

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
```

```
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/2276
				4889 Promenade	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.

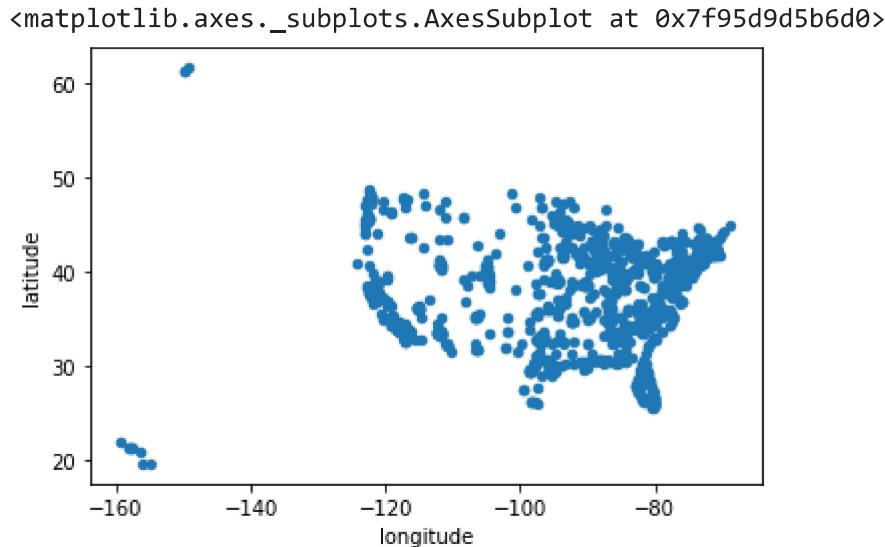
Longitud: Es la distancia en grados, minutos y segundos que hay con respecto al meridiano principal, que es el meridiano de Greenwich (0°). La longitud puede ser este y oeste.

```
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.

slu 33.224225 -86.804174

```
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
0	Alabaster	33.224225	-86.804174	250 S Colonial Dr, Alabaster, AL 35007-4657	205-564-2608	https://www.target.com/sl/alabaster/2276
				4889 Promenade	205-	

```
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
				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	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 -122.84000, -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',
```

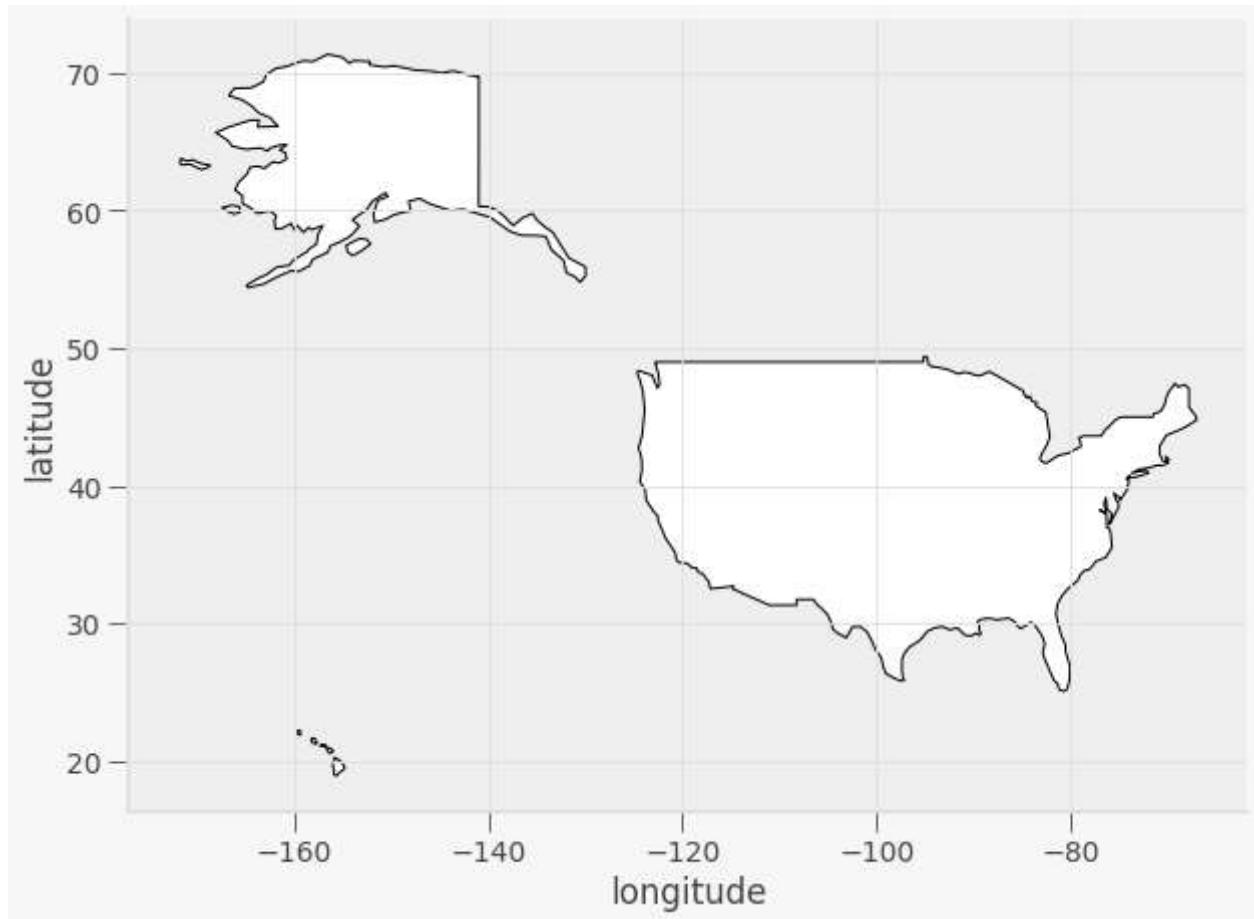
```
'Egypt', 'Libya', 'Ethiopia', 'Djibouti', 'Somaliland', 'Uganda',
'Rwanda', 'Bosnia and Herz.', 'Macedonia', 'Serbia', 'Montenegro',
'Kosovo', 'Trinidad and Tobago', 'S. Sudan'], dtype=object)
```

```
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='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.
```

