

P P1. Regresión

Luis Enrique Garcia Gallegos

Matricula: 649247

En este proyecto estarás trabajando con datos de tu interés, ya sea que los descargues de alguna base de datos pública o que los hayas generado por tu cuenta. Desarrolla los siguientes puntos en una *Jupyter Notebook*, de forma que se genere un reporte fácilmente comprensible, interpretable y replicable, con información para el lector descrita en formato de markdown. Indica claramente de dónde descargaste los datos, y los objetivos específicos que se desean alcanzar con el proyecto (e.g. identificar si a partir de información relacionada a la calidad del aire puedo predecir la temperatura, y caracterizar la relación que existe entre la cantidad de partículas pm25 y la temperatura).

1. Importa los datos a tu ambiente de trabajo. Describe características de tu base de datos, imprimiendo en consola y graficando información relevante. Por ejemplo: cantidad de observaciones, cantidad de variables, tipo de variables, nombre de variables, identificación de la respuesta de interés, etc. Asegúrate de que tus variables tengan un nombre fácilmente interpretable. En caso contrario, adjunta también un diccionario, un archivo .docx, .xlsx, .pdf, .csv, etc. donde se describa cada variable (por ejemplo: var1. Peso medido en gramos, var2. Diámetro medido en centímetros). Explica claramente qué de esta información te permitió suponer que realizar una regresión lineal pudiera ser una buena solución y/o por qué supones que otro método sería una mejor opción.

Información sobre cada columna -> [EXCEL](#).

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as st
import mlxtend.feature_selection as mlx
import random
from math import sqrt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor # Usa KNeighborsClassifier si es
from sklearn.metrics import mean_squared_error # Usa accuracy_score si es clasifica
BaseDeDatos=pd.read_csv('INE_SECCION_2020.csv')
print("Dimensiones de los datos: ",BaseDeDatos.shape, "\nNombres de las variables:
BaseDeDatos.head(5)
```

```
Dimensiones de los datos: (68806, 226)
Nombres de las variables: Index(['ID', 'ENTIDAD', 'DISTRITO', 'MUNICIPIO', 'SECCION', 'TIPO', 'POBTOT', 'POBFEM', 'POBMAS', 'P_0A2', ...
                             'VPH_TELEF', 'VPH_CEL', 'VPH_INTER', 'VPH_STVP', 'VPH_SPMVPI', 'VPH_CVJ', 'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCIN', 'VPH_SINTIC'],
                             dtype='object', length=226)
```

```
Out[1]:
```

	ID	ENTIDAD	DISTRITO	MUNICIPIO	SECCION	TIPO	POBTOT	POBFEM	POBMAS	P
0	118	1	3	1	1	2	2564	1360	1204	
1	122	1	3	1	2	2	889	462	427	
2	128	1	3	1	3	2	2003	1057	946	
3	137	1	3	1	4	2	1636	880	756	
4	138	1	3	1	5	2	808	445	363	

5 rows × 226 columns



Se tomaron los datos del INEGI de su base de datos para el publico, específicamente de **Estadísticas censales a escalas geoelectorales (2020)**, dicho apartado era una archivo *ZIP*, en el que venia mucha información, solamente se tomaron los arquivos adjuntos ([Descripción de columnas de la base de datos](#), [Base de datos](#), [CLASIFICACIONES DEL CENSO DE POBLACIÓN Y VIVIENDA 2020](#)).

2. Aplica soluciones para al menos tres de los cinco problemas típicos que se describieron en la presentación C1.5. Demuestra las modificaciones que se le realizaron a la base de datos, imprimiendo en consola información relevante, y explica por qué se realizaron. Por ejemplo: para el caso de variables cualitativas, podrías imprimir la cantidad y los nombres de las variables antes de aplicar la solución, y la cantidad y los nombres de las variables después de aplicar la solución.

