#**Maestría en Inteligencia Artificial Aplicada**

##**Curso: Ciencia y analítica de datos**

###**Tecnológico de Monterrey**

###**PhD. María de la Paz Rico Fdz**

Actividad Semanal -- 7

###**Regresiones y K means.**

###**Fecha de entrega: 09/11/2022.**

Alumno: Maximiliano Morones Gómez

Matrícula: A01793815

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-wheels/public/simple/
Collecting qeds
  Downloading qeds-0.7.0.tar.gz (24 kB)
Collecting fiona
  Downloading Fiona-1.8.22-cp37-cp37m-manylinux2014_x86_64.whl (16.7
MB)
ent already satisfied: xgboost in /usr/local/lib/python3.7/dist-
packages (0.90)
Requirement already satisfied: gensim in
/usr/local/lib/python3.7/dist-packages (3.6.0)
Requirement already satisfied: folium in
/usr/local/lib/python3.7/dist-packages (0.12.1.post1)
Collecting pyLDAvis
  Downloading pyLDAvis-3.3.1.tar.gz (1.7 MB)
ents to build wheel ... etadata ... ent already satisfied: descartes
in /usr/local/lib/python3.7/dist-packages (1.1.0)
Requirement already satisfied: pandas in
/usr/local/lib/python3.7/dist-packages (from qeds) (1.3.5)
Requirement already satisfied: requests in
/usr/local/lib/python3.7/dist-packages (from qeds) (2.23.0)
Collecting quandl
  Downloading Quandl-3.7.0-py2.py3-none-any.whl (26 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-
packages (from qeds) (1.7.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-
packages (from qeds) (1.21.6)
Collecting quantecon
  Downloading quantecon-0.5.3-py3-none-any.whl (179 kB)
ent already satisfied: matplotlib in /usr/local/lib/python3.7/dist-
packages (from qeds) (3.2.2)
Requirement already satisfied: pyarrow in
/usr/local/lib/python3.7/dist-packages (from qeds) (6.0.1)
Requirement already satisfied: openpyxl in
/usr/local/lib/python3.7/dist-packages (from qeds) (3.0.10)
Requirement already satisfied: plotly in
/usr/local/lib/python3.7/dist-packages (from qeds) (5.5.0)
Requirement already satisfied: pandas_datareader in
/usr/local/lib/python3.7/dist-packages (from qeds) (0.9.0)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.7/dist-packages (from qeds) (1.0.2)
Requirement already satisfied: seaborn in
/usr/local/lib/python3.7/dist-packages (from qeds) (0.11.2)
Requirement already satisfied: statsmodels in
/usr/local/lib/python3.7/dist-packages (from qeds) (0.12.2)
Requirement already satisfied: six>=1.7 in
/usr/local/lib/python3.7/dist-packages (from fiona) (1.15.0)
Requirement already satisfied: attrs>=17 in
/usr/local/lib/python3.7/dist-packages (from fiona) (22.1.0)
Requirement already satisfied: setuptools in
```

