CIENCIA Y ANALITICA DE DATOS

Actividad Semanal -- 7 Regresiones y K means

Notebook 2. K means.

Profesor Titular: Maria de la Paz Rico Fernandez

Profesor Tutor: Juan Miguel Meza Méndez

Alumno: Samuel Elías Flores González

Matrícula: A01793668

Fecha: 9/Noviembre/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
```

```
    Requirement already satisfied: pyLDAvis in /usr/local/lib/python3.7/dist-packages (3.
    Requirement already satisfied: descartes in /usr/local/lib/python3.7/dist-packages (1
    Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: quandl in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: openpyxl in /usr/local/lib/python3.7/dist-packages (fr
    Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (fr
    Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (
    Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: pyarrow in /usr/local/lib/python3.7/dist-packages (fro
    Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: pandas-datareader in /usr/local/lib/python3.7/dist-pac
    Requirement already satisfied: quantecon in /usr/local/lib/python3.7/dist-packages (f
    Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (fro
    Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.7/dist-packages (
    Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-packages (
    Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (
    Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages (fr
    Requirement already satisfied: munch in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.7/dist-pa
    Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (fro
    Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packages (f
    Requirement already satisfied: pyproj>=2.2.0 in /usr/local/lib/python3.7/dist-package
    Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
    Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist-pac
    Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.7/dist-package
    Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.7/dist-packages

    Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: funcy in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: sklearn in /usr/local/lib/python3.7/dist-packages (fro
    Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: numexpr in /usr/local/lib/python3.7/dist-packages (fro
    Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
```

```
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages (from p
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: inflection>=0.3.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: sympy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.7
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.7/dist-packages (
```

```
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	address	phone	websi
0	Alabaster	33.224225	-86.804174	250 S Colonial Dr, Alabaster, AL 35007-4657	205-564-2608	https://www.target.com/sl/alabaster/22
1	Bessemer	33.334550	-86.989778	4889 Promenade Pkwy, Bessemer, AL 35022-7305	205-565-3760	https://www.target.com/sl/bessemer/23

```
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.

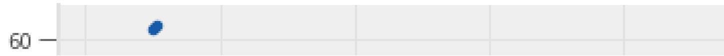
```
latlong=df[["latitude","longitude"]]
```

¡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 0x7fb05a91f9d0>
```




```
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	website

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

	name	latitude	longitude	address	phone	website
0	Alabaster	33.224225	-86.804174	250 S Colonial Dr, Alabaster, AL 35007-4657	205-564-2608	https://www.target.com/sl/alabaster/2276
1	Bessemer	33.334550	-86.989778	4889 Promenade Pkwy,	205-565-	https://www.target.com/sl/bessemer/2375

```
#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	geometr
iso_a3					
FJI	920938	Oceania	Fiji	8374.0	MULTIPOLYGON (((180.0000 -16.06713, 180.00000
TZA	53950935	Africa	Tanzania	150600.0	POLYGON ((33.90371 -0.9500 34.07262 -1.05982
ESH	603253	Africa	W. Sahara	906.5	POLYGON ((-8.66559 27.6564 -8.66512 27.58948
		North			MULTIPOLYGON (((-122.840(

