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
```

```
      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: certifi in /usr/local/lib/python3.7/dist-packag
    Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-pac
    Collecting pyproj>=2.2.0
```

```
  Downloading pyproj-3.2.1-cp37-cp37m-manylinux2010_x86_64.whl (6.3 MB)
     |████████████████████████████████| 6.3 MB 36.2 MB/s
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/d
Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/di
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: numexpr in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /us
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/d
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/d
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/d
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist
Collecting inflection>=0.3.1
  Downloading inflection-0.5.1-py2.py3-none-any.whl (9.5 kB)
Requirement already satisfied: sympy in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/py
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.7/dist-p
Building wheels for collected packages: qeds, pyLDAvis, sklearn
  Building wheel for qeds (setup.py) ... done
  Created wheel for qeds: filename=qeds-0.7.0-py3-none-any.whl size=27812 sha2
  Stored in directory: /root/.cache/pip/wheels/fc/8c/52/0cc036b9730b75850b9845
  Building wheel for pyLDAvis (PEP 517) ... done
  Created wheel for pyLDAvis: filename=pyLDAvis-3.3.1-py2.py3-none-any.whl siz
  Stored in directory: /root/.cache/pip/wheels/c9/21/f6/17bcf2667e8a68532ba2fb
  Building wheel for sklearn (setup.py) ... done
  Created wheel for sklearn: filename=sklearn-0.0.post1-py3-none-any.whl size=
  Stored in directory: /root/.cache/pip/wheels/42/56/cc/4a8bf86613aafd5b7f1b31
Successfully built qeds pyLDAvis sklearn
Installing collected packages: munch, inflection, cligj, click-plugins, sklear
Successfully installed click-plugins-1.1.1 cligj-0.7.2 fiona-1.8.22 funcy-1.17
```

```
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	
0	Alabaster	33.224225	-86.804174	250 S Colonial Dr, Alabaster, AL 35007-4657	205-564-2608	https://www.target.com/s
1	Bessemer	33.334550	-86.989778	4889 Promenade Pkwy, Bessemer, AL 35022-7305	205-565-3760	https://www.target.com/s
2	Daphne	30.602875	-87.895932	1698 US Highway 98,	251-621-	https://www.target.com

```
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.

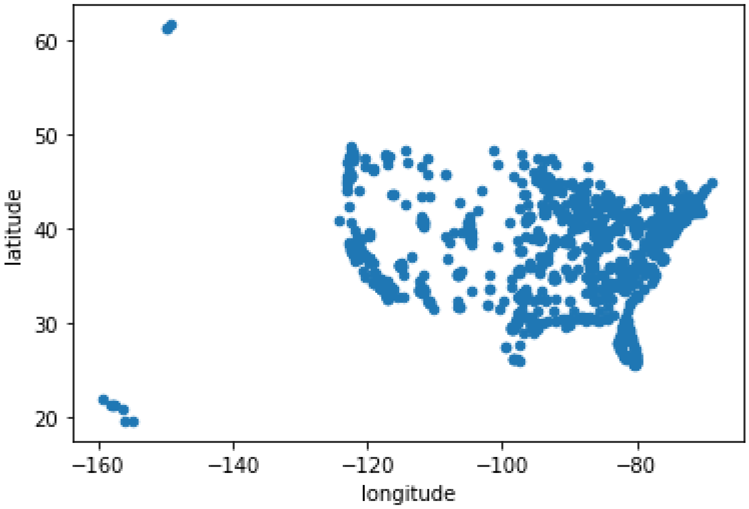
```
latlong=df[["latitude","longitude"]]
```

¡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 0x7f2370044590>
```



```
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.

std	5.272299	16.108046

```
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
1	Bessemer	33.334550	-86.989778	4889 Promenade Pkwy, Bessemer	205-565-	https://www.target.com/sl/bessemer/2

```
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
1	Bessemer	33.334550	-86.989778	4889 Promenade Pkwy, Bessemer	205-565-	https://www.target.com/sl/bessemer/2

