

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-datasience/blob/master/applications/maps.ipynb#scrollTo=u02oPtSCeAOz>

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

```
Requirement already satisfied: qeds in c:\users\oviwa\appdata\local\programs\python\p  
Requirement already satisfied: fiona in c:\users\oviwa\appdata\local\programs\python\p  
Requirement already satisfied: geopandas in c:\users\oviwa\appdata\local\programs\pyt  
Requirement already satisfied: xgboost in c:\users\oviwa\appdata\local\programs\pytho  
Requirement already satisfied: gensim in c:\users\oviwa\appdata\local\programs\python
```

```
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)
```

Exploraremos los datos.

```
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
				4889 Promenade Plaza	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 (0°). 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 (0°). La longitud puede ser este y oeste.

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

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

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")
```

```
latlong.describe()
```

latitude	longitude
----------	-----------

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

```
51.160000 10.100000
```

```
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();

findfont: Font family 'Source Sans Pro' not found.

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

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

```
findfont: Font family 'Source Sans Pro' not found.
```

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

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

```
#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	geometry
iso_a3						
FJI	889953.0	Oceania		Fiji	5496	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...))
TZA	58005463.0	Africa		Tanzania	63177	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...))
ESH	603253.0	Africa		W. Sahara	907	POLYGON ((-8.66559 27.65643, -8.66512))

```
#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.', 'North 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()
```

TUTORIAL: FONT FAMILY Source Sans Pro NOT FOUND.

¿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.

```
findfont: Font family 'Source Sans Pro' not found.
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

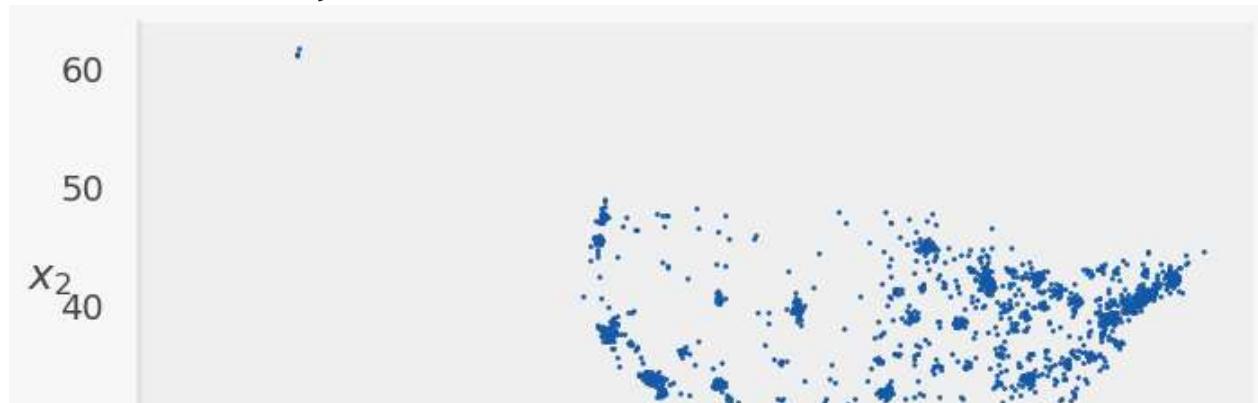
findfont: Font family 'Source Sans Pro' not found.
m=list(zip(df.longitude, df.latitude))

findfont: Font family 'Source Sans Pro' not found
```

