Nombre del estudiante: Diego Alonso Luna Ramirez

Matricula: A01793035

Nombre del trabajo: Actividad Semanal - 7 Regresiones y K means

Fecha de entrega: 08 de noviembre del 2022

Campus: Querétaro

Programa: Maestría en Inteligencia Artificial Aplicada (MNA-V)

Trimestre: Segundo

Materia: Ciencia y analítica de datos

Nombre del maestro: Maria de la Paz Rico Fernandez

Actividad Semanal -- 7 Regresiones y K means

Diego Alonso Luna Ramirez - A01793035

Profesor(a): María de la Paz Rico

Fecha: 8 Noviembre de 2022

Linear Models

- In supervised learning, the training data fed to the algorithm includes the desired solutions, called labels.
- In **regression**, the labels are continuous quantities.
- Linear models predict by computing a weighted sum of input features plus a bias term.

```
import numpy as np
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
# to make this notebook's output stable across runs
np.random.seed(42)
```

```
5-2
```

```
     3
```

Simple Linear Regression

Simple linear regression equation:

$y = ax + b$

a: slope

b: intercept

Generate linear-looking data with the equation:

$y = 3X + 4 + noise$

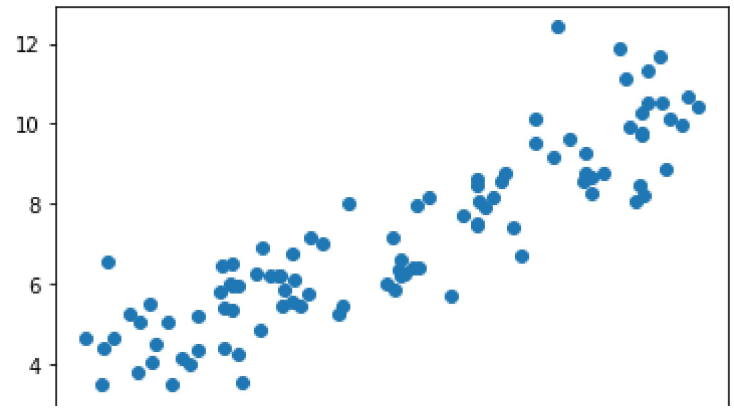
```
np.random.rand(100, 1)
```

```
     array([[0.37454012],
```

```
       [0.95071431],
       [0.73199394],
       [0.59865848],
       [0.15601864],
       [0.15599452],
       [0.05808361],
       [0.86617615],
       [0.60111501],
       [0.70807258],
       [0.02058449],
       [0.96990985],
       [0.83244264],
       [0.21233911],
       [0.18182497],
       [0.18340451],
       [0.30424224],
       [0.52475643],
       [0.43194502],
       [0.29122914],
       [0.61185289],
       [0.13949386],
       [0.29214465],
       [0.36636184],
       [0.45606998],
       [0.78517596],
       [0.19967378],
       [0.51423444],
       [0.59241457],
       [0.04645041],
       [0.60754485],
       [0.17052412],
       [0.06505159],
       [0.94888554],
       [0.96563203],
       [0.80839735],
       [0.30461377],
       [0.09767211],
       [0.68423303],
       [0.44015249],
       [0.12203823],
       [0.49517691],
       [0.03438852],
       [0.9093204 ],
       [0.25877998],
       [0.66252228],
       [0.31171108],
       [0.52006802],
       [0.54671028],
       [0.18485446],
       [0.96958463],
       [0.77513282],
       [0.93949894],
       [0.89482735],
       [0.59789998],
       [0.92187424],
       [0.0884925 ],
       [0.19598286],
```

```
X = 2*np.random.rand(100, 1)
```

```
y = 4 + 3 * X + np.random.randn(100, 1)
plt.scatter(X, y);
```




```
import pandas as pd
pd.DataFrame(y)
```

	0
0	3.508550
1	8.050716
2	6.179208
3	6.337073
4	11.311173
...	...
95	5.441928
96	10.121188
97	9.787643
98	8.061635

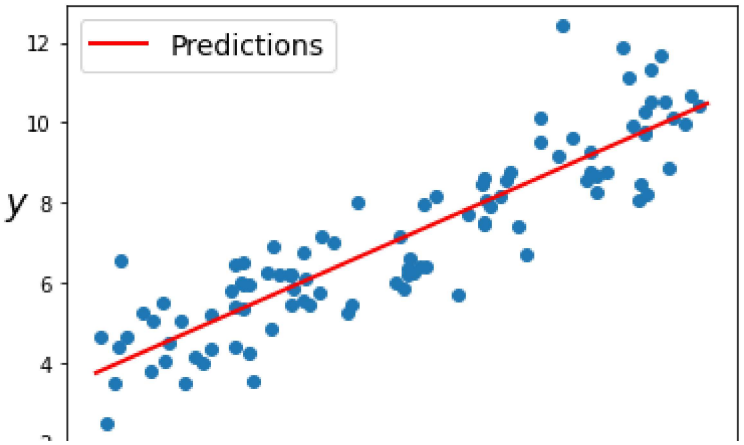
```
from sklearn.linear_model import LinearRegression

linear_reg = LinearRegression(fit_intercept=True)
linear_reg.fit(X, y)
```

```
LinearRegression()
```

Plot the model's predictions:

```
#X_fit[]
```

```
# construct best fit line
X_fit = np.linspace(0, 2, 100)
y_fit = linear_reg.predict(X_fit[:, np.newaxis])

plt.scatter(X, y)
plt.plot(X_fit, y_fit, "r-", linewidth=2, label="Predictions")
plt.xlabel("$X$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.legend(loc="upper left", fontsize=14);
```



Predictions are a good fit.

Generate new data to make predictions with the model:

```
X_new = np.array([[0], [2]])
X_new
```

```
array([[0],
       [2]])
```

```
X_new.shape
```

```
(2, 1)
```

```
y_new = linear_reg.predict(X_new)
y_new
```

```
array([[ 3.74406122],
       [10.47517611]])
```

```
 linear_reg.coef_, linear_reg.intercept_
```

```
(array([[3.36555744]]), array([3.74406122]))
```

The model estimates:

$$\hat{y} = 3.36X + 3.74$$

```
#|VENTAS|GANANCIAS|
#COEF*VENTAS+B
#|VENTAS|COMPRAS|GANANCIAS|
#COEF1*X1+COEF2*X2+B=Y
```

Polynomial Regression

If data is more complex than a straight line, you can use a linear model ti fit non-linear data adding powers of each feature as new features and then train a linear model on the extended set of features.