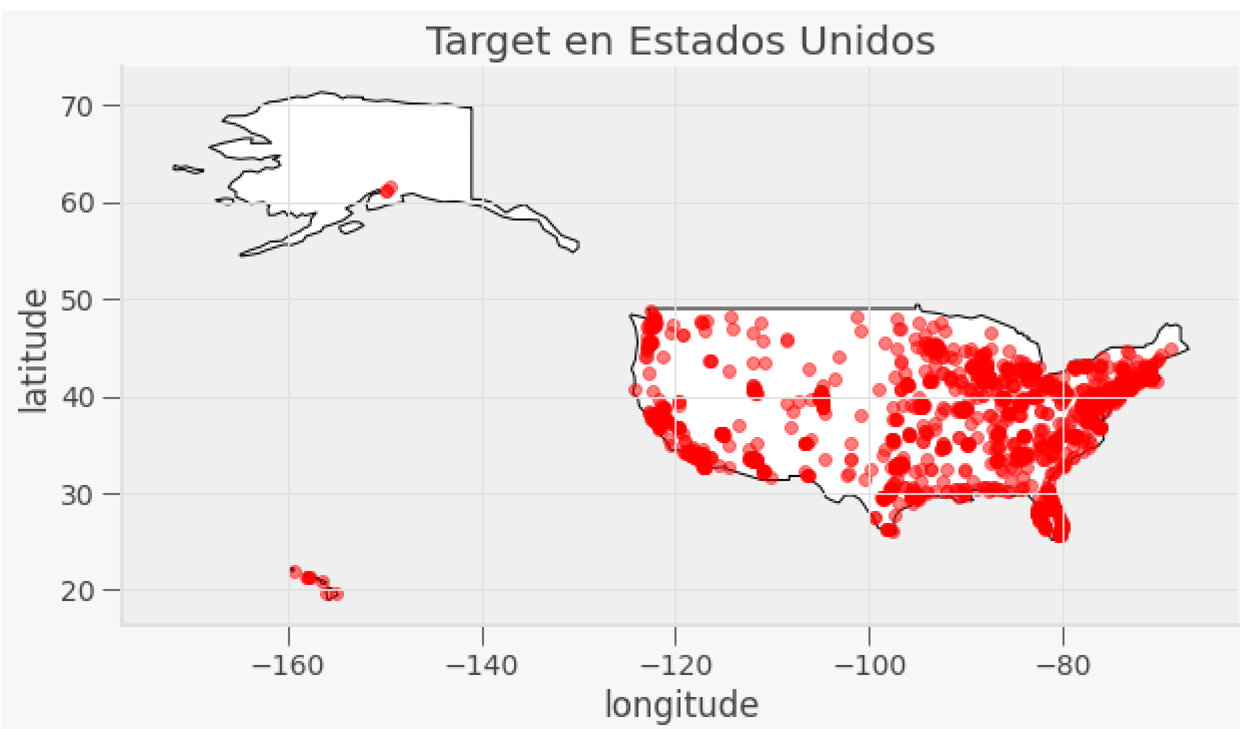
```
#graficar el mapa
world.name.unique()
```

```
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='wh

# This plot the cities. It's the same syntax, but we are plotting from a different GeoDataFra
# 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  #Importamos librerias
from sklearn.datasets import make_blobs
```

```
list_geo=list(zip(df.longitude, df.latitude)) #Definimos lista con coordenadas
list_geo
```

```
      (-93.5015727, 44.77603080000001),
      (-93.1439917, 45.0563308),
      (-93.3933865, 44.93602780000001),
      (-92.8379145, 45.03732),
      (-93.060212, 45.0526606),
      (-92.5492201, 47.5112575),
      (-93.7712845, 44.8405613),
      (-93.0779699, 44.8930259),
      (-95.0408857, 45.09797),
      (-91.6200757, 44.03211839999999),
      (-93.2140036, 44.7301054),
      (-93.1750944, 44.7237303),
      (-93.2691109, 45.1269214),
      (-93.3498563, 45.1944068),
      (-93.2371514, 44.9494431),
      (-93.3466055, 44.93725939999999),
      (-93.2293623, 45.005353),
      (-93.2745831, 44.9748302),
      (-93.2351128, 44.98185609999999),
      (-93.2961728, 44.9487059),
      (-93.4478659, 44.9698145),
      (-93.5055572, 44.9167572),
      (-92.5033946, 44.0624524),
      (-92.4656689, 43.9544007),
      (-94.2101411, 45.5571087),
      (-94.14564, 45.5648036),
      (-93.0291443, 44.9497439),
      (-93.1558509, 44.9537407),
      (-93.1882674, 44.9175067),
      (-92.9591225, 44.9270838),
      (-92.9096377, 44.9398704),
      (-88.89952009999999, 30.4557758),
      (-90.0613934, 32.3435065),
      (-89.384664, 31.3239115),
      (-90.00510020000002, 34.9663371)
```

```
(-90.0051002000000002, 34.9003371),
(-90.148254, 32.3988113),
(-89.8981586, 34.9654335),
(-90.3967635, 38.4121723),
(-90.547351, 38.5949945),
(-94.5157175, 38.81615499999999),
(-94.2482362, 39.0234008),
(-93.2261566, 36.6742513),
(-90.3428366, 38.6277927),
(-90.4256271, 38.7538807),
(-89.5796759, 37.2996366),
(-92.3772164, 38.9638174),
(-90.7688797, 38.7688176),
(-90.4475664, 38.5024026),
(-90.3097883, 38.803396),
(-94.369521, 39.0511202),
(-92.2136752, 38.5791869),
(-94.4736995, 37.0847655),
(-90.4041247, 38.5649019),
(-94.40952860000002, 38.9316746),
(-90.696137, 38.7771213),
(-92.6029586, 38.1610749),
(-90.5622786, 38.7873229),
...]
```

```
df.head()
```