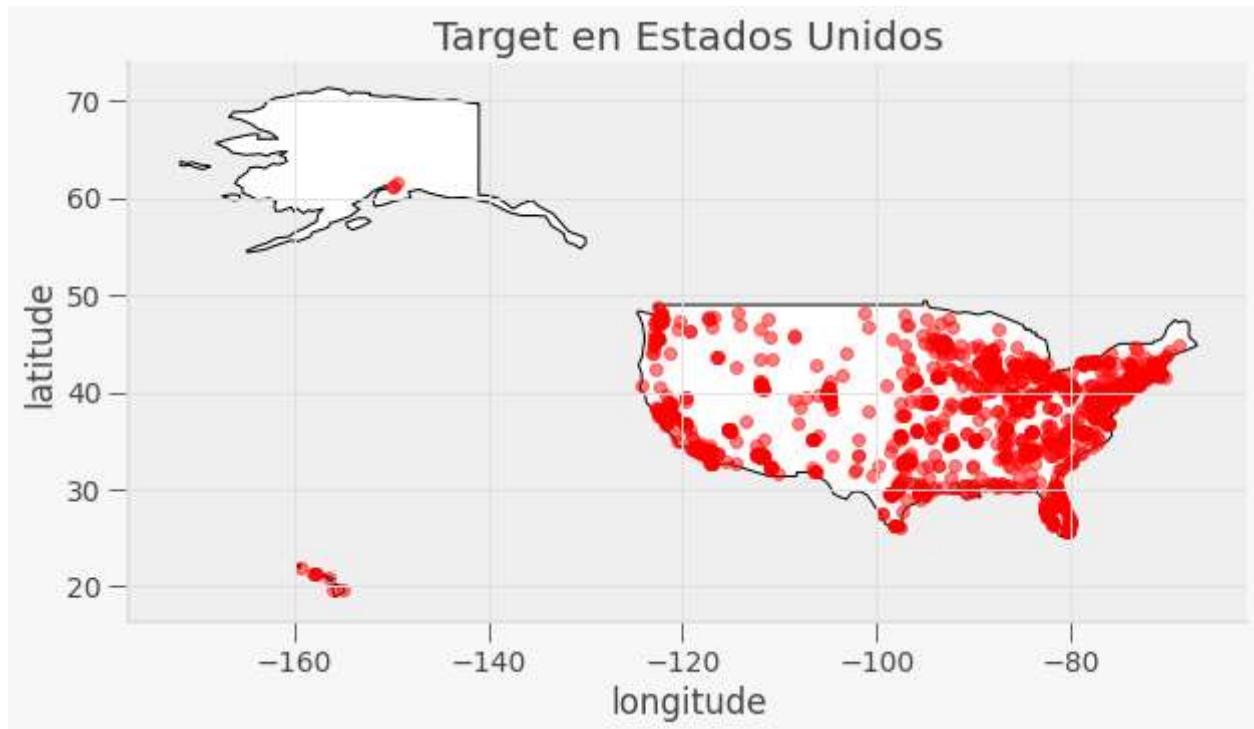
```
world.query("name == 'United States of America'").plot(ax = gax, edgecolor='black', color='wh
```

```
# 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.