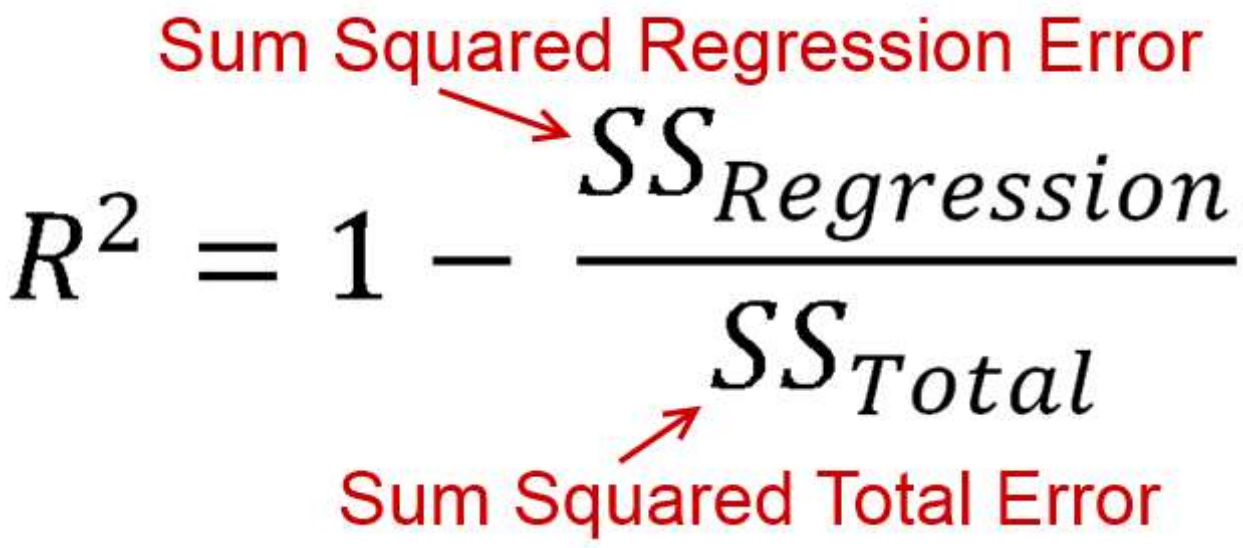
[ ] ↳ *18 celdas ocultas*

R square

R² es una medida estadística de qué tan cerca están los datos de la línea de regresión ajustada. También se conoce como el coeficiente de determinación o el coeficiente de determinación múltiple para la regresión múltiple. Para decirlo en un lenguaje más simple, R² es una medida de ajuste para los modelos de regresión lineal.

R² no indica si un modelo de regresión se ajusta adecuadamente a sus datos. Un buen modelo puede tener un valor R² bajo. Por otro lado, un modelo sesgado puede tener un valor alto de R².

SSres + SSreg = SStot, R² = Explained variation / Total Variation



$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \Rightarrow 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

$$R^2 = \frac{SS_{\text{reg}}}{SS_{\text{tot}}}$$

Ejercicio 1

Utiliza la base de datos de https://www.kaggle.com/vinicius150987/manufacturing-cost

Suponga que trabaja como consultor de una empresa de nueva creación que busca desarrollar un modelo para estimar el costo de los bienes vendidos a medida que varían el volumen de producción (número de unidades producidas). La startup recopiló datos y le pidió que desarrollara un modelo para predecir su costo frente a la cantidad de unidades vendidas.

[ ] ↳ *57 celdas ocultas*

Ejercicio 2

Realiza la regresión polinomial de los siguientes datos:

[ ] ↳ *43 celdas ocultas*

Actividad Semanal -- 7 Regresiones y K means

Diego Alonso Luna Ramirez - A01793035

Profesor(a): María de la Paz Rico

Fecha: 8 Noviembre de 2022

Este notebook se basa en información de target

Ahora imagina que somos parte del equipo de data science de la empresa Target, una de las tiendas con mayor presencia en Estados Unidos. El departamento de logistica acude a nosotros para saber donde le conviene poner sus almacenes, para que se optimice el gasto de gasolina, los tiempos de entrega de los productos y se disminuyan costos. Para ello, nos pasan los datos de latitud y longitud de cada una de las tiendas.

https://www.kaggle.com/datasets/saejinmahlauheinert/target-store-locations?select=target-locations.csv

Si quieres saber un poco más de graficas geográficas consulta el siguiente notebook https://colab.research.google.com/github/QuantEcon/quantecon-notebooks-datascience/blob/master/applications/maps.ipynb#scrollTo=uo2oPtSCeAOz

```
! pip install qeds fiona geopandas xgboost gensim folium pyLDAvis descartes
```

```
    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
```

```
Requirement already satisfied: qeds in /usr/local/lib/python3.7/dist-packages (0.7
Requirement already satisfied: fiona in /usr/local/lib/python3.7/dist-packages (1.
Requirement already satisfied: geopandas in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: xgboost in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: gensim in /usr/local/lib/python3.7/dist-packages (3
Requirement already satisfied: folium in /usr/local/lib/python3.7/dist-packages (0
Requirement already satisfied: pyLDAvis in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: descartes in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: pyarrow in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: quandl in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pandas-datareader in /usr/local/lib/python3.7/dist-
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: quantecon in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: openpyxl in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: munch in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pyproj>=2.2.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: sklearn in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: funcy in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: numexpr in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/lo
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/loc
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: inflection>=0.3.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (fr
```

```
import pandas as pd
```

```
import numpy as np
from tqdm import tqdm
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import geopandas
import seaborn as sb
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

Importa la base de datos

```
url="https://raw.githubusercontent.com/marypazrf/bdd/main/target-locations.csv"
df=pd.read_csv(url)
```

Exploremos los datos.

```
df.head()
```

	name	latitude	longitude	address	phone	webs
0	Alabaster	33.224225	-86.804174	250 S Colonial Dr, Alabaster, AL 35007-4657	205-564-2608	https://www.target.com/sl/alabaster/2
				4889 Promenade Pkwy	205-	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1839 entries, 0 to 1838
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   name       1839 non-null   object
 1   latitude   1839 non-null   float64
 2   longitude  1839 non-null   float64
 3   address    1839 non-null   object
 4   phone      1839 non-null   object
 5   website    1839 non-null   object
dtypes: float64(2), object(4)
memory usage: 86.3+ KB
```

Definición de Latitud y Longitud

Latitud Es la distancia en grados, minutos y segundos que hay con respecto al paralelo principal, que es el ecuador (0º). La latitud puede ser norte y sur.