	name	latitude	longitude	address	phone	website
0	Alabaster	33.224225	-86.804174	250 S Colonial Dr, Alabaster, AL 35007-4657	205-564-2608	https://www.target.com/sl/alabaster/2276
1	Bessemer	33.334550	-86.989778	4889 Promenade Pkwy, Bessemer	205-565-	https://www.target.com/sl/bessemer/2375

```
centers_b = list_geo
```

```
X, y = make_blobs(n_samples=1839, centers=centers_b, cluster_std=0.20,
                  random_state=7)
```

```
print(X)
```

```
[[ -81.4511825    32.10571602]
 [ -86.52523118   36.01168885]
 [-123.13073166   45.71017966]
 ...
 [-122.75515253   45.67169298]
 [ -71.68181402   41.71902706]
 [ -73.6808068    40.98018108]]
```

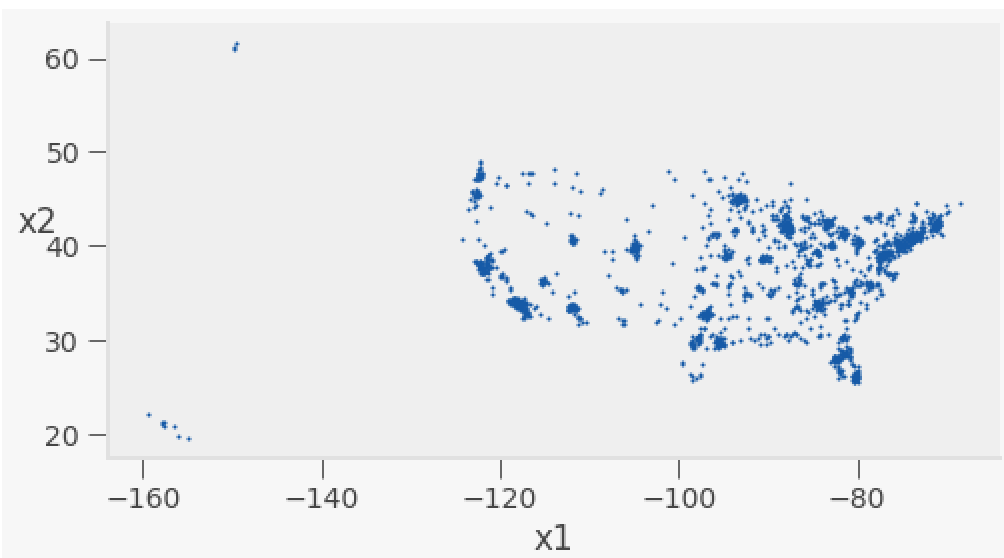
```
X.shape
```

```
(1839, 2)
```

```
kmeans = KMeans(n_clusters=100, random_state=2) #Definimos objeto kmeans con parametros n_clu
y_pred = kmeans.fit_predict(X) #Predecimos salida
```

```
#Definimos funcion
def clusters_plot (X, y=None):

    plt.scatter(X[:, 0], X[:, 1], c=y, s=1)
    plt.xlabel("x1")
    plt.ylabel("x2", rotation=0)
```

```
#Llamamos funcion y mostramos grafica
plt.figure(figsize=(8, 4))
clusters_plot(X)
plt.gca().set_axisbelow(True)
plt.grid()
plt.show()
```




```
y_pred #Mostramos y predecida
```

```
array([72,  9, 43, ..., 43, 22,  2], dtype=int32)
```