```
#mapa
```

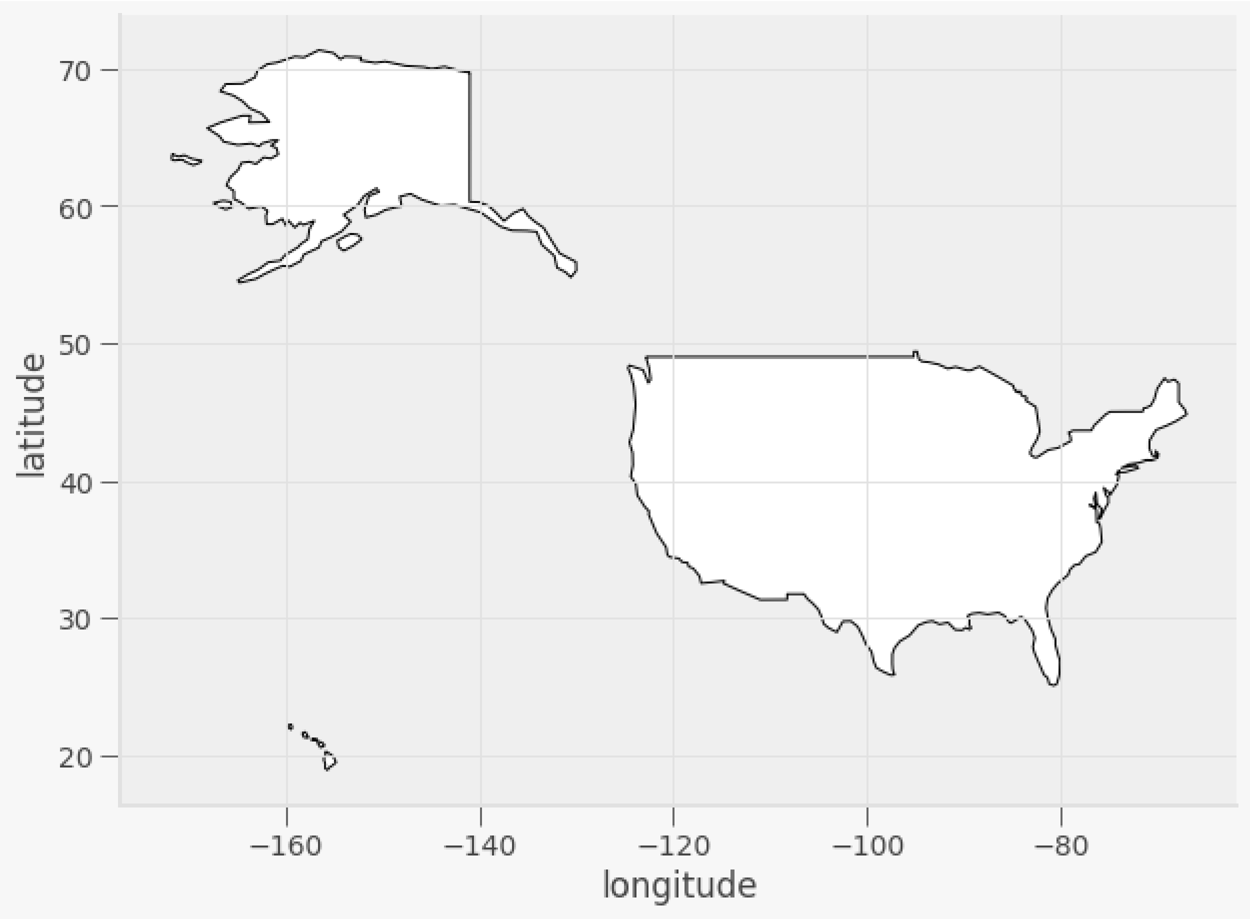
```
world = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))
world = world.set_index("iso_a3")