```
/usr/local/lib/python3.7/dist-packages (from fiona) (57.4.0)
Collecting click-plugins>=1.0
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Collecting cligj>=0.5
  Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Requirement already satisfied: click>=4.0 in
/usr/local/lib/python3.7/dist-packages (from fiona) (7.1.2)
Requirement already satisfied: certifi in
/usr/local/lib/python3.7/dist-packages (from fiona) (2022.9.24)
Collecting munch
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Collecting pyproj>=2.2.0
  Downloading pyproj-3.2.1-cp37-cp37m-manylinux2010_x86_64.whl (6.3
MB)
ent already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-
packages (from geopandas) (1.8.5.post1)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.7/dist-packages (from pandas->qeds) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in
/usr/local/lib/python3.7/dist-packages (from pandas->qeds) (2022.6)
Requirement already satisfied: smart-open>=1.2.1 in
/usr/local/lib/python3.7/dist-packages (from gensim) (5.2.1)
Requirement already satisfied: branca>=0.3.0 in
/usr/local/lib/python3.7/dist-packages (from folium) (0.5.0)
Requirement already satisfied: jinja2>=2.9 in
/usr/local/lib/python3.7/dist-packages (from folium) (2.11.3)
Requirement already satisfied: MarkupSafe>=0.23 in
/usr/local/lib/python3.7/dist-packages (from jinja2>=2.9->folium)
(2.0.1)
Requirement already satisfied: future in
/usr/local/lib/python3.7/dist-packages (from pyLDAvis) (0.16.0)
Requirement already satisfied: joblib in
/usr/local/lib/python3.7/dist-packages (from pyLDAvis) (1.2.0)
Requirement already satisfied: numexpr in
/usr/local/lib/python3.7/dist-packages (from pyLDAvis) (2.8.4)
Collecting sklearn
  Downloading sklearn-0.0.post1.tar.gz (3.6 kB)
Collecting funcy
  Downloading funcy-1.17-py2.py3-none-any.whl (33 kB)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->qeds) (1.4.4)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->qeds)
(0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!
=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from
matplotlib->qeds) (3.0.9)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1-
>matplotlib->qeds) (4.1.1)
```

```
Requirement already satisfied: et-xmlfile in
/usr/local/lib/python3.7/dist-packages (from openpyxl->qeds) (1.1.0)
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-
packages (from pandas_datareader->qeds) (4.9.1)
Requirement already satisfied: idna<3,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests->qeds) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests->qeds)
(1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests->qeds) (3.0.4)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.7/dist-packages (from plotly->qeds) (8.1.0)
Collecting inflection>=0.3.1
  Downloading inflection-0.5.1-py2.py3-none-any.whl (9.5 kB)
Requirement already satisfied: more-itertools in
/usr/local/lib/python3.7/dist-packages (from quandl->qeds) (9.0.0)
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-
packages (from quantecon->qeds) (0.56.4)
Requirement already satisfied: sympy in /usr/local/lib/python3.7/dist-
packages (from quantecon->qeds) (1.7.1)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in
/usr/local/lib/python3.7/dist-packages (from numba->quantecon->qeds)
(0.39.1)
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from numba->quantecon->qeds)
(4.13.0)
Requirement already satisfied: zipp>=0.5 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata-
>numba->quantecon->qeds) (3.10.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn->qeds)
(3.1.0)
Requirement already satisfied: patsy>=0.5 in
/usr/local/lib/python3.7/dist-packages (from statsmodels->qeds)
(0.5.3)
Requirement already satisfied: mpmath>=0.19 in
/usr/local/lib/python3.7/dist-packages (from sympy->quantecon->qeds)
(1.2.1)
Building wheels for collected packages: qeds, pyLDAvis, sklearn
  Building wheel for qeds (setup.py) ... e=qeds-0.7.0-py3-none-any.whl
size=27812
sha256=4d522f166b60d2936c315ef2705056066c6e182fb3880adb8f38ab4485114e7
5
  Stored in directory:
/root/.cache/pip/wheels/fc/8c/52/0cc036b9730b75850b9845770780f8d05ed08
ff38a67cbaa29
  Building wheel for pyLDAvis (PEP 517) ... e=pyLDAvis-3.3.1-py2.py3-
none-any.whl size=136897
sha256=37f0906b899b63137d65f7e962ee1b81625bfeee43e823e91d9cf70a7984952
```