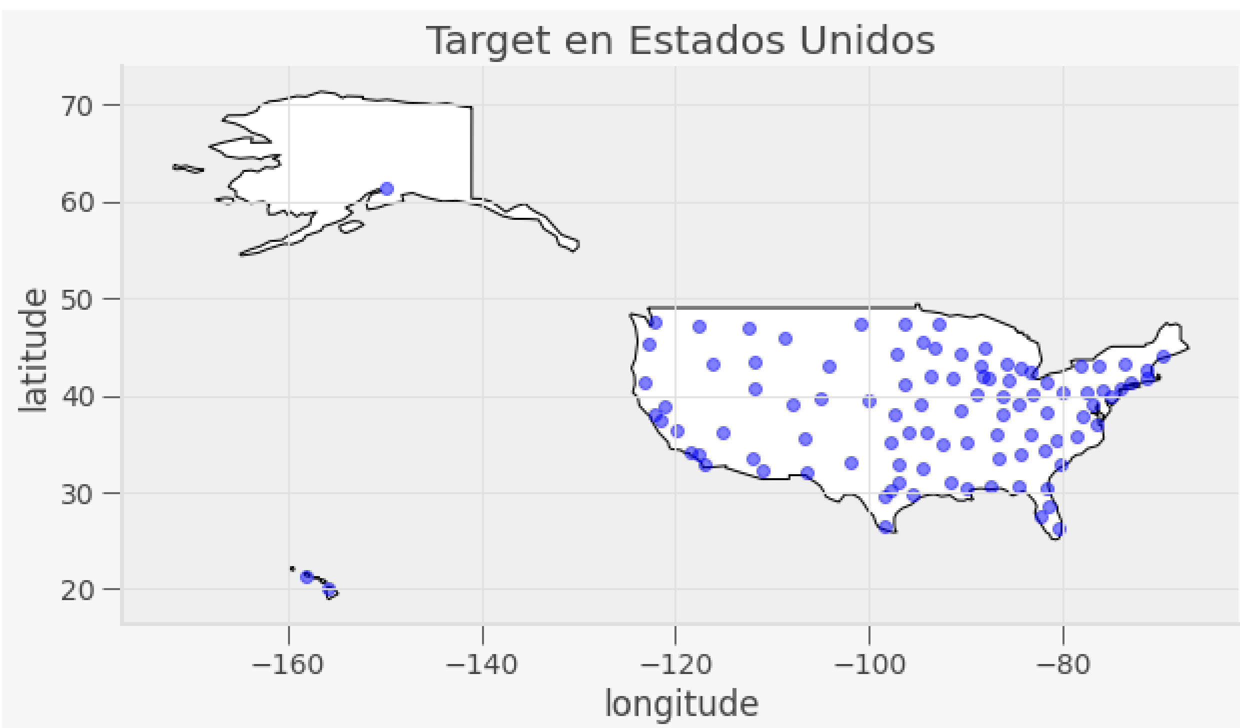
```
Clusters=kmeans.cluster_centers_
Clusters
```

```
       [ -78.47113973,   35.76042448],
       [-122.73175627,   45.25103523],
       [-100.90831003,   47.45449001],
       [ -87.70376935,   41.79667682],
```

```
       [ -80.04291603,   40.35455027],
       [ -90.00802009,   30.30842822],
       [ -72.98243246,   41.27351301],
       [ -96.28588706,   41.22556439],
       [-116.89064013,   32.92381061],
       [-112.4679378 ,   46.88608533],
       [-106.35100673,   31.97414993],
       [ -75.89179716,   40.46961067],
       [-121.5927009 ,   37.3837165 ],
       [ -91.38928989,   41.74055706],
       [ -97.82163952,   35.22119444],
       [ -89.87380436,   35.2481873 ],
       [ -98.37953827,   26.58229635],
       [ -86.23086048,   39.847359  ],
       [ -84.6489124 ,   30.63465268],
       [ -92.46755203,   34.90574706],
       [ -83.07949003,   40.10265828],
       [ -81.86759186,   34.235684  ],
       [-116.24103723,   43.2124044 ],
       [ -91.49810785,   31.04067197],
       [-155.79152443,   20.01054938],
       [-117.64961898,   33.83192449],
       [ -86.57790539,   33.41874223],
       [ -85.81713157,   43.15369902],
       [ -73.54484315,   43.22433036],
       [ -76.22272988,   42.97175131],
       [ -80.13261957,   32.85870697],
       [ -97.22781457,   44.34809271],
       [-107.82989994,   39.16251008],
       [ -96.31741692,   47.34161558],
       [ -75.07668911,   39.94991781],
       [ -83.3463613 ,   42.35934963],
       [ -76.40573822,   37.06030064],
       [-111.0062895 ,   32.17009971],
       [ -69.65508332,   44.05911464],
       [ -76.98813534,   39.00946902],
       [ -92.91266642,   47.28797696],
       [ -81.57868804,   30.32495605],
       [ -85.56345359,   41.6222008 ],
       [ -81.74803624,   38.17625463],
       [ -96.94184217,   30.97417068],
       [ -96.00530033,   36.11923369],
       [-108.7236249 ,   45.82446134],
       [ -71.2592803 ,   42.62247358],
       [ -88.28614698,   42.03306888],
       [ -86.18253671,   37.98407522],
       [-123.14742951,   41.31738226],
       [ -93.72556927,   41.88024912],
       [-100.01520195,   39.42785467],
       [ -87.9671316 ,   44.78615402],
       [-119.83243906,   36.34001446],
       [-121.06516903,   38.92545849],
       [ -97.78150597,   30.18704439],
       [-104.10128461,   42.96679808]])
```

```
#Definimos funcion para graficar el cluster de tiendas
def map_plot(Clusters):

    Clusters = pd.DataFrame(Clusters, columns = ['Lat','Long'])
    Clusters["Coordinates"] = list(zip(Clusters.Lat, Clusters.Long))
    Clusters["Coordinates"] = Clusters["Coordinates"].apply(Point)
    gdf = gpd.GeoDataFrame(Clusters, geometry="Coordinates")

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

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

    gdf.plot(ax=gax, color='Blue', 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)

    return plt.show()
```