Longitud: Es la distancia en grados, minutos y segundos que hay con respecto al meridiano principal, que es el meridiano de Greenwich (0º).La longitud puede ser este y oeste.

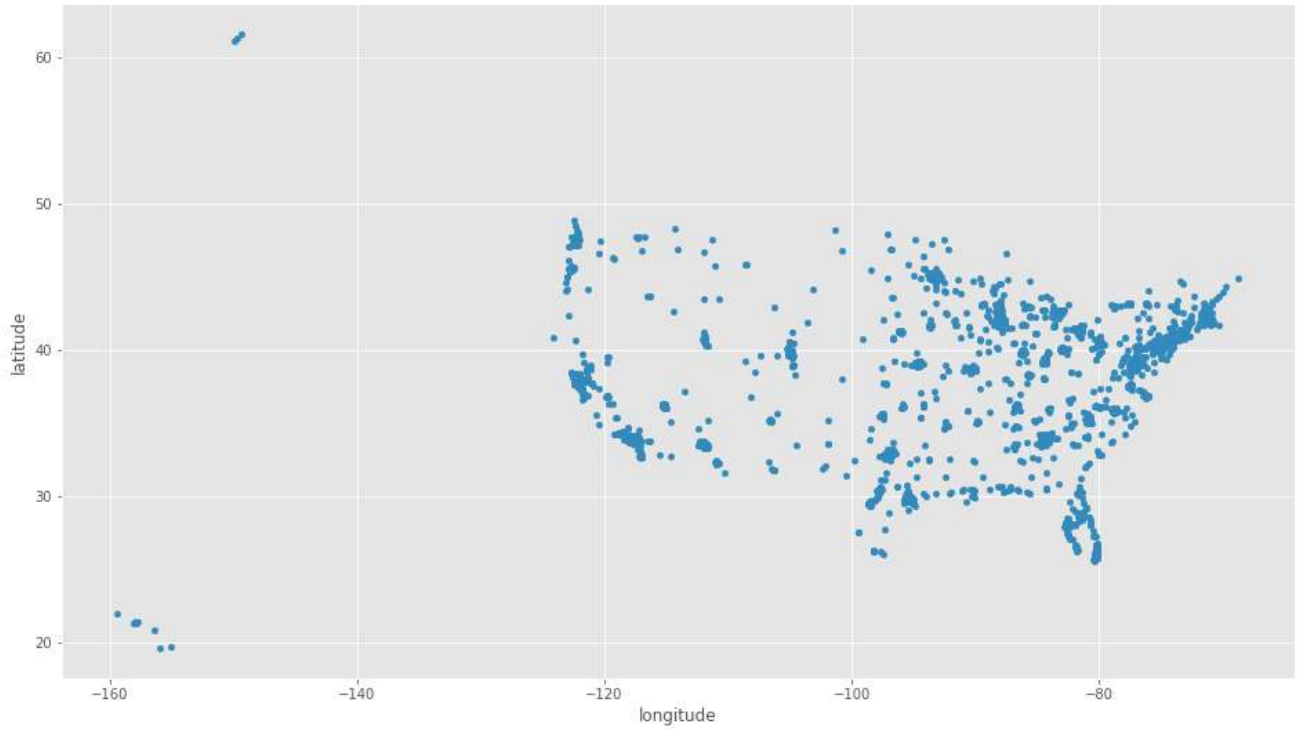
```
latlong=df[["latitude","longitude"]]
```

¡Visualizemos los datos!, para empezar a notar algún patron.

A simple vista pudieramos pensar que tenemos algunos datos atípicos u outliers, pero .... no es así, simplemente esta grafica no nos está dando toda la información.

```
#extrae los datos interesantes
latlong.plot.scatter( "longitude","latitude")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f849e853650>
```



```
latlong.describe()
```

	latitude	longitude
count	1839.000000	1839.000000
mean	37.791238	-91.986881
std	5.272299	16.108046
min	19.647855	-159.376962
25%	33.882605	-98.268828
50%	38.955432	-87.746346
75%	41.658341	-80.084833
max	61.577919	-68.742331

Para entender un poco más, nos auxiliaremos de una librería para graficar datos geográficos. Esto nos ayudara a tener un mejor entendimiento de ellos.

```
import geopandas as gpd
import matplotlib.pyplot as plt
import pandas as pd

from shapely.geometry import Point

%matplotlib inline
# activate plot theme
import qeds
qeds.themes.mpl_style();
```

```
df["Coordinates"] = list(zip(df.longitude, df.latitude))
df["Coordinates"] = df["Coordinates"].apply(Point)
df.head()
```

	name	latitude	longitude	address	phone	webs
0	Alabaster	33.224225	-86.804174	250 S Colonial Dr, Alabaster, AL 35007-4657	205-564-2608	https://www.target.com/sl/alabaster/2
				4889 Promenade Pkwy	205-	

```
gdf = gpd.GeoDataFrame(df, geometry="Coordinates")
gdf.head()
```

	name	latitude	longitude	address	phone	webs
0	Alabaster	33.224225	-86.804174	250 S Colonial Dr, Alabaster, AL 35007-4657	205-564-2608	https://www.target.com/sl/alabaster/2
				4889 Promenade		

```
#mapa

world = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))
world = world.set_index("iso_a3")

world.head()
```