```
6
  Stored in directory:
/root/.cache/pip/wheels/c9/21/f6/17bcf2667e8a68532ba2fbf6d5c72fdf4c7f7
d9abfa4852d2f
  Building wheel for sklearn (setup.py) ... e=sklearn-0.0.post1-py3-
none-any.whl size=2344
sha256=5a754ab7e0d76ab52baf226e3649645dd9166e93b35a920a6f5716eaeb8720e
4
  Stored in directory:
/root/.cache/pip/wheels/42/56/cc/4a8bf86613aafd5b7f1b310477667c1fca5c5
1c3ae4124a003
Successfully built qeds pyLDAvis sklearn
Installing collected packages: munch, inflection, cligj, click-
plugins, sklearn, quantecon, quandl, pyproj, funcy, fiona, qeds,
pyLDAvis, geopandas
Successfully installed click-plugins-1.1.1 cligj-0.7.2 fiona-1.8.22
funcy-1.17 geopandas-0.10.2 inflection-0.5.1 munch-2.5.0 pyLDAvis-
3.3.1 pyproj-3.2.1 qeds-0.7.0 quandl-3.7.0 quantecon-0.5.3 sklearn-
0.0.post1
```

```
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
```

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  \
0  Alabaster  33.224225 -86.804174
1   Bessemer  33.334550 -86.989778
2     Daphne  30.602875 -87.895932
3    Decatur  34.560148 -86.971559
4     Dothan  31.266061 -85.446422

                                          address         phone  \
0        250 S Colonial Dr, Alabaster, AL 35007-4657  205-564-2608
1       4889 Promenade Pkwy, Bessemer, AL 35022-7305  205-565-3760
2          1698 US Highway 98, Daphne, AL 36526-4252  251-621-3540
3  1235 Point Mallard Pkwy SE, Decatur, AL 35601-...  256-898-3036
```

```
4         4601 Montgomery Hwy, Dothan, AL 36303-1522  334-340-1112

                                        website
0  https://www.target.com/sl/alabaster/2276
1   https://www.target.com/sl/bessemer/2375
2     https://www.target.com/sl/daphne/1274
3    https://www.target.com/sl/decatur/2084
4     https://www.target.com/sl/dothan/1468
```

```
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.

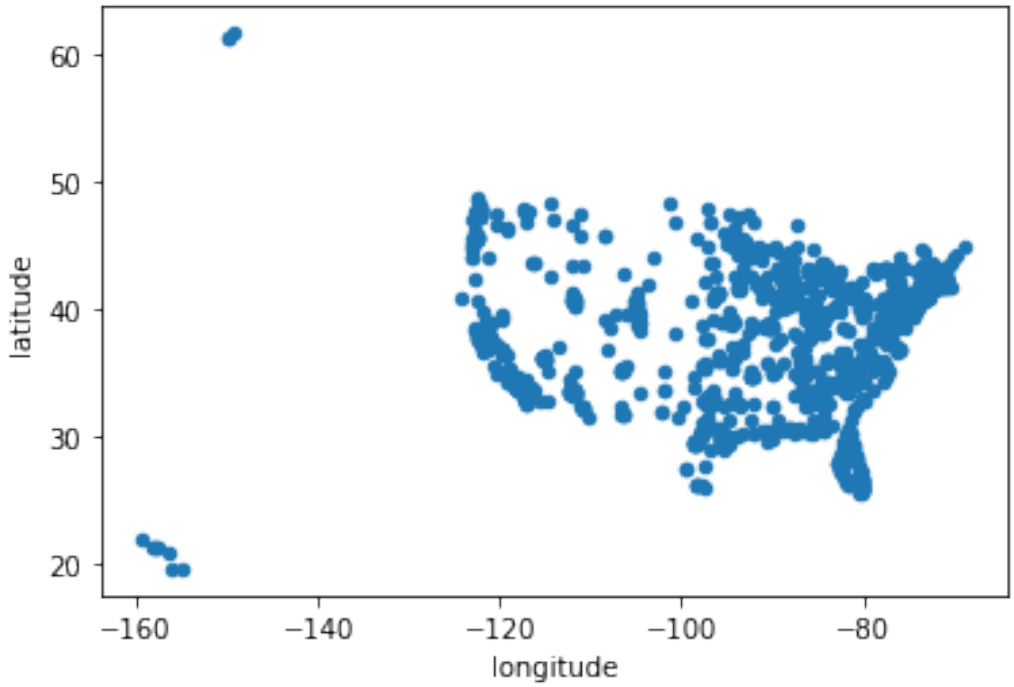
```
latlong=df[["latitude","longitude"]]
```

¡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 0x7f2e03844d10>
```




```
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  \
0  Alabaster  33.224225 -86.804174
1   Bessemer  33.334550 -86.989778
2     Daphne  30.602875 -87.895932
3    Decatur  34.560148 -86.971559
4     Dothan  31.266061 -85.446422

                                              address         phone  \
0        250 S Colonial Dr, Alabaster, AL 35007-4657  205-564-2608
1       4889 Promenade Pkwy, Bessemer, AL 35022-7305  205-565-3760
2          1698 US Highway 98, Daphne, AL 36526-4252  251-621-3540
3  1235 Point Mallard Pkwy SE, Decatur, AL 35601-...  256-898-3036
4         4601 Montgomery Hwy, Dothan, AL 36303-1522  334-340-1112

                                    website  \
0  https://www.target.com/sl/alabaster/2276
1   https://www.target.com/sl/bessemer/2375
2     https://www.target.com/sl/daphne/1274
3    https://www.target.com/sl/decatur/2084
4     https://www.target.com/sl/dothan/1468

                            Coordinates
0  POINT (-86.80417369999999 33.2242254)
1  POINT (-86.98977789999999 33.3345501)
2  POINT (-87.89593169999999 30.6028747)
3         POINT (-86.9715595 34.5601477)
4         POINT (-85.4464222 31.2660613)
```