```
In [2]: datos=BaseDeDatos.drop(columns=['ID', 'DISTRITO', 'MUNICIPIO', 'SECCION', 'POBFEM',
print(f"Entidades en México\n{datos['ENTIDAD'].value_counts()}")
print(f"\nTipos (2. Urbana\t3. Mixta\t4. Rural)\n{datos['TIPO'].value_counts()}\t")
datos['RURAL']=datos['TIPO']//4
datos['MIXTA']=(datos['TIPO']//3)-datos['RURAL']
for i in range(32, 1, -1):
    name='ENTIDAD'+str(i)
    datos[name]=datos['ENTIDAD']//i
    existe=datos[name]*i
    datos['ENTIDAD']=datos['ENTIDAD']-existe
datos=datos.drop(columns=['TIPO', 'ENTIDAD'])
datos.head(5)
```

Entidades en México

ENTIDAD

15	6544
9	5535
30	4865
25	3765
14	3613
8	3195
11	3161
19	2818
16	2703
12	2685
21	2684
20	2479
7	2099
28	2017
2	2016
24	1793
32	1777
13	1762
5	1701
26	1528
10	1369
27	1144
31	1122
23	1038
18	968
17	914
22	891
1	622
29	612
4	532
3	484
6	370

Name: count, dtype: int64

Tipos (2. Urbana 3. Mixta 4. Rural)

TIPO

2	43815
4	18605
3	6386

Name: count, dtype: int64

Out[2]:

	POBTOT	P_0A2	P3A5_NOA	P6A11_NOA	P12A14NOA	P15A17A	P18A24A	P8A14AN
0	2564	36	8	5	0	98	203	2
1	889	27	9	6	4	26	46	3
2	2003	56	10	5	9	72	170	1
3	1636	40	17	3	5	50	158	1
4	808	14	4	3	0	34	66	1

5 rows × 48 columns



Como esta base de datos incluye demasiada información, se descarto toda aquella que fuera referente a otro tema respecto a la educación y a la `ENTIDAD`, esto porque no lo senti relevante para este análisis de conocer la población de niños de entre 3 a 5 años de edad que no van a la escuela. Tanto la variable `TIPO` y `ENTIDAD`, son cualitativas para el `TIPO` existen 3 categorías **2. Urbana**, **3. Mixta** y **4. Rural**, mientras que `ENTIDAD` hace referencia a los **32 estados de México**; considerando esto se crearon columnas binarias, por ejemplo si tenemos dos datos en `ENTIDAD` que fueran **10** y **11** y creáramos una columna que fuera `ENTIDAD11`, entonces nuestro datos pasarían **0** y **1**, respectivamente. Gran parte de este procesos fue logrado mediante la extension de VisualCode [Data Wrangler](#), la cual permitió visualizar los datos y realizar operaciones básicas en las columnas facilitando la codificación y limpieza de los datos.

3. Realiza un proceso de selección de características. Puedes llevar a cabo una metodología de selección hacia adelante o eliminación hacia atrás, o incluso mezclarlas. También puedes utilizar un método de regularización. Evidencia tus resultados imprimiendo en consola la cantidad y los nombres de las variables antes y después del proceso de selección de características. Explica claramente por qué utilizaste la metodología empleada, así como alguna conclusión sobre los resultados de este proceso.

In [3]:

```

random.seed(0)
caliz, prueba=train_test_split(datos, train_size=0.8)
nC=caliz.shape[0]
mC=caliz.shape[1]
nP=prueba.shape[0]
mP=prueba.shape[1]
print("Datos de entrenamiento: ", caliz.shape, "\nDatos de prueba: ",prueba.shape,
XC=caliz.drop('P3A5_NOA', axis=1)
YC=caliz['P3A5_NOA']
XP=prueba.drop('P3A5_NOA', axis=1)
YP=prueba['P3A5_NOA']
sfs=mlx.SequentialFeatureSelector(LinearRegression(), k_features=(30, (mP-1)), forw
sfs.fit(XC,YC)
variables=sfs.k_feature_names_
print(f"Características:\n\t{caliz.columns}\nCaracterísticas seleccionadas:\n\t{var

```