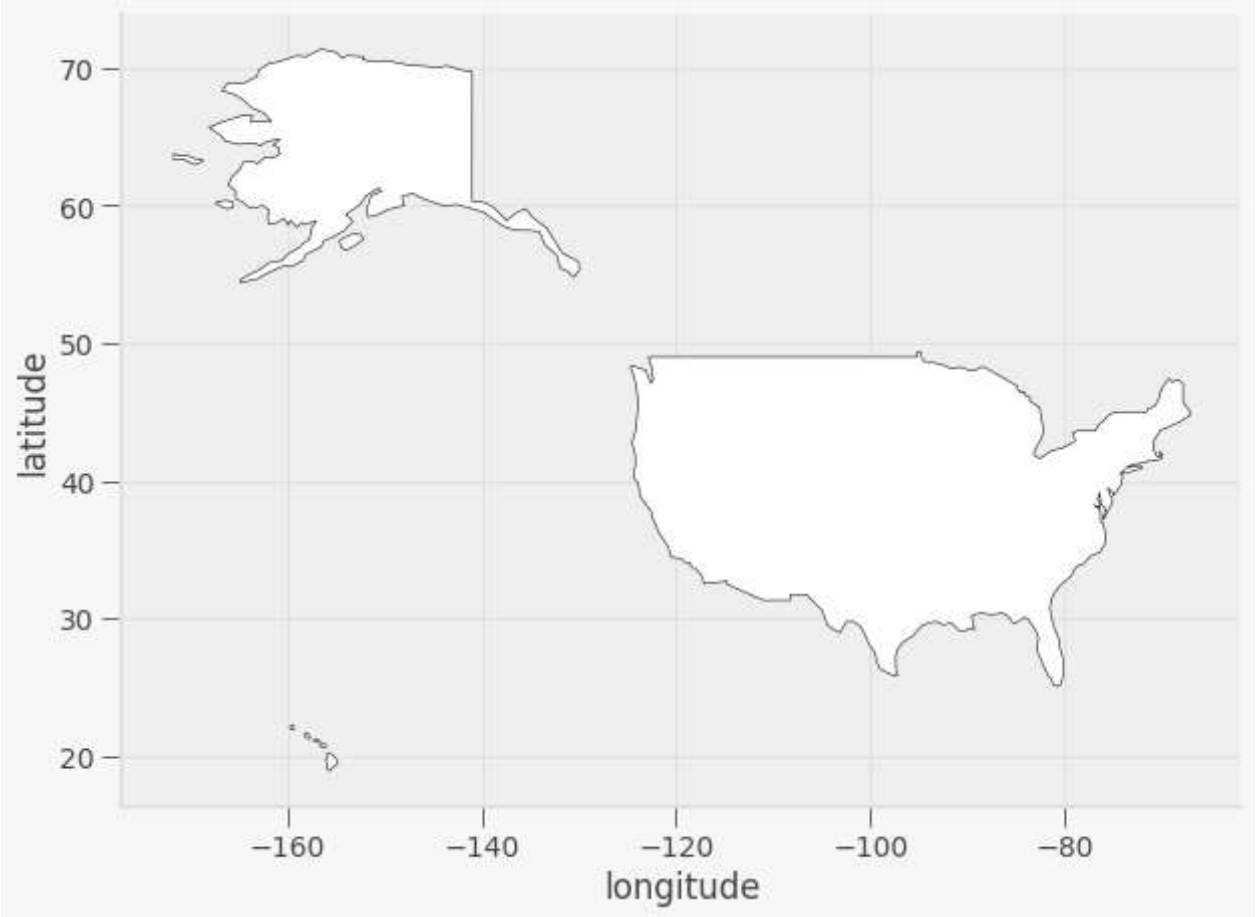
iso_a3	pop_est	continent	name	gdp_md_est	geometry
FJI	920938	Oceania	Fiji	8374.0	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...
TZA	53950935	Africa	Tanzania	150600.0	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...
ESH	603253	Africa	W. Sahara	906.5	POLYGON ((-8.66559 27.65643, -8.66512 27.58948...
		North			MULTIPOLYGON (((-122.84000

```
#graficar el mapa
world.name.unique()
```

```
array(['Fiji', 'Tanzania', 'W. Sahara', 'Canada',
       'United States of America', 'Kazakhstan', 'Uzbekistan',
       'Papua New Guinea', 'Indonesia', 'Argentina', 'Chile',
       'Dem. Rep. Congo', 'Somalia', 'Kenya', 'Sudan', 'Chad', 'Haiti',
       'Dominican Rep.', 'Russia', 'Bahamas', 'Falkland Is.', 'Norway',
       'Greenland', 'Fr. S. Antarctic Lands', 'Timor-Leste',
       'South Africa', 'Lesotho', 'Mexico', 'Uruguay', 'Brazil',
       'Bolivia', 'Peru', 'Colombia', 'Panama', 'Costa Rica', 'Nicaragua',
       'Honduras', 'El Salvador', 'Guatemala', 'Belize', 'Venezuela',
       'Guyana', 'Suriname', 'France', 'Ecuador', 'Puerto Rico',
       'Jamaica', 'Cuba', 'Zimbabwe', 'Botswana', 'Namibia', 'Senegal',
       'Mali', 'Mauritania', 'Benin', 'Niger', 'Nigeria', 'Cameroon',
       'Togo', 'Ghana', "Côte d'Ivoire", 'Guinea', 'Guinea-Bissau',
       'Liberia', 'Sierra Leone', 'Burkina Faso', 'Central African Rep.',
       'Congo', 'Gabon', 'Eq. Guinea', 'Zambia', 'Malawi', 'Mozambique',
       'eSwatini', 'Angola', 'Burundi', 'Israel', 'Lebanon', 'Madagascar',
       'Palestine', 'Gambia', 'Tunisia', 'Algeria', 'Jordan',
       'United Arab Emirates', 'Qatar', 'Kuwait', 'Iraq', 'Oman',
       'Vanuatu', 'Cambodia', 'Thailand', 'Laos', 'Myanmar', 'Vietnam',
       'North Korea', 'South Korea', 'Mongolia', 'India', 'Bangladesh',
       'Bhutan', 'Nepal', 'Pakistan', 'Afghanistan', 'Tajikistan',
       'Kyrgyzstan', 'Turkmenistan', 'Iran', 'Syria', 'Armenia', 'Sweden',
       'Belarus', 'Ukraine', 'Poland', 'Austria', 'Hungary', 'Moldova',
       'Romania', 'Lithuania', 'Latvia', 'Estonia', 'Germany', 'Bulgaria',
```