```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

m=list(zip(df.longitude, df.latitude))
```

m

```
(-93.2484629, 44.0866616),  
(-93.4507297, 45.0317265),  
(-92.5828117, 44.5694605),  
(-93.2484629, 44.8855423),  
(-93.5538446, 45.1970446),  
(-93.1643718, 45.0080432),  
(-93.3770002, 44.7437151),  
(-93.5015727, 44.77603080000001),  
(-93.1439917, 45.0563308),  
(-93.3933865, 44.9360278000001),  
(-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),  
(-88.8995200999999, 30.4557758),  
(-90.0613934, 32.3435065),  
(-89.384664, 31.3239115),  
(-90.0051002000002, 34.9663371),  
(-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),  
  
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], dtype=int32)
```

```
30 -
```

```
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')
```

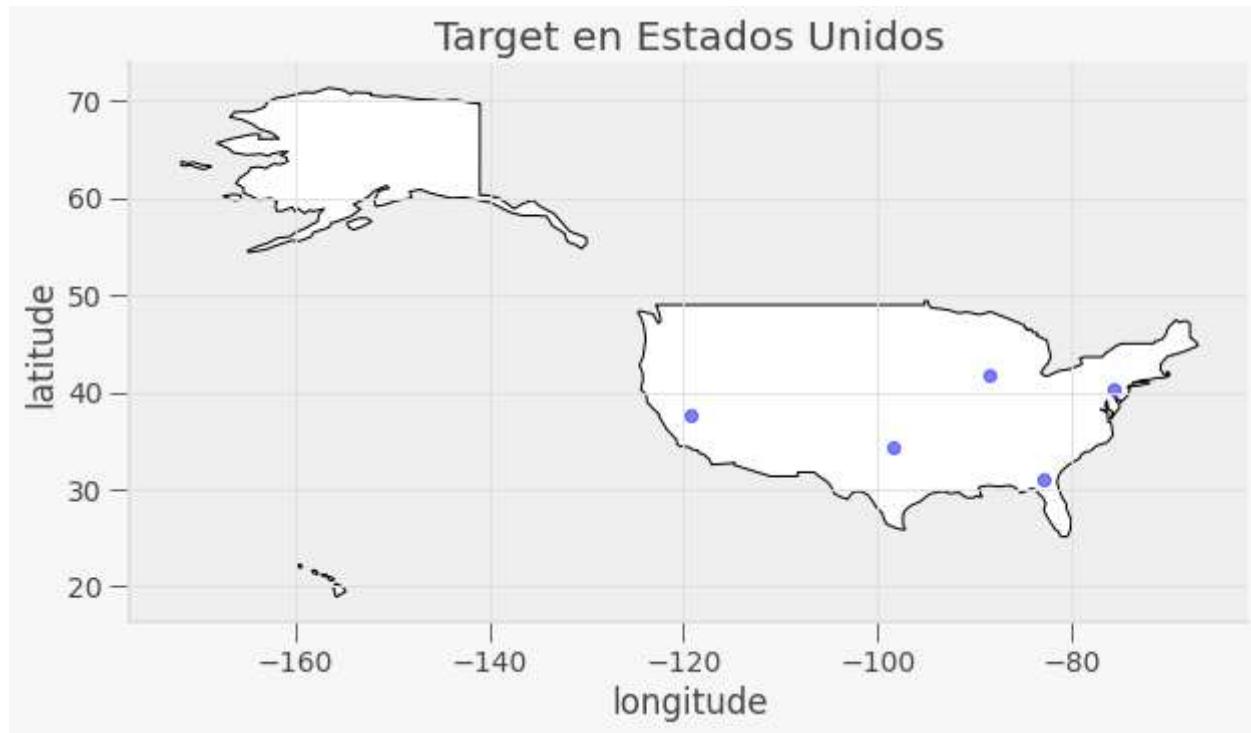
```
    gax.spines['top'].set_visible(False)
```

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

```
return plt.show()
```

▼ Posicion de los almacenes propuestos

```
mappita(Clusters)
```



```
kmeans.inertia_
```

```
62068.76665807959
```

```
kmeans.transform(X).round(2)
```

```
array([[11.87, 38.13, 16.99, 10.1 , 1.81],
       [ 6.02, 32.7 , 11.88, 11.76, 6.1 ],
       [34.97, 9.05, 27.3 , 47.84, 42.78],
       ...,
       [34.59, 8.86, 26.95, 47.46, 42.41],
       [16.7 , 47.69, 27.6 , 4.15, 15.49],
       [14.72, 45.64, 25.48, 2.02, 13.56]])
```