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

```
        name   latitude  longitude  \
0  Alabaster  33.224225 -86.804174
1   Bessemer  33.334550 -86.989778
2     Daphne  30.602875 -87.895932
3    Decatur  34.560148 -86.971559
4     Dothan  31.266061 -85.446422

                                              address         phone  \
0        250 S Colonial Dr, Alabaster, AL 35007-4657  205-564-2608
1       4889 Promenade Pkwy, Bessemer, AL 35022-7305  205-565-3760
2          1698 US Highway 98, Daphne, AL 36526-4252  251-621-3540
3  1235 Point Mallard Pkwy SE, Decatur, AL 35601-...  256-898-3036
4         4601 Montgomery Hwy, Dothan, AL 36303-1522  334-340-1112

                                    website
Coordinates
0  https://www.target.com/sl/alabaster/2276  POINT (-86.80417
33.22423)
1   https://www.target.com/sl/bessemer/2375  POINT (-86.98978
```

```
33.33455)
2     https://www.target.com/sl/daphne/1274  POINT (-87.89593
30.60287)
3    https://www.target.com/sl/decatur/2084  POINT (-86.97156
34.56015)
4     https://www.target.com/sl/dothan/1468  POINT (-85.44642
31.26606)
```

```
#mapa

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

world.head()
```

```
          pop_est      continent                      name  gdp_md_est
\
iso_a3

FJI        920938        Oceania                      Fiji      8374.0

TZA      53950935         Africa                  Tanzania    150600.0

ESH        603253         Africa                 W. Sahara       906.5

CAN      35623680  North America                    Canada   1674000.0

USA     326625791  North America  United States of America  18560000.0


                                                  geometry
iso_a3
FJI     MULTIPOLYGON (((180.00000 -16.06713, 180.00000...
TZA     POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...
ESH     POLYGON ((-8.66559 27.65643, -8.66512 27.58948...
CAN     MULTIPOLYGON (((-122.84000 49.00000, -122.9742...
USA     MULTIPOLYGON (((-122.84000 49.00000, -120.0000...
```