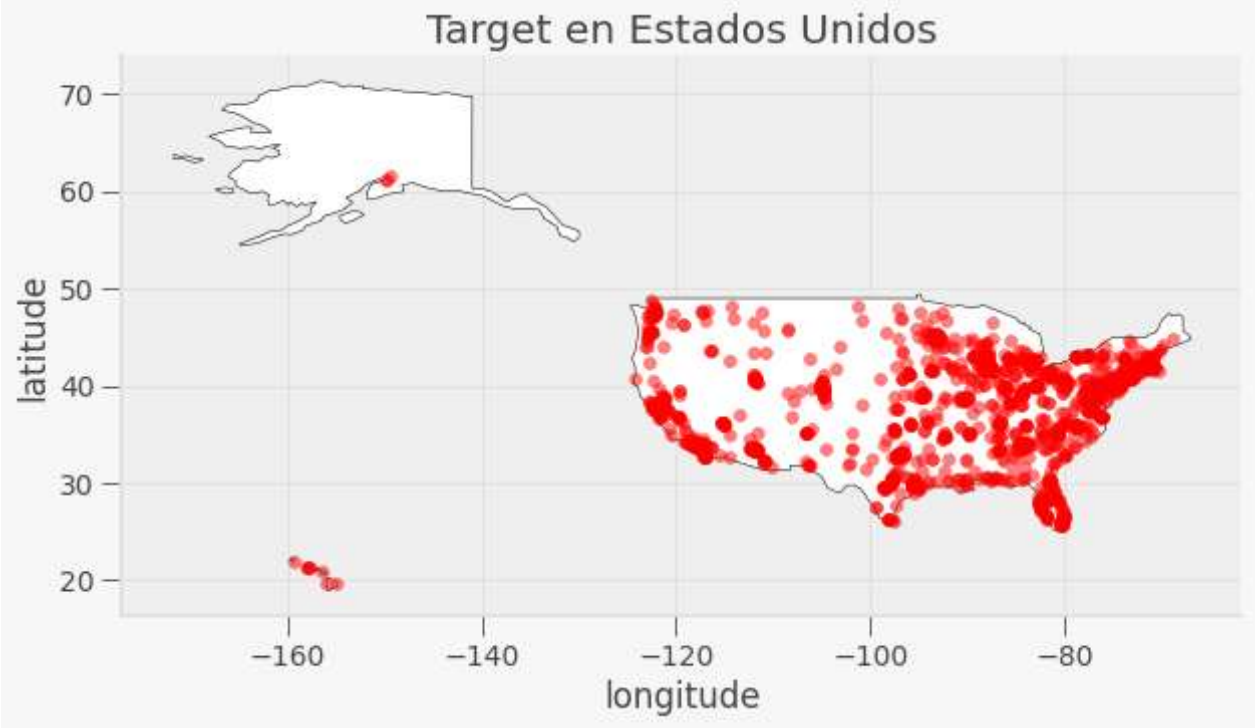
```
        'Greece', 'Turkey', 'Albania', 'Croatia', 'Switzerland',
        'Luxembourg', 'Belgium', 'Netherlands', 'Portugal', 'Spain',
        'Ireland', 'New Caledonia', 'Solomon Is.', 'New Zealand',
        'Australia', 'Sri Lanka', 'China', 'Taiwan', 'Italy', 'Denmark',
        'United Kingdom', 'Iceland', 'Azerbaijan', 'Georgia',
        'Philippines', 'Malaysia', 'Brunei', 'Slovenia', 'Finland',
        'Slovakia', 'Czechia', 'Eritrea', 'Japan', 'Paraguay', 'Yemen',
        'Saudi Arabia', 'Antarctica', 'N. Cyprus', 'Cyprus', 'Morocco',
        'Egypt', 'Libya', 'Ethiopia', 'Djibouti', 'Somaliland', 'Uganda',
        'Rwanda', 'Bosnia and Herz.', 'Macedonia', 'Serbia', 'Montenegro',
        'Kosovo', 'Trinidad and Tobago', 'S. Sudan'], dtype=object)
```

```
fig, gax = plt.subplots(figsize=(10,10))

# By only plotting rows in which the continent is 'South America' we only plot SA.
world.query("name == 'United States of America'").plot(ax=gax, edgecolor='black',color='wh

# By the way, if you haven't read the book 'longitude' by Dava Sobel, you should...
gax.set_xlabel('longitude')
gax.set_ylabel('latitude')

gax.spines['top'].set_visible(False)
gax.spines['right'].set_visible(False)
```




```
# Step 3: Plot the cities onto the map
# We mostly use the code from before --- we still want the country borders plotted --- and
# add a command to plot the cities
fig, gax = plt.subplots(figsize=(10,10))
```

```
# By only plotting rows in which the continent is 'South America' we only plot, well,
# South America.
world.query("name == 'United States of America'").plot(ax = gax, edgecolor='black', color=

# This plot the cities. It's the same syntax, but we are plotting from a different GeoData
# I want the cities as pale red dots.
gdf.plot(ax=gax, color='red', alpha = 0.5)

gax.set_xlabel('longitude')
gax.set_ylabel('latitude')
gax.set_title('Target en Estados Unidos')

gax.spines['top'].set_visible(False)
gax.spines['right'].set_visible(False)

plt.show()
```



¿qué tal ahora?, tiene mayor sentido verdad, entonces los datos lejanos no eran atípicos, de aquí la importancia de ver los datos con el tipo de gráfica correcta.

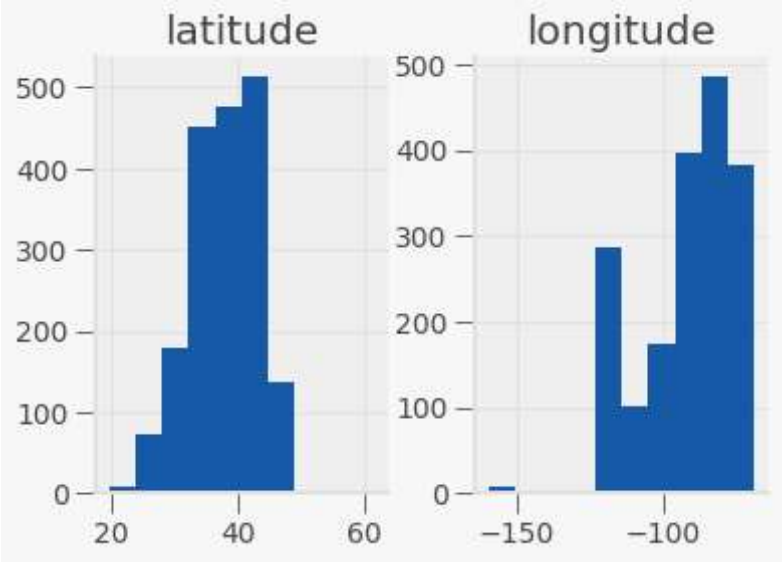
Ahora sí, implementa K means a los datos de latitud y longitud :) y encuentra donde colocar los almacenes.

Nota: si te llama la atención implementar alguna otra visualización con otra librería, lo puedes hacer, no hay restricciones.