```

xAjustadoC=XC
xAjustadoP=XP
for i in range(len(variables)):
    xAjustadoC=xAjustadoC.drop(variables[i], axis=1)
borrar=xAjustadoC.columns
xAjustadoC=XC
for i in range(len(borrar)):
    xAjustadoC=xAjustadoC.drop(borrar[i], axis=1)
    xAjustadoP=xAjustadoP.drop(borrar[i], axis=1)

```

Datos de entrenamiento: (55044, 48) Datos de prueba: (13762, 48)
 Total de datos: 68806

Características:

```

Index(['POBTOT', 'P_0A2', 'P3A5_NOA', 'P6A11_NOA', 'P12A14NOA', 'P15A17A',
      'P18A24A', 'P8A14AN', 'P15YM_AN', 'P15YM_SE', 'P15PRI_IN', 'P15PRI_CO',
      'P15SEC_IN', 'P15SEC_CO', 'P18YM_PB', 'RURAL', 'MIXTA', 'ENTIDAD32',
      'ENTIDAD31', 'ENTIDAD30', 'ENTIDAD29', 'ENTIDAD28', 'ENTIDAD27',
      'ENTIDAD26', 'ENTIDAD25', 'ENTIDAD24', 'ENTIDAD23', 'ENTIDAD22',
      'ENTIDAD21', 'ENTIDAD20', 'ENTIDAD19', 'ENTIDAD18', 'ENTIDAD17',
      'ENTIDAD16', 'ENTIDAD15', 'ENTIDAD14', 'ENTIDAD13', 'ENTIDAD12',
      'ENTIDAD11', 'ENTIDAD10', 'ENTIDAD9', 'ENTIDAD8', 'ENTIDAD7',
      'ENTIDAD6', 'ENTIDAD5', 'ENTIDAD4', 'ENTIDAD3', 'ENTIDAD2'],
      dtype='object')

```

Características seleccionadas:

```

('POBTOT', 'P_0A2', 'P6A11_NOA', 'P12A14NOA', 'P15A17A', 'P18A24A', 'P8A14A
N', 'P15YM_AN', 'P15YM_SE', 'P15PRI_IN', 'P15PRI_CO', 'P15SEC_CO', 'P18YM_PB', 'RURA
L', 'MIXTA', 'ENTIDAD32', 'ENTIDAD31', 'ENTIDAD30', 'ENTIDAD29', 'ENTIDAD28', 'ENTID
AD27', 'ENTIDAD26', 'ENTIDAD24', 'ENTIDAD22', 'ENTIDAD21', 'ENTIDAD20', 'ENTIDAD19',
'ENTIDAD18', 'ENTIDAD17', 'ENTIDAD16', 'ENTIDAD15', 'ENTIDAD14', 'ENTIDAD12', 'ENTID
AD11', 'ENTIDAD9', 'ENTIDAD8', 'ENTIDAD7', 'ENTIDAD6', 'ENTIDAD5', 'ENTIDAD4', 'ENTI
DAD3', 'ENTIDAD2')

```

Se uso una semilla para mantener un punto fijo a la hora de hacer una partición de datos entre prueba y entrenamiento, en el cual por gusto propio decidí hacer una partición de 80 para entrenamiento y el resto para las pruebas. Se opto por llevar una metodología de selección de características hacia atrás, ya que existen muchas variables *dummies* por lo que si vamos hacia adelante, vamos a tener que hacer más iteraciones, si bien yo fije un aproximado de cuantas variables voy a usar esto es con el fin de limitar el tiempo de cálculos, no obstante se uso validación cruzada.

4. Genera un modelo de regresión lineal y al menos uno no lineal, para predecir una variable de interés; explica con detalle por qué seguiste los pasos mostrados en tu código para generar el modelo, en aras de obtener la mejor predicción y resultados generalizables. Imprime en consola los coeficientes estimados al entrenar el modelo, e indica para cuáles de ellos puedes afirmar que existe una asociación significativa con la respuesta, y el por qué de tu aseveración.