```
np.linalg.norm(np.tile(X, (1, k)).reshape(-1, k, 2)
              - kmeans.cluster_centers_, axis=2).round(2)
```

```
array([[11.87, 38.13, 16.99, 10.1 , 1.81],
       [ 6.02, 32.7 , 11.88, 11.76, 6.1 ],
       [34.97, 9.05, 27.3 , 47.84, 42.78],
```

```
...,
[34.59, 8.86, 26.95, 47.46, 42.41],
[16.7 , 47.69, 27.6 , 4.15, 15.49],
[14.72, 45.64, 25.48, 2.02, 13.56]])
```

▼ Numero de linea de la tienda mas Cercana

```
from sklearn.metrics import pairwise_distances_argmin_min
#mas cercano
closest, _ = pairwise_distances_argmin_min(Clusters, X)
closest

array([ 211, 152, 1624, 1822, 558])
```

▼ Direccion de la tienda mas cercana

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

```
910 Eastlake Pkwy, Chula Vista, CA 91914-3558
8800 Whittier Blvd, Pico Rivera, CA 90660-2658
10801 Westheimer Rd, Houston, TX 77042-3201
660 S Grand Ave, Sun Prairie, WI 53590-9832
4701 N Illinois St, Fairview Heights, IL 62208-3416
```

```
# Predicting the clusters
labels = kmeans.predict(X)
# Getting the cluster centers
C = kmeans.cluster_centers_
colores=['Almacen 1 California ','Almacen 2 California ','Almacen 3 Texas','Almacen 4 Wisconsin']
asignar=[]
```

Lugar de la tienda mas cercana y cuantas tiendas tiene al rededor

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



	Tienda	cantidad
0	Almacen 1 California	427
1	Almacen 2 California	368
2	Almacen 3 Texas	292
3	Almacen 4 Wisconsin	480
4	Almacen 5 Illinois	272

```

def plot_data(X):
    plt.plot(X[:, 0], X[:, 1], 'k.', markersize=2)

def plot_centroids(centroids, weights=None, circle_color='w', cross_color='k'):
    if weights is not None:
        centroids = centroids[weights > weights.max() / 10]
    plt.scatter(centroids[:, 0], centroids[:, 1],
                marker='o', s=35, linewidths=8,
                color=circle_color, zorder=10, alpha=0.9)
    plt.scatter(centroids[:, 0], centroids[:, 1],
                marker='x', s=2, linewidths=12,
                color=cross_color, zorder=11, alpha=1)

def plot_decision_boundaries(clusterer, X, resolution=1000, show_centroids=True,
                             show_xlabels=True, show_ylabels=True):
    mins = X.min(axis=0) - 0.1
    maxs = X.max(axis=0) + 0.1
    xx, yy = np.meshgrid(np.linspace(mins[0], maxs[0], resolution),
                         np.linspace(mins[1], maxs[1], resolution))
    Z = clusterer.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    W=Z

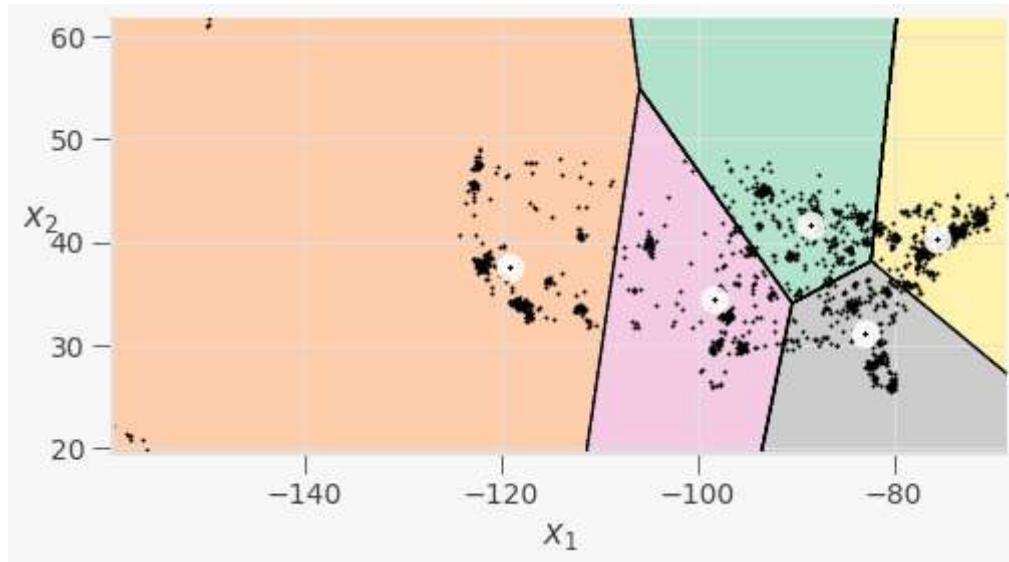
    plt.contourf(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]),
                  cmap="Pastel2")
    plt.contour(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]),
                linewidths=1, colors='k')
    plot_data(X)
    if show_centroids:
        plot_centroids(clusterer.cluster_centers_)