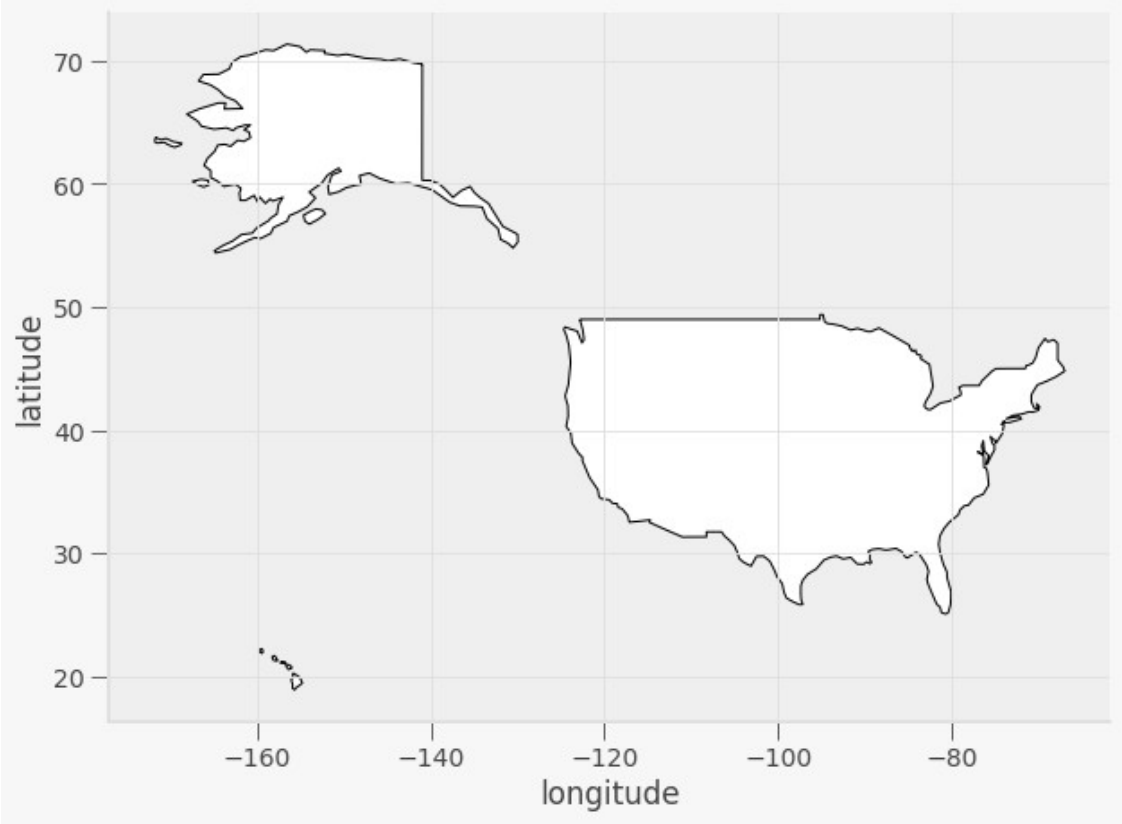
```
#graficar el mapa
world.name.unique()
```

```
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)
```