m

```
[(-86.80417369999999, 33.2242254),  
 (-86.98977789999999, 33.3345501),  
 (-87.89593169999999, 30.6028747),  
 (-86.9715595, 34.5601477),  
 (-85.4464222, 31.2660613),  
 (-87.6339077, 34.8420853),  
 (-86.80555729999999, 33.6060399),  
 (-87.6799946, 30.3039796),  
 (-86.77509979999999, 33.467142),  
 (-86.7562505, 34.7488201),  
 (-86.1600739, 32.3606937),  
 (-85.4050434, 32.6204324),  
 (-85.7846536, 33.6092005),  
 (-86.3957891, 32.4570652),  
 (-87.51825509999999, 33.1923759),  
 (-86.7125552, 33.425259),  
 (-86.6376483, 33.6054494),  
 (-86.8532462, 33.3616601),  
 (-86.68163709999999, 34.7446779),  
 (-86.5443483, 34.674648),  
 (-88.1204442, 30.6709831),  
 (-88.2260919, 30.6734013),  
 (-149.4029599, 61.5779193),  
 (-149.745967, 61.2299524),  
 (-149.8814512, 61.13006249999999),  
 (-114.5918998, 35.0528237),  
 (-111.6591788, 35.1840575),  
 (-111.7184273, 33.5733931),  
 (-110.9601638, 32.3990239),  
 (-112.4315678, 34.549193),  
 (-111.6407335, 33.2552871),  
 (-110.2558558, 31.5561559),  
 (-112.3537192, 33.6368072),  
 (-112.269511, 33.4236583),  
 (-114.6061139, 32.7050523),  
 (-111.8995238, 33.29558859999999),  
 (-111.8376548, 33.2503002),  
 (-111.7582987, 33.3342412),  
 (-111.785687, 33.2786159),  
 (-112.2568904, 33.5524031),  
 (-112.1532676, 33.5802437),  
 (-112.3550121, 33.46325460000001),  
 (-112.4213234, 33.4373884),  
 (-111.7855007, 33.3943485),  
 (-111.6830273, 33.3884182),  
 (-111.6826546, 33.4545859),  
 (-111.8652426, 33.3916851),  
 (-111.6903953, 33.3230012),  
 (-112.228156, 33.6356253),  
 (-112.2758045, 33.7101161),  
 (-111.9896967, 33.6004996),  
 (-112.2179103, 33.47579810000001),  
 (-111.9811417, 33.3218984),  
 (-111.98431, 33.47888),
```

```
(-112.1218108, 33.6711241),  
(-111.97187, 33.675998),  
(-112.0347245, 33.3793622),  
(-112.0688753, 33.6381086),  
  
blob_centers = m  
  
X, y = make_blobs(n_samples=1839, centers=blob_centers, 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)  
  
k = 5  
kmeans = KMeans(n_clusters=k, random_state=42)  
y_pred = kmeans.fit_predict(X)  
  
def plot_clusters(X, y=None):  
    plt.scatter(X[:, 0], X[:, 1], c=y, s=1)  
    plt.xlabel("$x_1$")  
    plt.ylabel("$x_2$", rotation=0)  
  
plt.figure(figsize=(8, 4))  
plot_clusters(X)  
plt.gca().set_axisbelow(True)  
plt.grid()  
plt.show()
```



y_pred

```
array([4, 0, 1, ..., 1, 3, 3])
```

1

y pred is kmeans.labels

True

```

Clusters=kmeans.cluster_centers_
Clusters

array([[ -88.38560066,   41.74051783],
       [-119.18772238,   37.56156168],
       [ -98.2896299 ,  34.37940884],
       [ -75.59459074,  40.33885587],
       [ -82.93798241,  31.07271813]]))

def mappita(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")
# 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

# 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='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()

```

► Posicion de los almacenes propuestos

[] ↴ 5 celdas ocultas

► Numero de linea de la tienda ma Cercana

[] ↴ 1 celda oculta

► Direccion de la tienda mas cercana

[] ↴ 43 celdas ocultas

- ▶ Encuentra las latitudes y longitudes de los almacenes, ¿que ciudad es?, ¿a cuantas tiendas va surtir?, ¿sabes a que distancia estara?

– Despues de correr varias veces el sistema y llegar los mejore valores se encontro que las ciudades donde conviene poner un alamcen son las mostradas en la siguiente tabla – La cantidad de tiendas a la que surtira son tambien mostradas en la tabla – La distancia exacta a la que estara de cada una de las tiendas es mostrada en la tabla 2 la cual nos muestra la distancia de cada almacen a cada tienda

[] ↴ 3 celdas ocultas

- ▶ ¿Cómo elegiste el numero de almacenes?, justifica tu respuesta tecnicamente?

Inicialmente se decido analizar la opcion de tener el mismo numero de almacenes que unda de las empresas mas grandes del mundo la cual es Bimbo, donde ellos tienen un total de 64 Almacenes alrededor de todo Mexico Despues de esto se econtró que la mejor opcion para esto es calcular el elbow de K y asi obtener la mejor respuesta dejando de lado los puntos que estan muy fuera de lo esperado como Alaska y la isla

Adicionalmente, en el notebook notaras que al inicio exploramos los datos y los graficamos de manera simple, despues nos auxiliamos de una librería de datos geograficos.

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

Esta ves utilizamos varias librerias , existen otras como folium y Geopandas siendo las utilizadas el dia de hoy

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

Si es necesario para tener de forma grafica una idea de que es lo que estamos buscando y necesitamos encontrar

Agrega las conclusiones

El uso de herramientas como esta nos ayuda a poder predecir de forma mas precisa donde establecer un nuevo objetivo no solo para tienda o cosas asi si no tambien para sistemas de vision y sensores

[] ↴ 1 celda oculta

[Productos de pago de Colab](#) - Cancelar contratos