    if show_xlabels:
        plt.xlabel("$x_1$")
    else:
        plt.tick_params(labelbottom=False)
    if show_ylabels:
        plt.ylabel("$x_2$", rotation=0)
    else:
        plt.tick_params(labelleft=False)

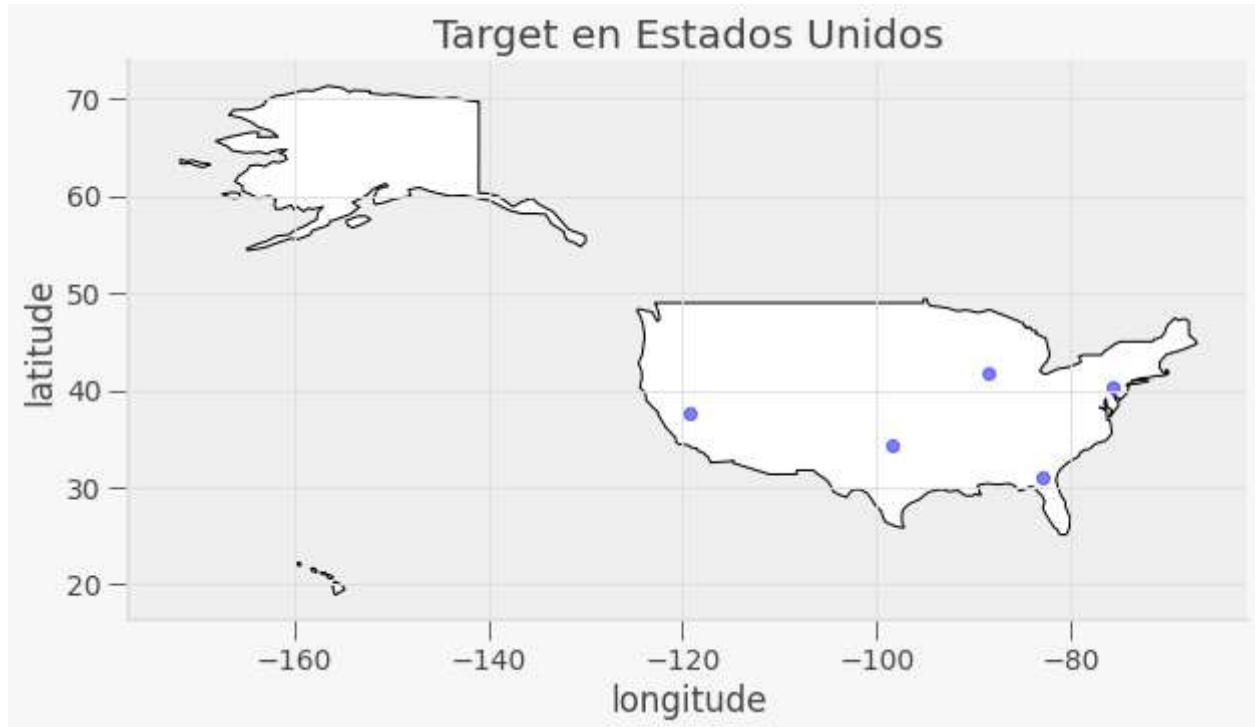
```

```
plt.figure(figsize=(8, 4))
plot_decision_boundaries(kmeans, X)
plt.show()
```

```
Clusters2=kmeans.cluster_centers_
```



```
mappita(Clusters2)
```



```
from sklearn.metrics import pairwise_distances_argmin_min
closest, _ = pairwise_distances_argmin_min(Clusters2, X)
closest
```

```

users=df[ 'address' ].values
for row in closest:
    print(users[row])

910 Eastlake Pkwy, Chula Vista, CA 91914-3558
8800 Whittier Blvd, Pico Rivera, CA 90660-2658
10801 Westheimer Rd, Houston, TX 77042-3201
660 S Grand Ave, Sun Prairie, WI 53590-9832
4701 N Illinois St, Fairview Heights, IL 62208-3416

```

```

from mpl_toolkits.mplot3d import Axes3D

# Predicting the clusters
labels = kmeans.predict(X)
# Getting the cluster centers
C = kmeans.cluster_centers_
colores=['Almacen 1','Almacen 2 ','Almacen 3','Almacen 4 ','Almacen 5 ']
asignar=[]

```