world.head()
```

	pop_est	continent	name	gdp_md_est	
iso_a3					
FJI	920938	Oceania	Fiji	8374.0	MULTIPOLYGON (((180.
TZA	53950935	Africa	Tanzania	150600.0	POLYGON ((33.90371 -0
ESH	603253	Africa	W. Sahara	906.5	POLYGON ((-8.66559 27
CAN	35623680	North	Canada	1674000.0	MULTIPOLYGON (((-122

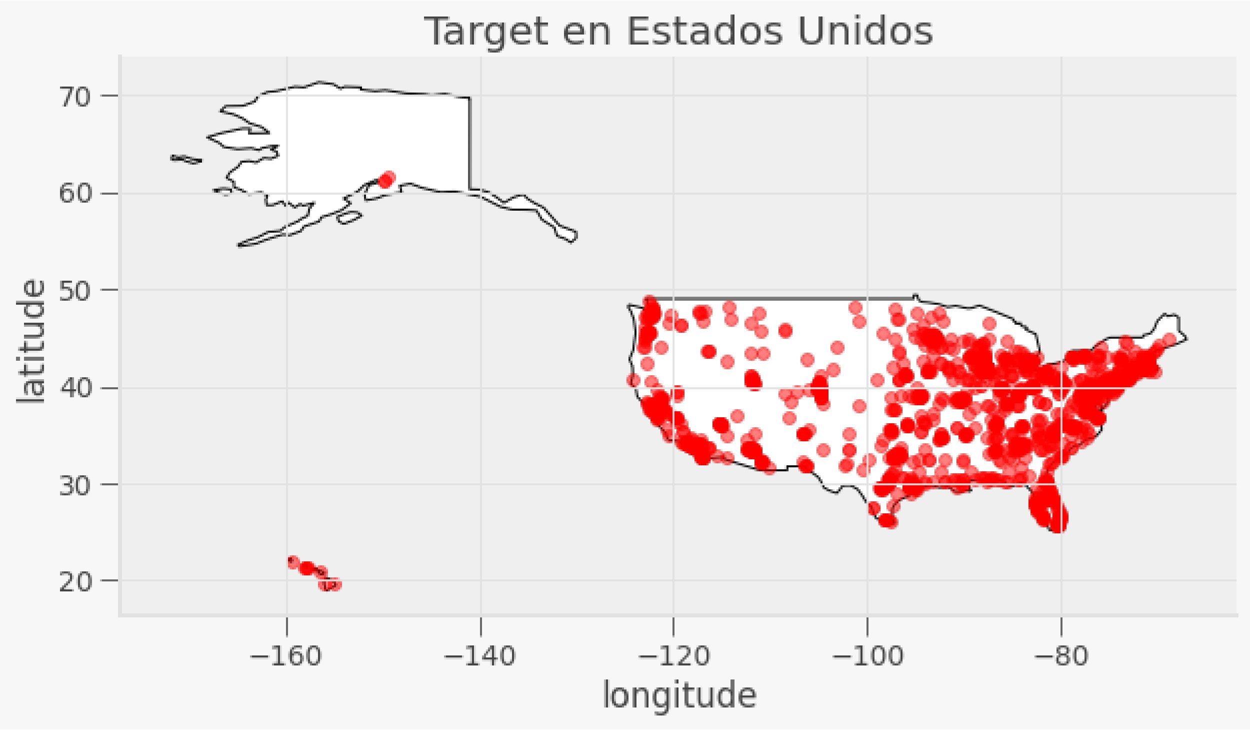
```
#graficar el mapa
world.name.unique()
```

```
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',colo

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



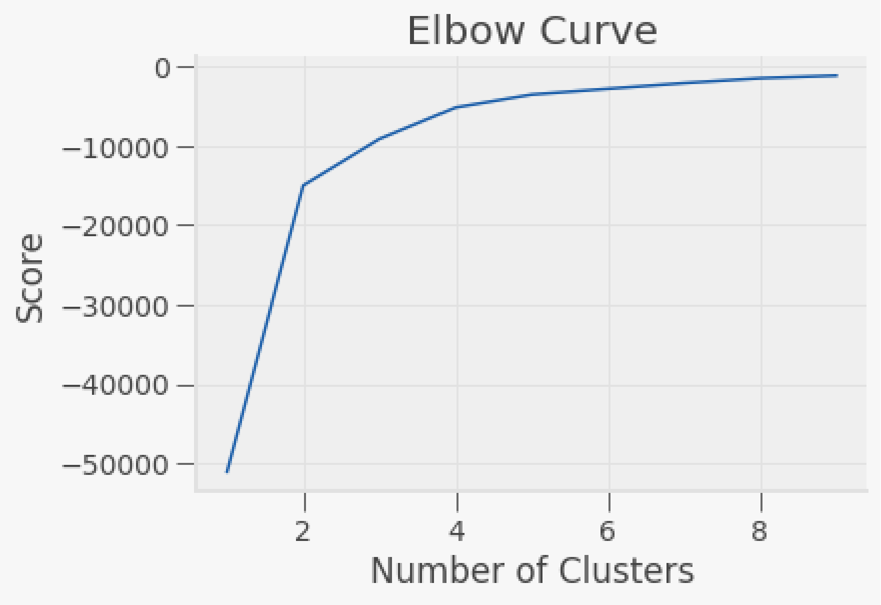
¿qué tal ahora?, tiene mayor sentido verdad, entonces los datos lejanos no eran atípicos, de aquí la importancia de ver los datos con el tipo de gráfica correcta.