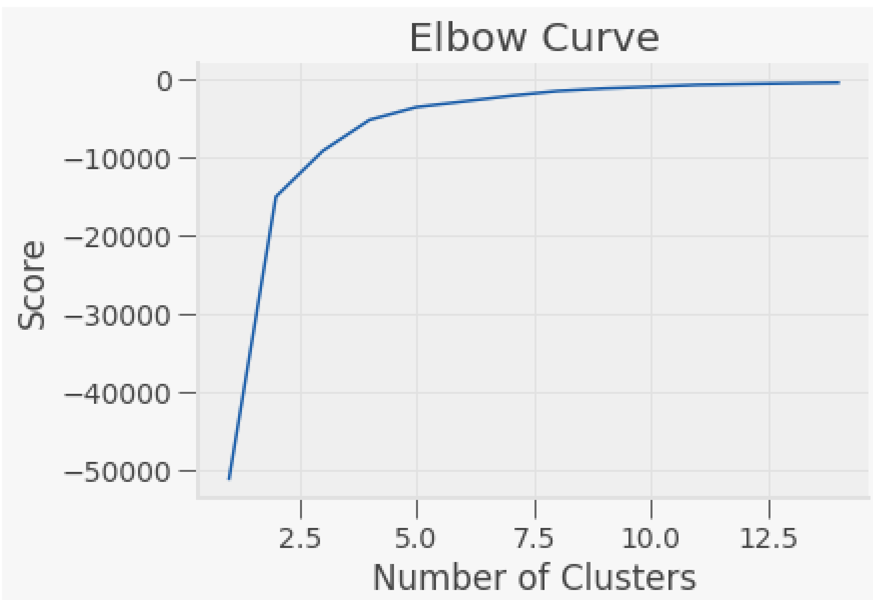
```
map_plot(Clusters) #Mostramos 100 tiendas
```




```
#Definimos rango de busqueda de valor
K_clusters_rango = range(1,15)
```

```
#Buscamos valor de kmean en rango
```

```
kmeans_2 = [KMeans(n_clusters=i) for i in K_clusters_rango]

#Definimos eje x y y
Y_axis = latlong[['latitude']]
X_axis = latlong[['longitude']]

#Determina score (valor de distancia de centroide con sus vecinos, entre mas vecinos valor ma
score = [kmeans_2[i].fit(Y_axis).score(Y_axis) for i in range(len(kmeans_2))]


#Graficamos
plt.plot(K_clusters_rango, score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```




```
kmeans = KMeans(n_clusters = 3, init ='k-means++') #definimos objeto Kmeans con 3 clusters

#Entrenamos modelo
kmeans.fit(latlong[latlong.columns[0:2]])