```

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

```

	Tienda	cantidad	
0	Almacen 1	427	
1	Almacen 2	368	
2	Almacen 3	292	
3	Almacen 4	480	
4	Almacen 5	272	

```

kmeans_iter1 = KMeans(n_clusters=5, init="random", n_init=1, max_iter=1,
                      random_state=5)
kmeans_iter2 = KMeans(n_clusters=5, init="random", n_init=1, max_iter=2,
                      random_state=5)
kmeans_iter3 = KMeans(n_clusters=5, init="random", n_init=1, max_iter=3,
                      random_state=5)
kmeans_iter1.fit(X)
kmeans_iter2.fit(X)

```

```
kmeans_iter3.fit(X)

plt.figure(figsize=(10, 8))

plt.subplot(321)
plot_data(X)
plot_centroids(kmeans_iter1.cluster_centers_, circle_color='r', cross_color='w')
plt.ylabel("$x_2$", rotation=0)
plt.tick_params(labelbottom=False)
plt.title("Update the centroids (initially randomly)")

plt.subplot(322)
plot_decision_boundaries(kmeans_iter1, X, show_xlabels=False,
                         show_ylabels=False)
plt.title("Label the instances")

plt.subplot(323)
plot_decision_boundaries(kmeans_iter1, X, show_centroids=False,
                         show_xlabels=False)
plot_centroids(kmeans_iter2.cluster_centers_)

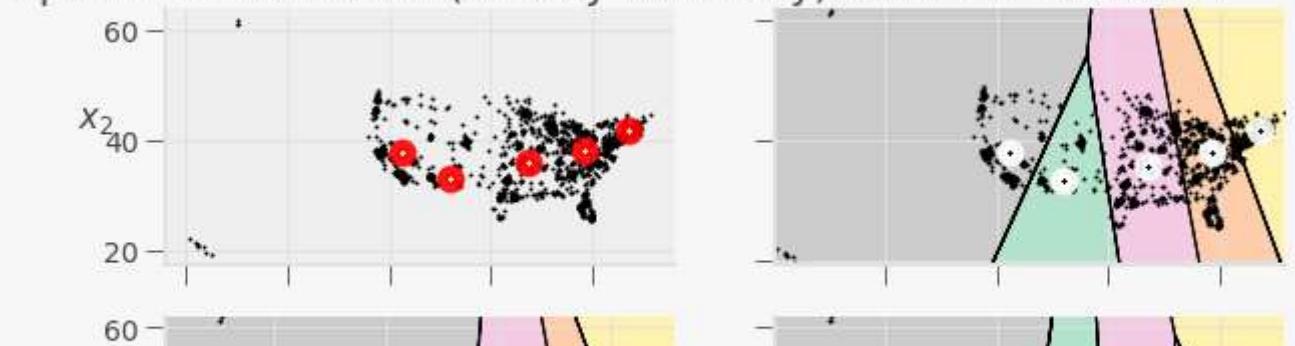
plt.subplot(324)
plot_decision_boundaries(kmeans_iter2, X, show_xlabels=False,
                         show_ylabels=False)

plt.subplot(325)
plot_decision_boundaries(kmeans_iter2, X, show_centroids=False)
plot_centroids(kmeans_iter3.cluster_centers_)

plt.subplot(326)
plot_decision_boundaries(kmeans_iter3, X, show_ylabels=False)

plt.show()
```

Update the centroids (initially randomly) Label the instances



```
from sklearn.metrics import pairwise_distances_argmin_min

closest, _ = pairwise_distances_argmin_min(kmeans_iter1.cluster_centers_, X)
users=df['address'].values
for row in closest:
    print(users[row])
```

150 E Stacy Rd, Allen, TX 75002-8756
 3489 Lowery Pkwy, Fultondale, AL 35068-1677
 1900 CHESTNUT ST, Philadelphia, PA 19103-4610
 449 Howe Ave, Cuyahoga Falls, OH 44221-4943
 25901 Highway 290, Cypress, TX 77429-1099



```
# Predicting the clusters
labels = kmeans_iter1.predict(X)
# Getting the cluster centers
C = kmeans_iter1.cluster_centers_
colores=['Almacen 1','Almacen 2 ','Almacen 3','Almacen 4 ','Almacen 5 ']
asignar=[]
```

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

Tienda	cantidad
--------	----------