Encuentra el numero ideal de almacenes, justifica tu respuesta:

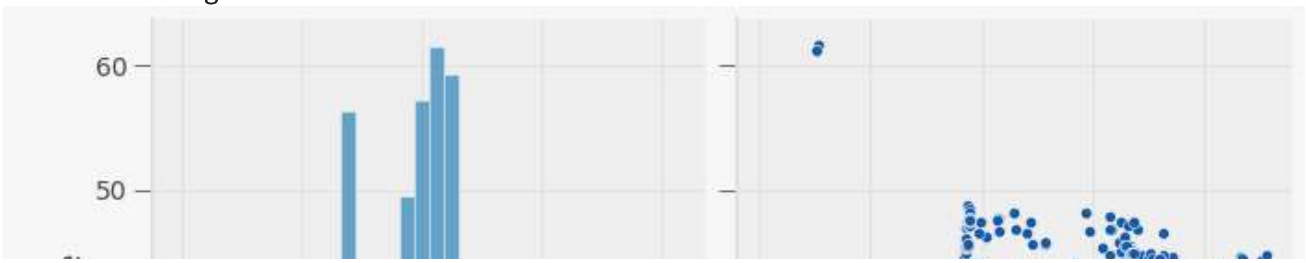
Encuentra las latitudes y longitudes de los almacenes, ¿qué ciudad es?, ¿a cuantas tiendas va surtir?, ¿sabes a que distancia estará? ¿Cómo elegiste el número de almacenes?, justifica tu respuesta técnicamente.

```
#Graficamos para tener una mejor visualización de la dispersión de los datos
latlong.hist()
plt.show()
```




```
sb.pairplot(df.dropna(),size=5,vars=['latitude','longitude'],kind='scatter')
```

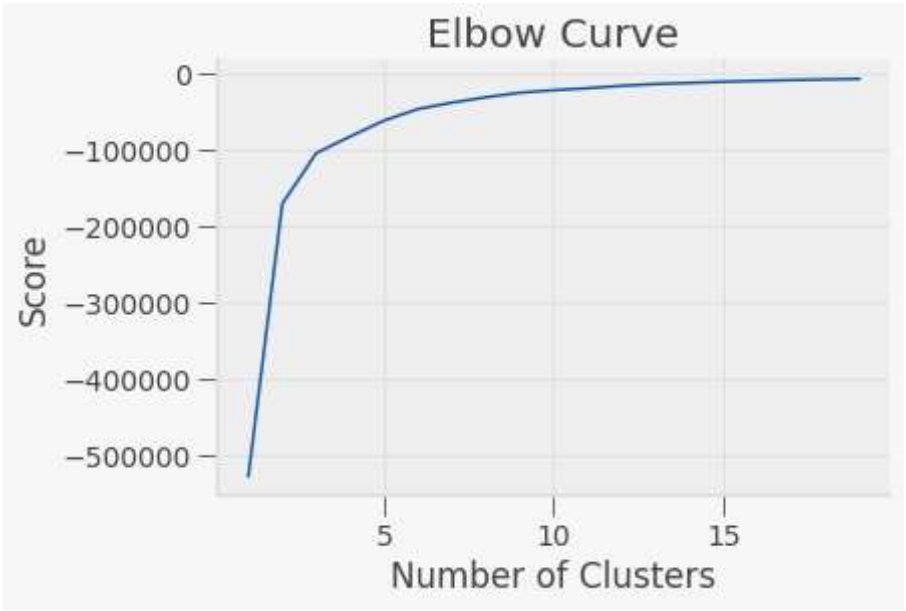
```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:2076: UserWarning: The `si
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.PairGrid at 0x7f8498dbc910>
```




```
#Precisamos la estructura de datos para alimentar el algoritmo

X = np.array(df[['latitude', 'longitude']])
y = np.array(df['Coordinates'])
y.shape
type(X)
```

```
numpy.ndarray
```

```
Nc = range(1, 20)
kmeans = [KMeans(n_clusters=i) for i in Nc]
kmeans
score = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]
score
plt.plot(Nc,score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```



Con la gráfica "ELBOW" definimos a K con un valor de 3 que es donde la linea cambia de dirección.

```
kmeans = KMeans(n_clusters=3).fit(X)
centroids = kmeans.cluster_centers_
```

```
print(centroids)
```

```
[[  37.789554    -78.56990807]
 [  37.48734203 -118.62447332]
 [  37.98006261  -93.3271723 ]]
```

Transformamos los datos en coordenadas

```
lt= ['longitude','latitude']
df_wh·=·pd.DataFrame(centroids,·columns=['longitude','latitude'])
df_wh["Warehouse"]·=·list(zip(df_wh.latitude,df_wh.longitude))
df_wh["Warehouse"]·=·df_wh["Warehouse"].apply(Point)
df_wh['City'] = ['Scottsville, VA', 'Mono County, CA', 'Hickory County, MO']
df_wh
```

	longitude	latitude	Warehouse	City
0	37.789554	-78.569908	POINT (-78.56990807484885 37.789554004474006)	Scottsville, VA
1	37.487342	-118.624473	POINT (-118.62447331844157 37.48734203064935)	Mono County, CA