```

In [4]: modeloML=sm.OLS(YC, sm.add_constant(xAjustadoC))
resultadosML=modeloML.fit()
resumenML=resumenML.summary()
print(resumenML)
scaler = StandardScaler()

```

```
xAjustadoCKNN=scaler.fit_transform(xAjustadoC)
xAjustadoPKNN=scaler.transform(xAjustadoP)
mse_scores={}
for k in range(1, 101):
    knn=KNeighborsRegressor(n_neighbors=k)
    knn.fit(xAjustadoCKNN, YC)
    y_pred=knn.predict(xAjustadoPKNN)
    mse_scores[k]=mean_squared_error(list(YP), y_pred)
best_k=min(mse_scores, key=mse_scores.get)
modeloKNN=KNeighborsRegressor(n_neighbors=best_k)
modeloKNN.fit(xAjustadoCKNN, YC)
print(knn.get_params())
```

OLS Regression Results

```

=====
Dep. Variable:          P3A5_NOA      R-squared:                0.943
Model:                  OLS           Adj. R-squared:           0.943
Method:                 Least Squares F-statistic:              2.175e+04
Date:                   Tue, 18 Feb 2025 Prob (F-statistic):        0.00
Time:                   13:45:57      Log-Likelihood:          -2.2240e+05
No. Observations:       55044         AIC:                    4.449e+05
Df Residuals:           55001         BIC:                    4.453e+05
Df Model:                42
Covariance Type:        nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
const	1.0743	0.184	5.844	0.000	0.714	1.435
POBTOT	0.0255	0.001	39.899	0.000	0.024	0.027
P_0A2	0.2287	0.003	78.443	0.000	0.223	0.234
P6A11_NOA	0.5609	0.009	62.848	0.000	0.543	0.578
P12A14NOA	0.2276	0.011	21.244	0.000	0.207	0.249
P15A17A	-0.0956	0.003	-28.661	0.000	-0.102	-0.089
P18A24A	-0.0229	0.002	-10.462	0.000	-0.027	-0.019
P8A14AN	0.1157	0.009	13.035	0.000	0.098	0.133
P15YM_AN	-0.0390	0.002	-22.019	0.000	-0.042	-0.036
P15YM_SE	-0.0216	0.001	-15.788	0.000	-0.024	-0.019
P15PRI_IN	-0.0386	0.002	-25.472	0.000	-0.042	-0.036
P15PRI_CO	-0.0264	0.001	-21.516	0.000	-0.029	-0.024
P15SEC_CO	0.0092	0.001	9.980	0.000	0.007	0.011
P18YM_PB	-0.0321	0.001	-42.873	0.000	-0.034	-0.031
RURAL	-0.5111	0.159	-3.219	0.001	-0.822	-0.200
MIXTA	-3.0875	0.223	-13.862	0.000	-3.524	-2.651
ENTIDAD32	-4.9011	0.403	-12.162	0.000	-5.691	-4.111
ENTIDAD31	-7.4748	0.498	-15.007	0.000	-8.451	-6.499
ENTIDAD30	2.7942	0.287	9.740	0.000	2.232	3.357
ENTIDAD29	-3.0984	0.638	-4.854	0.000	-4.349	-1.847
ENTIDAD28	2.7666	0.383	7.227	0.000	2.016	3.517
ENTIDAD27	-16.1916	0.487	-33.272	0.000	-17.145	-15.238
ENTIDAD26	5.3405	0.428	12.473	0.000	4.501	6.180
ENTIDAD24	-10.8199	0.399	-27.144	0.000	-11.601	-10.039
ENTIDAD22	-2.8958	0.545	-5.317	0.000	-3.963	-1.828
ENTIDAD21	-8.6388	0.355	-24.336	0.000	-9.335	-7.943
ENTIDAD20	-6.1514	0.364	-16.918	0.000	-6.864	-5.439
ENTIDAD19	-4.0344	0.342	-11.790	0.000	-4.705	-3.364
ENTIDAD18	-2.2351	0.521	-4.286	0.000	-3.257	-1.213
ENTIDAD17	-2.0843	0.536	-3.887	0.000	-3.135	-1.033
ENTIDAD16	-5.5736	0.347	-16.072	0.000	-6.253	-4.894
ENTIDAD15	-0.9993	0.261	-3.827	0.000	-1.511	-0.487
ENTIDAD14	-1.9108	0.314	-6.093	0.000	-2.525	-1.296
ENTIDAD12	-9.3839	0.350	-26.842	0.000	-10.069	-8.699
ENTIDAD11	-6.3596	0.325	-19.588	0.000	-6.996	-5.723
ENTIDAD9	-5.0077	0.282	-17.743	0.000	-5.561	-4.454
ENTIDAD8	3.1107	0.322	9.668	0.000	2.480	3.741
ENTIDAD7	-7.0726	0.409	-17.281	0.000	-7.875	-6.270
ENTIDAD6	-3.9690	0.820	-4.839	0.000	-5.577	-2.361
ENTIDAD5	-1.5303	0.412	-3.718	0.000	-2.337	-0.724
ENTIDAD4	-9.3976	0.685	-13.723	0.000	-10.740	-8.055
ENTIDAD3	-0.8250	0.709	-1.163	0.245	-2.216	0.566