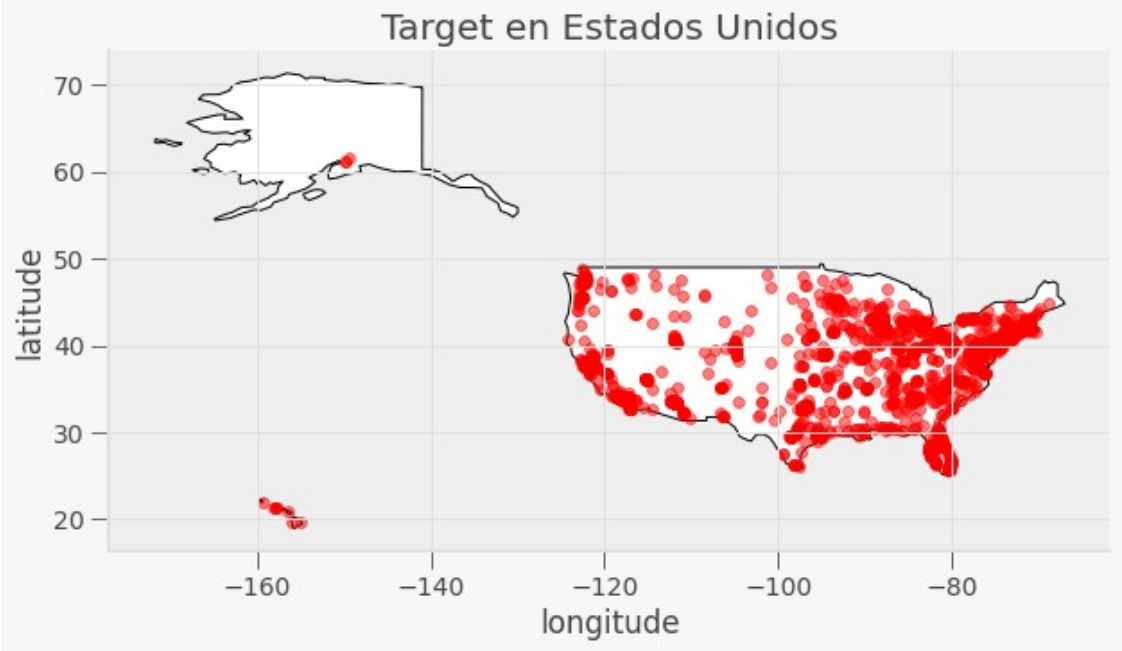
```
fig, gax = plt.subplots(figsize=(10,10))

# 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='white')

# 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 we
# 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='white')

# This plot the cities. It's the same syntax, but we are plotting from
a different GeoDataFrame.
# 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()
```