```
from sklearn.metrics import pairwise_distances_argmin_min

closest, _ = pairwise_distances_argmin_min(kmeans_iter2.cluster_centers_, X)
users=df['address'].values
for row in closest:
    print(users[row])
```

195 River Rd, Lisbon, CT 06351-3253
 820 Oviedo Mall Blvd, Oviedo, FL 32765-9348
 203 S Broadway, Salem, NH 03079-3311
 10720 SW Village Pkwy, Port Saint Lucie, FL 34987-2188
 8401 S Tamiami Trl, Sarasota, FL 34238-2927

```
# Predicting the clusters
labels = kmeans_iter2.predict(X)
# Getting the cluster centers
C = kmeans_iter2.cluster_centers_
colores=['Almacen 1','Almacen 2 ','Almacen 3','Almacen 4 ','Almacen 5 ']
asignar=[]
```

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

Tienda	cantidad
--------	----------



0	Almacen 1	136
1	Almacen 2	579
2	Almacen 3	432
3	Almacen 4	391
4	Almacen 5	301

```
from sklearn.metrics import pairwise_distances_argmin_min

closest, _ = pairwise_distances_argmin_min(kmeans_iter3.cluster_centers_, X)
users=df['address'].values
```

```

for row in closest:
    print(users[row])

    195 River Rd, Lisbon, CT 06351-3253
    6700 Topanga Canyon Blvd, Canoga Park, CA 91303-2624
    203 S Broadway, Salem, NH 03079-3311
    1664 W Division St, Chicago, IL 60622-3922
    600 Kirkwood Mall, Bismarck, ND 58501-5701

# Predicting the clusters
labels = kmeans_iter3.predict(X)
# Getting the cluster centers
C = kmeans_iter3.cluster_centers_
colores=['Almacen 1','Almacen 2 ','Almacen 3','Almacen 4 ','Almacen 5 ']
asignar=[]

```

```

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

```

	Tienda	cantidad	
0	Almacen 1	140	
1	Almacen 2	561	
2	Almacen 3	428	
3	Almacen 4	414	
4	Almacen 5	296	

```

def plot_clusterer_comparison(clusterer1, clusterer2, X, title1=None,
                               title2=None):
    clusterer1.fit(X)
    clusterer2.fit(X)

    plt.figure(figsize=(10, 3.2))

    plt.subplot(121)
    plot_decision_boundaries(clusterer1, X)
    if title1:
        plt.title(title1)

    plt.subplot(122)
    plot_decision_boundaries(clusterer2, X, show_ylabels=False)

```

```

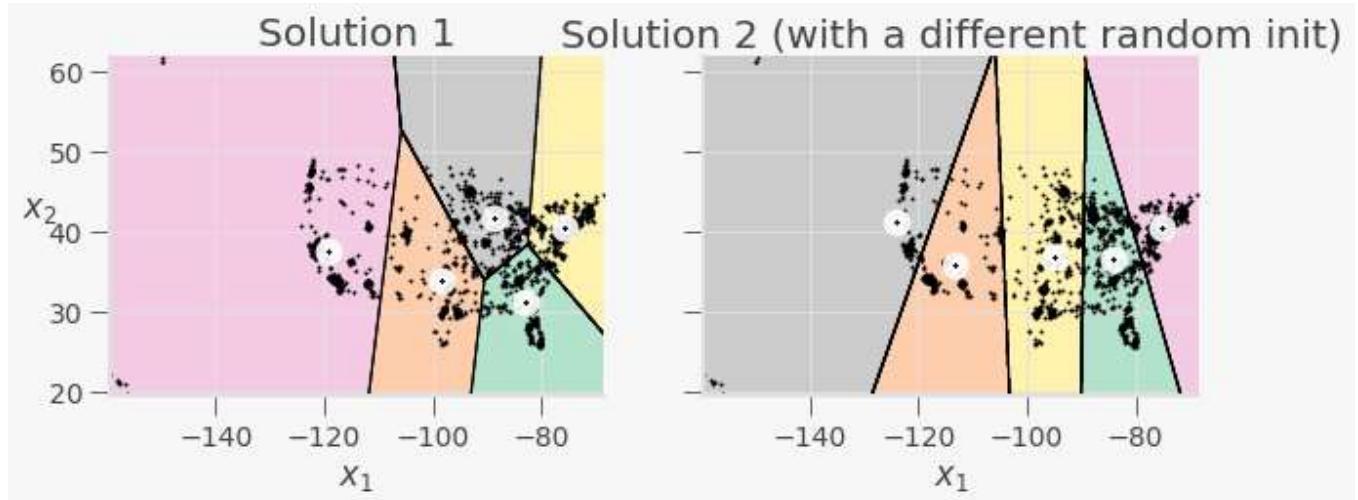
if title2:
    plt.title(title2)

kmeans_rnd_init1 = KMeans(n_clusters=5, init="random", n_init=1, random_state=2)
kmeans_rnd_init2 = KMeans(n_clusters=5, init="random", n_init=1, random_state=9)

plot_clusterer_comparison(kmeans_rnd_init1, kmeans_rnd_init2, X,
                           "Solution 1",
                           "Solution 2 (with a different random init)")

plt.show()

```



```

from sklearn.metrics import pairwise_distances_argmin_min

closest, _ = pairwise_distances_argmin_min(kmeans_rnd_init1.cluster_centers_, X)
users = df['address'].values
for row in closest:
    print(users[row])