ENTIDAD2	5.1618	0.385	13.394	0.000	4.406	5.917
=====						
Omnibus:		28365.502	Durbin-Watson:			2.009
Prob(Omnibus):		0.000	Jarque-Bera (JB):		5783622.518	
Skew:		1.367	Prob(JB):			0.00
Kurtosis:		53.142	Cond. No.			5.32e+04
=====						

Notes:

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

[2] The condition number is large, 5.32e+04. This might indicate that there are strong multicollinearity or other numerical problems.

```
{'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 100, 'p': 2, 'weights': 'uniform'}
```

Como tenemos más de una variables no se podría utilizar un modelo de regresión lineal simple por lo que se usa el modelo de regresión lineal multiple. Asi mismo como tenemos más de una variable no sabemos que forma tendría función, por lo que un modelo KNN seria una manera de tener un modelo no lineal. Si analizamos el `summary()` del modelo lineal, veremos que tenemos una R^2 de **0.943**, el cual es bastante bueno, en cuanto a los **p-values** la mayoría son menores a **0.05**, no obstante hay algunos que superan estos valores, esto se puede deber a que existan pocos datos de esa columna por lo que en la mayoría de los casos no influye en el modelo.

5. Calcula al menos una métrica de error y una del nivel de linealidad del modelo, con la intención de que ambas métricas representen el funcionamiento esperado del modelo para cualquier dato que se le presente. Agrega un comentario que describa en palabras sencillas el significado de los mismos.