#Almacenamos resultado
labels_kmean = kmeans.labels_
labels_kmean
```

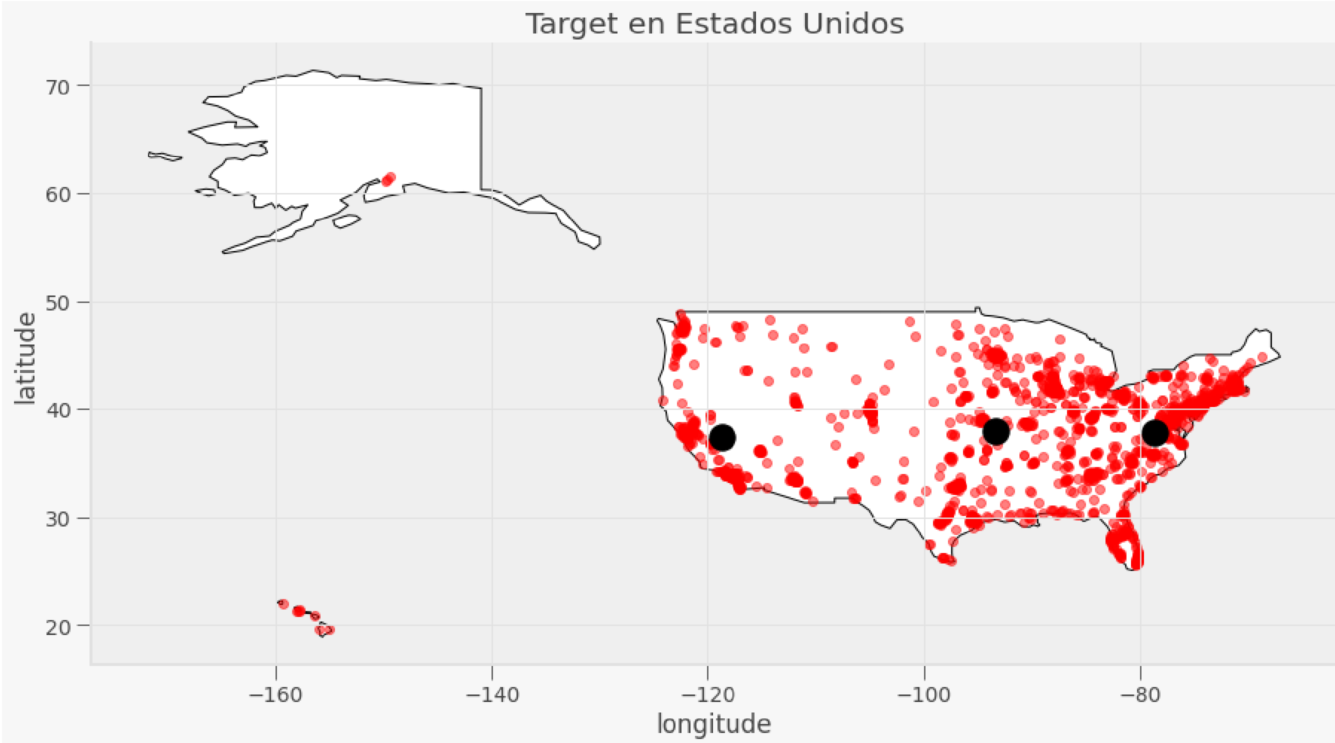
```
array([0, 0, 0, ..., 2, 0, 2], dtype=int32)
```

```
#Extraemos solo valores de ubicacion del dataframe
X = df[["longitude","latitude"]]

#Entrenamos modelo
kmeans = KMeans(n_clusters=3).fit(X)
```

```
#Definimos centroides de kmeans
centroids = kmeans.cluster_centers_

#Predecimos salida
labels = kmeans.predict(X)

#centroides
C = kmeans.cluster_centers_

#Definimos dataframe
Center_DF = pd.DataFrame(C)

#Creamos lista con coordenadas
Center_DF["Coordinates"] = list(zip(Center_DF[0], Center_DF[1]))
Center_DF["Coordinates"] = Center_DF["Coordinates"].apply(Point)


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

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

```
fig, gax = plt.subplots(figsize=(15,10))

# 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='wh

# This plot the cities. It's the same syntax, but we are plotting from a different GeoDataFra
# I want the cities as pale red dots.
gdf.plot(ax=gax, color='red', alpha = 0.5) #Aqui grafica los datos originales
gdf_C.plot(ax=gax, color='black', alpha = 1, markersize = 300) #Aqui grafica los datos de nue

#Grafica resultado
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()
```




```
gdf_C #Coordenadas de los 3 almacenes
```