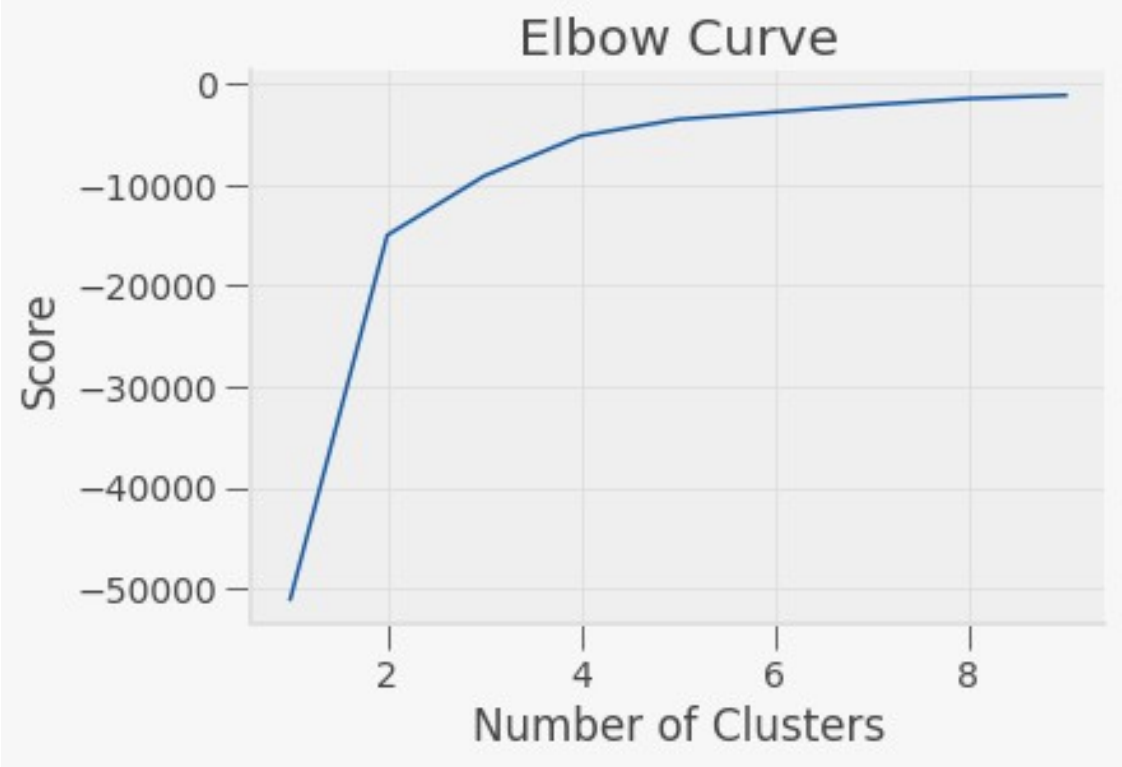


¿qué tal ahora?, tiene mayor sentido verdad, entonces los datos lejanos no eran atípicos, de aquí la importancia de ver los datos con el tipo de gráfica correcta.

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.

```
#tu codigo aquí
from sklearn.cluster import KMeans
K_clusters = range(1,10)
kmeans = [KMeans(n_clusters=i) for i in K_clusters]
Y_axis = latlong[['latitude']]
X_axis = latlong[['longitude']]
score = [kmeans[i].fit(Y_axis).score(Y_axis) for i in
range(len(kmeans))]
plt.plot(K_clusters, score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```



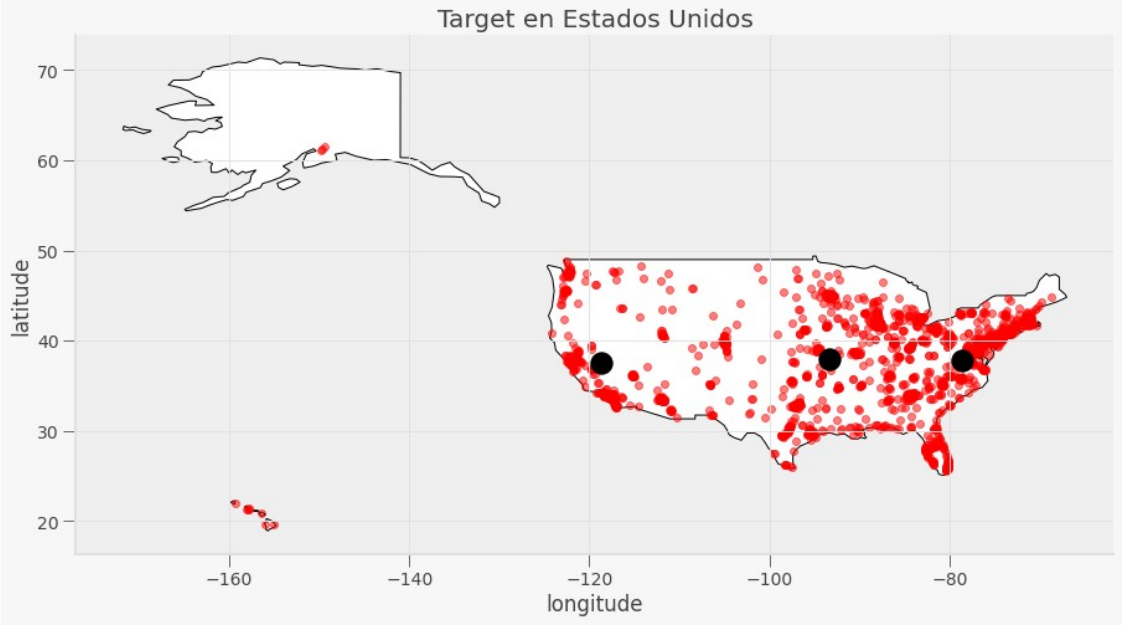