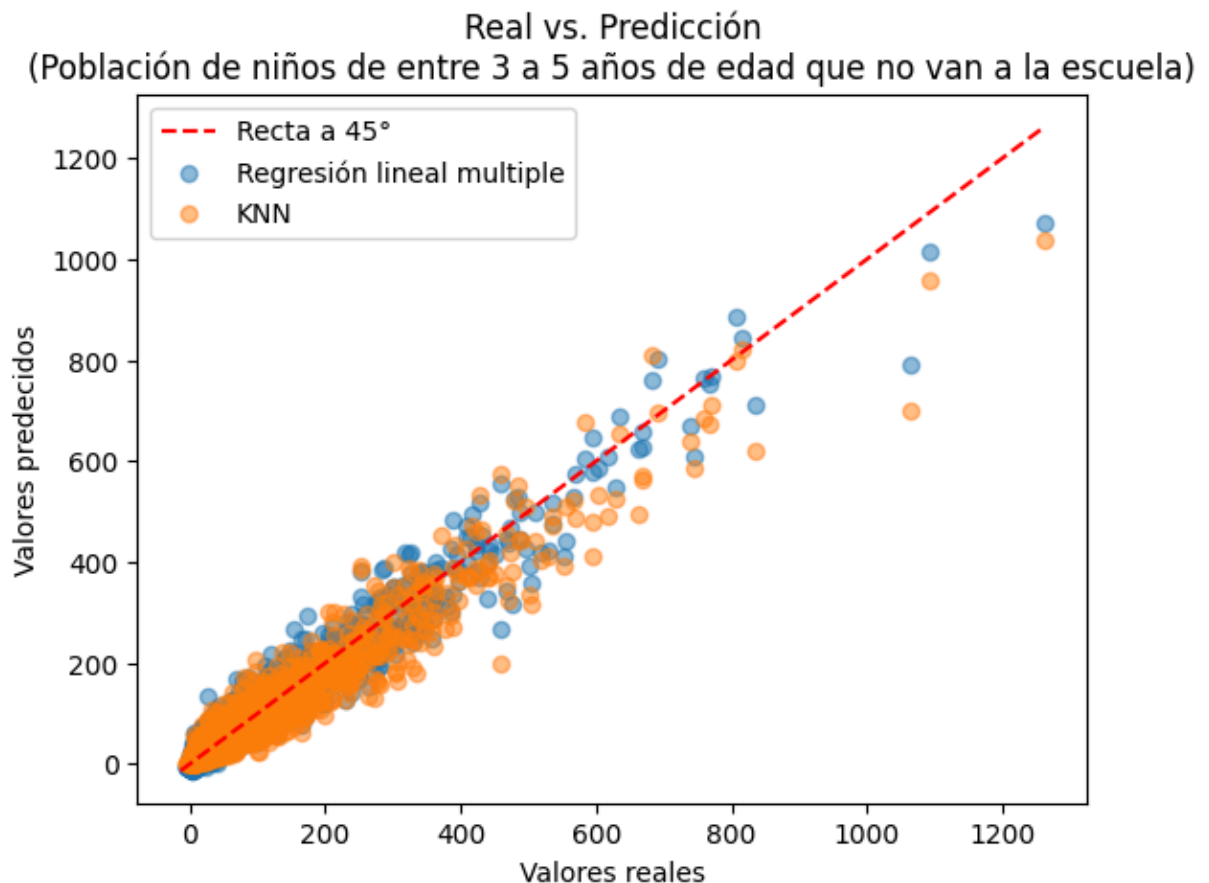
```
In [5]: estimacionML=resultadosML.predict(sm.add_constant(xAjustadoP))
estimacionKNN=modeloKNN.predict(xAjustadoPKNN)
mseKNN=mean_squared_error(estimacionKNN, YP)
mseML=mean_squared_error(estimacionML, YP)
print(f"MSE KNN: {round(mseKNN)}\tMSE regresión lineal multiple: {round(mseML)}")
```

MSE KNN: 233 MSE regresión lineal multiple: 189

Con los **MSE** de cada modelo podemos ver que tanto varían nuestros resultados entre lo previsto y lo real, como estamos prediciendo la población, entonces estos valores deben de ser enteros. Como ambos modelos tienen similares **MSE** si tomamos como referencia que la población más alta de niños de 3 a 5 años esta por los **1200**. Por lo tanto ambos modelos se adaptan bien a los datos.