```
gdf_wh = gpd.GeoDataFrame(df_wh, geometry="Warehouse")
gdf_wh.head()
```

	longitude	latitude	Warehouse	City
0	37.789554	-78.569908	POINT (-78.56991 37.78955)	Scottsville, VA
1	37.487342	-118.624473	POINT (-118.62447 37.48734)	Mono County, CA
2	37.980063	-93.327172	POINT (-93.32717 37.98006)	Hickory County, MO

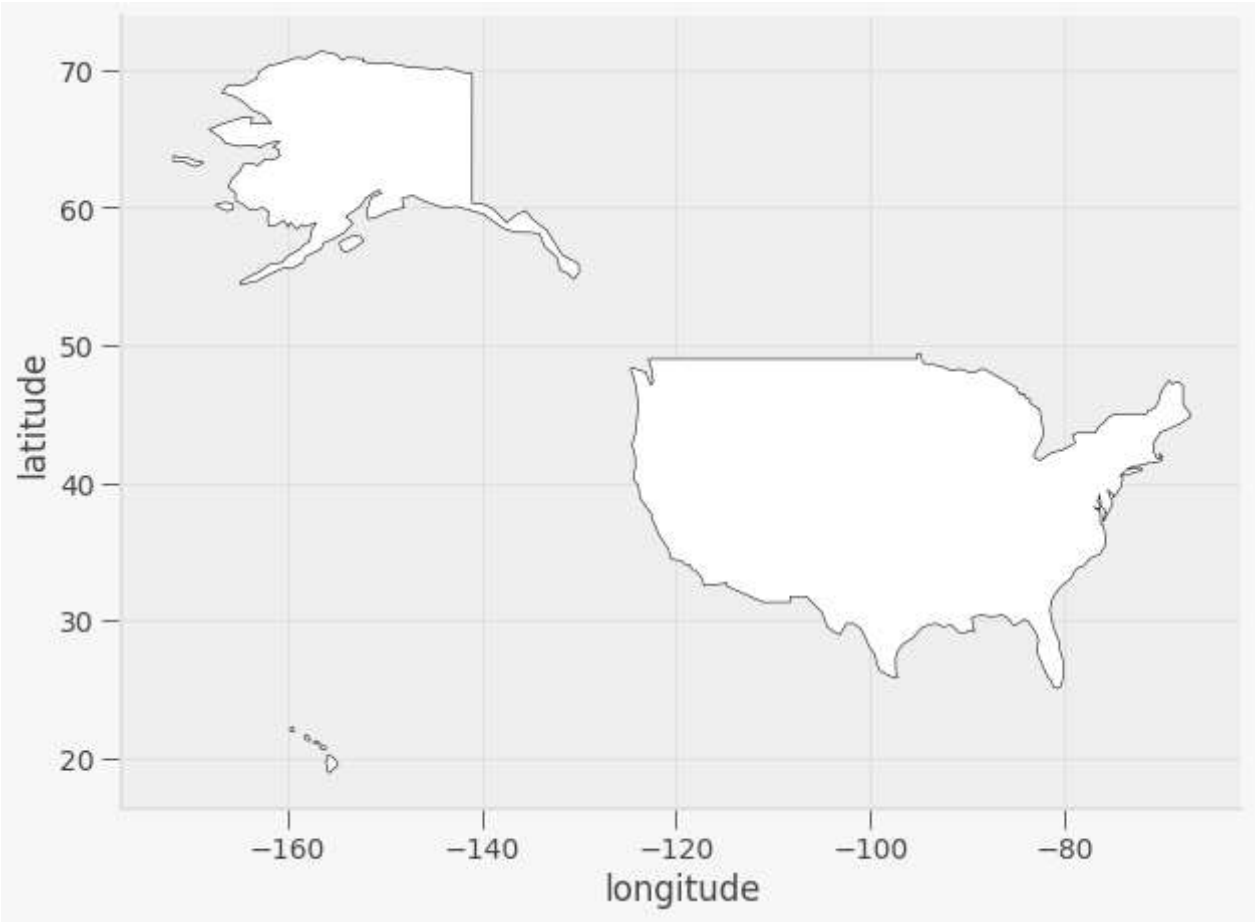
```
print('gdf is of type:', type(gdf_wh))
```

```
gdf is of type: <class 'geopandas.geodataframe.GeoDataFrame'>
```

```
print('\nThe geometry column is:', gdf.geometry.name)
```

```
The geometry column is: Coordinates
```

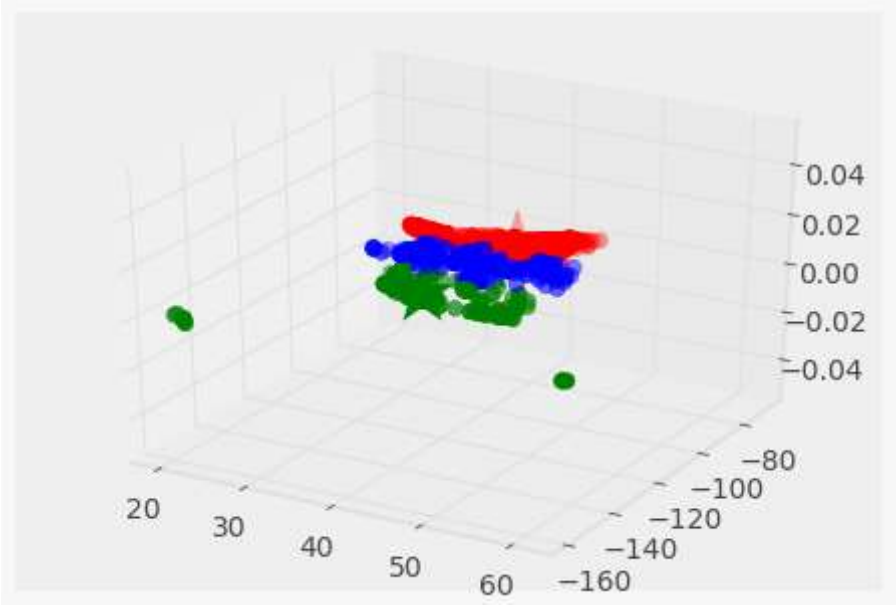
```
fig, gax = plt.subplots(figsize=(10,10))

world.query("name == 'United States of America'").plot(ax=gax, edgecolor='black',color='wh

gax.set_xlabel('longitude')
gax.set_ylabel('latitude')

gax.spines['top'].set_visible(False)
gax.spines['right'].set_visible(False)
```




```
# Step 3: Plot the cities onto the map
# We mostly use the code from before --- we still want the country borders plotted --- and
# add a command to plot the cities
fig, gax = plt.subplots(figsize=(10,10))

# By only plotting rows in which the continent is 'South America' we only plot, well,
# South America.
world.query("name == 'United States of America'").plot(ax = gax, edgecolor='black', color=

# This plot the cities. It's the same syntax, but we are plotting from a different GeoData
# I want the cities as pale red dots.
gdf_wh.plot(ax=gax, color='red', alpha = 0.5)

gax.set_xlabel('longitude')
gax.set_ylabel('latitude')
gax.set_title('Almacenes recomendados para Target en Estados Unidos')

gax.spines['top'].set_visible(False)
gax.spines['right'].set_visible(False)

# Label the cities
for x, y, label in zip(gdf_wh['Warehouse'].x, gdf_wh['Warehouse'].y, gdf_wh['City']):
  gax.annotate(label, xy=(x,y), xytext=(4,4), textcoords='offset points')

plt.show()
```


```
# Predicting the clusters
labels = kmeans.predict(X)
# Getting the cluster centers
C = kmeans.cluster_centers_
colores=['red','green','blue']
asignar=[]
for row in labels:
    asignar.append(colores[row])

fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X[:, 0], X[:, 1], c=asignar,s=60)
ax.scatter(C[:, 0], C[:, 1], marker='*', c=colores, s=1000)
```

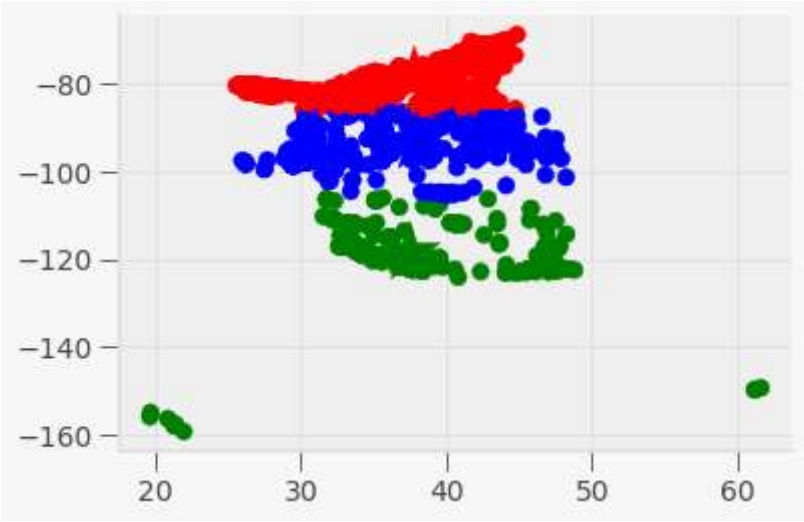
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7f8498e08590>
```