```
kmeans = KMeans(n_clusters = 3, init ='k-means++')
kmeans.fit(latlong[latlong.columns[0:2]])
labels = kmeans.labels_
labels
```

```
array([2, 2, 2, ..., 1, 2, 1], dtype=int32)
```

```
X = df[["longitude","latitude"]]
kmeans = KMeans(n_clusters=3).fit(X)
centroids = kmeans.cluster_centers_
labels = kmeans.predict(X)
C = kmeans.cluster_centers_

C_DF = pd.DataFrame(C)
C_DF["Coordinates"] = list(zip(C_DF[0], C_DF[1]))
C_DF["Coordinates"] = C_DF["Coordinates"].apply(Point)


gdf_C = gpd.GeoDataFrame(C_DF, geometry="Coordinates")
gdf_C
```

```
            0          1                    Coordinates
0  -78.569908  37.789554   POINT (-78.56991 37.78955)
1 -118.624473  37.487342  POINT (-118.62447 37.48734)
2  -93.327172  37.980063   POINT (-93.32717 37.98006)
```

```
fig, gax = plt.subplots(figsize=(15,10))
world.query("name == 'United States of America'").plot(ax = gax,
edgecolor='black', color='white')
gdf.plot(ax=gax, color='red', alpha = 0.5)
gdf_C.plot(ax=gax, color='black', alpha = 1, markersize = 300)
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()
```



¿A cuantas tiendas va surtir?

```
latlong['kmeans'] = kmeans.labels_
latlong.loc[:, 'kmeans'].value_counts()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.

0    826
2    628
1    385
Name: kmeans, dtype: int64
```

¿Qué ciudad es?

```
from pandas.core.internals.concat import concat_arrays
Location1 = str(gdf_C[1][0]) + ", " + str(gdf_C[0][0])
print(Location1)
Location2 = str(gdf_C[1][1]) + ", " + str(gdf_C[0][1])
print(Location2)
Location3 = str(gdf_C[1][2]) + ", " + str(gdf_C[0][2])
print(Location3)
```

```
37.789554004474006, -78.56990807484885
37.48734203064935, -118.62447331844157
37.98006260590112, -93.32717230430622
```

¿Sabes a que distancia estará?

```
from geopy.geocoders.yandex import Location
from geopy.geocoders import Nominatim
from geopy.distance import geodesic
geolocator = Nominatim(user_agent="my-application")
Locations = [Location1, Location2, Location3]
for i in Locations:
  location = geolocator.reverse(i)
  print('almacen en ---', location.address)
```

```
almacen en --- Langhorne Road, Totier Hills, Albemarle County,
Virginia, 22946, United States
almacen en --- Paradise Estates, Mono County, California, United
States
almacen en --- Hickory County, Missouri, United States
```

¿Sabes a que distancia estará?

```
distancia1 = str(geodesic(Location1, Location2).miles)
print("\nDistancia entre el primer y segundo almacén : ", distancia1,
" ft2 \n")
distancia2 = str(geodesic(Location2, Location3).miles)
print("Distancia entre el segundo y tercer almacén : ", distancia2, "
ml \n")
```

```
Distancia entre el primer y segundo almacén :  2179.654449831999  ft2

Distancia entre el segundo y tercer almacén :  1381.7597109962394  ml
```

¿Qué librerías nos pueden ayudar a graficar este tipo de datos?

Geopandas es una de las librerías que pueden ayudarnos a trabajar con este tipo de datos, ya que nos facilita el trabajo con datos geoespaciales.

GeoPy es otra de las librerías que pueden apoyarnos para este tipo de trabajos, ya que en el caso de esta librería, se nos facilita el poder localizar las coordenadas de las direcciones deseadas.

¿Consideras importante que se grafique en un mapa?, ¿por qué?

Si, ya que mediante la ayuda del mapa es más fácil poder ubicar puntos relevantes, los cuales pueden contener información importante, así como también estos pueden apoyarnos en la ubicación de los lugares de interés.