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()
```



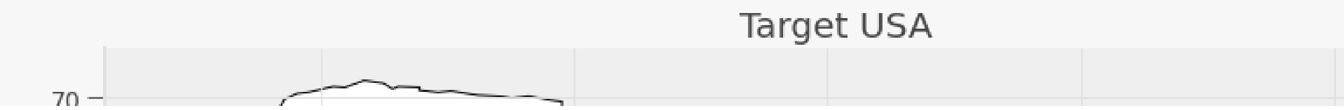
```
kmeans = KMeans(n_clusters = 3, init ='k-means++')
kmeans.fit(latlong[latlong.columns[0:2]])
labels = kmeans.labels_
labels
```

```
    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
```

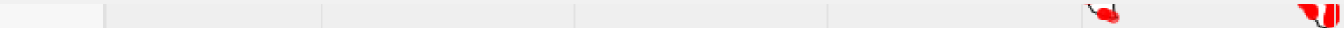
	0	1	Coordinates
0	-78.569908	37.789554	POINT (-78.56991 37.78955)
1	-93.327172	37.980063	POINT (-93.32717 37.98006)
2	-118.624473	37.487342	POINT (-118.62447 37.48734)

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

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

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 USA')

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

plt.show()
```



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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyW
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/stab
  """Entry point for launching an IPython kernel.
0    826
1    628
2    385
Name: kmeans, dtype: int64
```



```
gdf_C
```

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

```
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
37.98006260590112, -93.32717230430622
37.48734203064935, -118.62447331844157
```

```
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:', location.address)
```

```
Almacen: Langhorne Road, Totier Hills, Albemarle County, Virginia, 22946, United
Almacen: Hickory County, Missouri, United States
Almacen: Paradise Estates, Mono County, California, United States
```

```
distancia12 = str(geodesic(Location1, Location2).miles)
print("\nDistancia entre primer-segundo almacen : ", distancia12, " miles")
distancia23 = str(geodesic(Location2, Location3).miles)
print("Distancia entre segundo-tercer almacen : ", distancia23, "miles")
```

```
Distancia entre primer-segundo almacen :  805.9209470497035  miles
Distancia entre segundo-tercer almacen :  1381.7597109962394 miles
```

Conclusiones

Encuentra las latitudes y longitudes de los almacenes, ¿qué ciudad es?, ¿a cuantas tiendas va surtir?, ¿sabes a que distancia estará? -78.569908 37.789554 POINT (-78.56991 37.78955) 1 -93.327172 37.980063 POINT (-93.32717 37.98006) 2 -118.624473 37.487342 POINT (-118.62447 37.48734)

Almacen: Langhorne Road, Totier Hills, Albemarle County, Virginia, 22946, United States Almacen: Hickory County, Missouri, United States Almacen: Paradise Estates, Mono County, California, United States

Distancia entre primer-segundo almacen : 805.9209470497035 millas Distancia entre segundo-tercer almacen : 1381.7597109962394 millas

¿Cómo elegiste el número de almacenes?, justifica tu respuesta técnicamente. Con la grafica de 'Elbow curve' se puede apreciar como la curva se va desarrollando y se pudiera agrupar las diferentes tiendas (puntos) con tan solo 3 almacenes. Mas almacenes pudieran resultar redundantes.

¿qué librerías nos pueden ayudar a graficar este tipo de datos? GeoDataFrame nos pudiera ayudar a obtener los datos de manera geografica y utilizar el matplot para visualizar los datos.

¿Consideras importante que se grafique en un mapa?, ¿por qué? si creo que sea importante dado que visualimente se pudiera ver mejor las agrupaciones y ver con mayor claridad los clusters.

✓ 1s completed at 7:02 PM ● ×