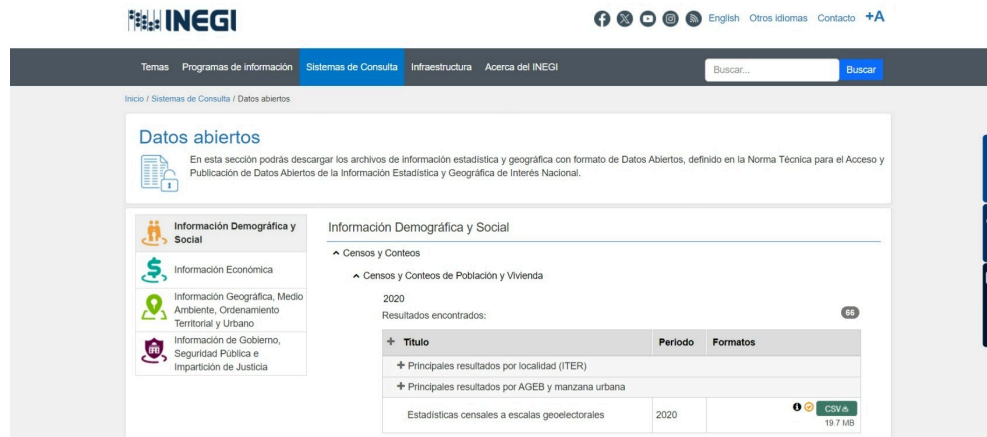
6. Genera un modelo de regresión lineal o no lineal (del mismo tipo que el que haya obtenido el mejor desempeño en los puntos anteriores), pero para realizar un análisis de inferencia. A partir del mismo, realiza una o varias inferencias sobre los datos, asegurándote de incluir márgenes de error para tus conclusiones.


```
In [6]: min_val=min(min(estimacionML), min(estimacionKNN), min(YP))
max_val=max(max(estimacionML), max(estimacionKNN), max(YP))
plt.plot([min_val, max_val], [min_val, max_val], color='red', linestyle='--')
plt.scatter(YP, estimacionML, alpha=0.5)
plt.scatter(YP, estimacionKNN, alpha=0.5)
plt.title('Real vs. Predicción\n(Población de niños de entre 3 a 5 años de edad que no van a la escuela)')
plt.xlabel("Valores reales")
plt.ylabel("Valores prededidos")
plt.legend(["Recta a 45°", "Regresión lineal multiple", "KNN"])
plt.show()
```



Se gráfico las predicciones de ambos modelos ya que al tener un **MSE** similar las predicciones serian similares, mediante ayuda de ChatGPT se logro graficar una recta a 45° la cual nos permitirá ver que tan buenas predicciones se hicieron con los datos de prueba.

7. (Opcional, valor de 10 puntos extra, solamente válido si utilizaste datos de <https://www.inegi.org.mx/datosabiertos/>)



INEGI





Temas Programas de Información **Sistemas de Consulta** Infraestructura Acerca del INEGI

Buscar... **Buscar**

Inicio / Sistemas de Consulta / Datos abiertos

Datos abiertos

En esta sección podrás descargar los archivos de información estadística y geográfica con formato de Datos Abiertos, definido en la Norma Técnica para el Acceso y Publicación de Datos Abiertos de la Información Estadística y Geográfica de Interés Nacional.

 **Información Demográfica y Social**
 Información Económica
 Información Geográfica, Medio Ambiente, Ordenamiento Territorial y Urbano
 Información de Gobierno, Seguridad Pública e Impartición de Justicia


Información Demográfica y Social

^ Censos y Conteos

^ Censos y Conteos de Población y Vivienda

2020

Resultados encontrados: 66

+	Título	Periodo	Formatos
+	Principales resultados por localidad (ITER)		
+	Principales resultados por AGEB y manzana urbana		
	Estadísticas censales a escalas geoelectorales	2020	 CSV 19.7 MB

Crea una presentación corta (aproximadamente 5 slides), y expón al grupo la metodología y resultados obtenidos, como si estuvieras presentando resultados a una entidad responsable de resolver problemáticas sociales. Sube la presentación en la bandeja de Blackboard junto con el resto de los documentos.

Firma de Honor: Doy mi palabra que he realizado esta actividad con integridad académica