	0	1	Coordinates
0	-78.569908	37.789554	POINT (-78.56991 37.78955)
1	-118.657146	37.481742	POINT (-118.65715 37.48174)
2	-93.347476	37.982702	POINT (-93.34748 37.98270)

```
#Importamos libreria
from pandas.core.internals.concat import concat_arrays

#Determinamos las localizacion/coordenadas de los almacenes
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)
```

```
37.98006260590112, -93.32717230430622
37.789554004474006, -78.56990807484885
```

```
37.48734203064935, -118.62447331844157
```

```
#¿qué ciudad es?

#Importamos libreria
from geopy.geocoders.yandex import Location
from geopy.geocoders import Nominatim
from geopy.distance import geodesic

geolocator = Nominatim(user_agent="my-application")

#Ingresamos coordenadas de los 3 almacenes
Locations = [Location1, Location2, Location3]

for i in Locations:

  #Definimos la ciudad usando las coordenadas
  location = geolocator.reverse(i)
  print('Localizacion de almacen en ---', location.address)
```

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

```
#¿a cuantas tiendas va surtir?

##Determinamos la cantidad de tiendas que le corresponderan a cada cluster
latlong['kmeans'] = kmeans.labels_
latlong.loc[:, 'kmeans'].value_counts()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: 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_
  after removing the cwd from sys.path.
1    826
0    628
2    385
Name: kmeans, dtype: int64
```

```
#¿sabes a que distancia estará?

#Determinamos distancias entre almacenes
distancia12 = str(geodesic(Location1, Location2).miles)
print("\nDistancia entre el almacen 1 y 2 : ", distancia12, " mi \n")

distancia23 = str(geodesic(Location2, Location3).miles)
```

```
print("Distancia entre el almacen 2 y 3 : ", distancia23, " mi \n")
```

```
Distancia entre el almacen 1 y 2 :  2181.490837424155  mi

Distancia entre el almacen 2 y 3 :  1382.4721323777403  mi
```

Encuentra las latitudes y longitudes de los almacenes,

¿qué ciudad es?

1-Localizacion de almacen en --- Langhorne Road, Totier Hills, Albemarle County, Virginia, 22946, United States

2-Localizacion de almacen en --- Mono County, California, United States

3-Localizacion de almacen en --- State Highway Y, Hickory County, Missouri, 65732, United States

¿a cuantas tiendas va surtir?

1- 827 Tiendas

2- 627 Tiendas

3- 385 Tiendas

¿sabes a que distancia estará?

Distancia entre el almacen 1 y 2 : 2181.49 mi

Distancia entre el almacen 2 y 3 : 1382.47 mi

¿Cómo elegiste el número de almacenes?

Haciendo uso de la grafica de codo, y valiendonos del valor de score graficado. A medida que el score era mayor y negativo, el almacen tenia menos tiendas a sus alrededores. es por esa razon que se escogieron 3 tiendas, las cuales nos presenta un valor de 10,000, mientras que a partir de el almacen 4 la diferencia entre el valor de score es muy pequeña, por lo cual no se consideraria conveniente hacer uso de ese 4to almacen.

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

Geopandas es una librería de gran utilidad para graficar casos como este en particular, donde es requerido trabajar con datos geo espaciales. esta libreria combina las capacidades de pandas and shapely. Geopandas nos permite realizar de forma sencilla operaciones geo espaciales en python sin la necesidad de utilizar bases de datos espaciales tales como PostGIS.

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

Si, es de gran importancia la visualización de los resultados, esto debido a que el ser humano es un ser visual. La visión es el sentido que más influencia tiene en nuestra toma de desiciones. Por tal razón, el hecho de contar con un mapa de los datos de trabajo, nos permitirá realizar un mejor análisis del problema en cuestión. Además de facilitarnos la comprensión de los datos, acelerando el proceso de la toma de desiciones.

Agrega las conclusiones

Hemos aprendido como resolver un problema que pueden llegar a presentar las cadenas de suministros mediante el uso de los algoritmos de agrupamiento o clustering.

Gracias a KMean se pudo determninar la cantidad y ubicación mas óptima de almacenes para suministrar o abastecer a las diferentes tiendas distribuidas a lo largo de Estados Unidos.

En este ejercicio en particular, observamos que nuestro resultado consistió en 3 tiendas distribuidas al este, centro y oeste del país. Según la gráfica de codo que mostramos anteriormente, determinamos que 2 almacenes eran muy poco, ya que habría varias tiendas que quedarían retiradas, mientras que 4 almacenes no mejoraban de manera significativa el score graficado.

Referencias:

[1] Géron, A. (s. f.). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition. O'Reilly Online Learning. https://www.oreilly.com/library/view/hands-on-machine-learning/9781098125967/

0s completed at 10:38 PM