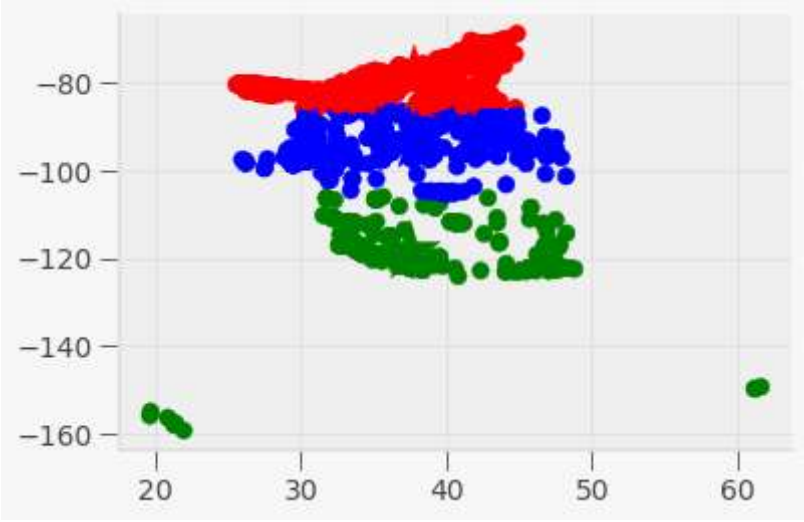
Observamos con el algoritmo K-Means que "K=3" ha agrupado a diferentes tiendas de Target por su localización, teniendo en cuenta las 2 dimensiones que utilizamos.

Realizaremos algunas gráficas en 2 dimensiones con las proyecciones a partir de nuestra gráfica 3D para que nos ayude a visualizar los grupos y su clasificación.

```
# Getting the values and plotting it
f1 = df['latitude'].values
f2 = df['longitude'].values

plt.scatter(f1, f2, c=asignar, s=70)
plt.scatter(C[:, 0], C[:, 1], marker='*', c=colores, s=1000)
plt.show()
```

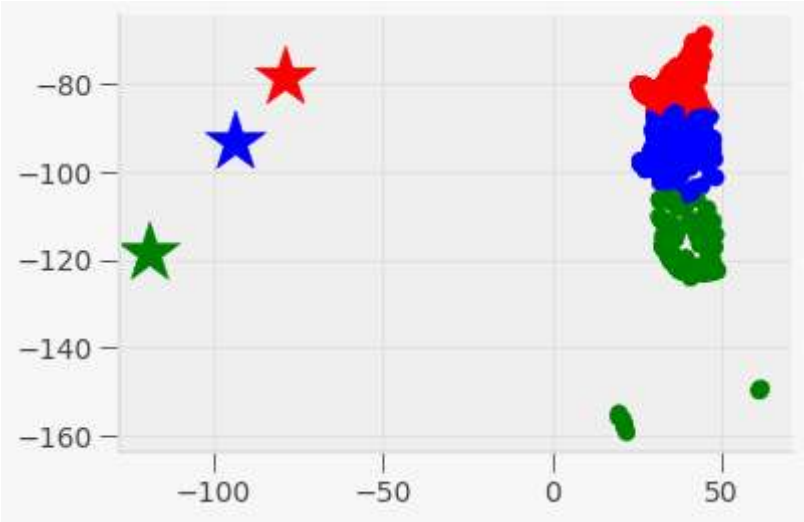



```
# Getting the values and plotting it
f1 = df['latitude'].values
f2 = df['longitude'].values

plt.scatter(f1, f2, c=asignar, s=70)
plt.scatter(C[:, 0], C[:, 1], marker='*', c=colores, s=1000)
plt.show()
```


```
f1 = df['latitude'].values
f2 = df['longitude'].values
```

```
plt.scatter(f1, f2, c=asignar, s=70)
plt.scatter(C[:, 1], C[:, 1], marker='*', c=colores, s=1000)
plt.show()
```



Podemos ver cuantas tiendas hay en cada cluster

```
copy =  pd.DataFrame()
copy['latitude']=df['latitude'].values
copy['longitude']=df['longitude'].values
copy['label'] = labels;
cantidadGrupo =  pd.DataFrame()
cantidadGrupo['color']=colores
cantidadGrupo['cantidad']=copy.groupby('label').size()
cantidadGrupo
```

	color	cantidad
0	red	826
1	green	385
2	blue	628

Podemos observar la tiendas mas cerca de su centroide

```
closest, _ = pairwise_distances_argmin_min(kmeans.cluster_centers_, X)
closest
```

```
array([1689,  212,  998])
```

```
stores=df['Coordinates'].values
for row in closest:
    print(stores[row])
```

```
POINT (-78.44076369999999 38.1289566)
POINT (-119.6989832 36.8065383)
```

```
POINT (-92.6029586 38.1610749)
```

Vemos las coordenadas de las tiendas más cercanas a los centroides. De igual forma, podemos agrupar y etiquetar nuevas localidades con sus características y clasificarlas. Vemos el ejemplo con la tienda de Alabaster y nos devuelve que pertenece al grupo 3.

```
X_new = np.array([[33.224225, -86.804174]])

new_labels = kmeans.predict(X_new)
print(new_labels)
```

```
[2]
```

¿Qué librerías nos pueden ayudar a graficar este tipo de datos? Numpy, Geemap, PyViz/HoloViz, Geopandas y Matplotlib

**¿Consideras importante que se grafique en un mapa? ¿por qué?* Si, ya que facilitan la visualización de mapas, mostrar demasiada información en capas distintas, ademas de ser interactivas, poder escoger.

Conclusiones

Estas nuevas librerias geoespaciales, son demasiado utiles para precisar mucha información que pueda ser necesario en temas de logistica y analisis, esta practica da una muestra de lo util que puede ser este tipo de librerias, impulsando mejor su utilidad con el uso de librerias de ciertos modelos previamente visto.