distribución asimetrica positiva, aun así puede que sí sea una distribución normal y a continuación haremos una prueba para verificar su normalidad.

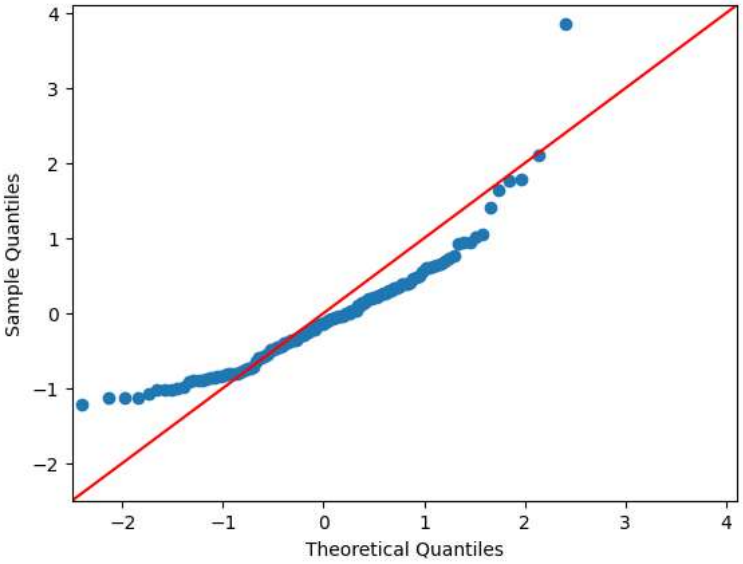
```
media = tabla['Errores'].mean()
std = tabla['Errores'].std()
errores_est = (tabla['Errores'] - media)/std
```

```
import statsmodels.api as sm
from scipy import stats
stats.kstest(errores_est, 'norm')
```

```
KstestResult(statistic=0.07811199222152843, pvalue=0.42455282141846074, statistic_location=0.5982597821639776, statistic_sign=1)
```

Por medio de la prueba de Kolmogorov Smirnov, analizamos el p-valor que si es mayor a 0.05, se puede concluir que la distribución es normal, en caso de que sea menor a 0.05, se conluye que la distribución no es normal. En nuestro caso, el p-valor es 0.4255, por lo que la prueba nos dice que los datos están distribuidos de manera normal.

```
QQ = sm.qqplot(tabla['Errores'], stats.norm, line = '45')
```



Graficamos los cuartiles teoricos y los cuartiles empiricos y nos damos cuenta que mantienen cierta relación lineal.

Conclusiones

Resumiendo la evidencia:

- $r^2 = 0.264$, que nos indica la veracidad del modelo, teniendo a 1 como el mayor nivel de veracidad y a 0 como el menor nivel de veracidad.
- La falta de linealidad de los datos reales con las predicciones.
- La dispersión algo alta de los errores en comparación al cero.
- El histograma de residuos nos indica que la distribución podría ser normal.
- La prueba de Kolmogorov Smirnov nos indica que efectivamente es una distribución normal de los datos.
- La grafica QQ-Plot nos indica que los cuartiles teoricos y empiricos sí mantienen una relación lineal.

Podemos inferir con todos estos datos que el modelo de regresión lineal con nuestro dataset no es el mejor modelo que podríamos usar. Sobre todo por su baja veracidad, no es un modelo confiable.

Credito:

- A00227694
- Siddhatha López Valenzuela