4701 N Illinois St, Fairview Heights, IL 62208-3416
1280 Auto Park Way, Escondido, CA 92029-2231
8800 Whittier Blvd, Pico Rivera, CA 90660-2658
660 S Grand Ave, Sun Prairie, WI 53590-9832
7003 S Broadway Ave, Tyler, TX 75703-4737

```

```
mappita(kmeans_rnd_init1.cluster_centers_)
```



```
# Predicting the clusters
labels = kmeans_rnd_init1.predict(X)
# Getting the cluster centers
C = kmeans_rnd_init1.cluster_centers_
colores = ['Almacen 1', 'Almacen 2 ', 'Almacen 3', 'Almacen 4 ', 'Almacen 5 ']
asignar = []

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

	Tienda	cantidad	
0	Almacen 1	274	
1	Almacen 2	270	
2	Almacen 3	368	
3	Almacen 4	484	
4	Almacen 5	443	

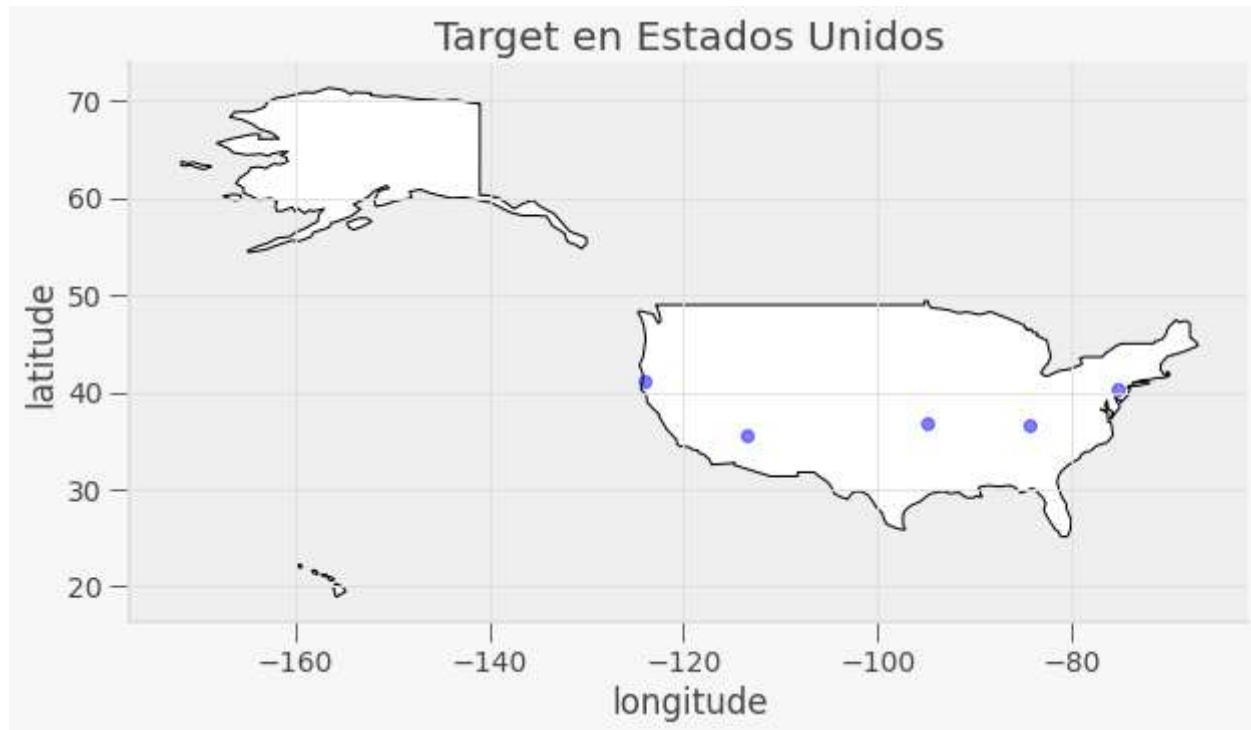
```
from sklearn.metrics import pairwise_distances_argmin_min

closest, _ = pairwise_distances_argmin_min(kmeans_rnd_init2.cluster_centers_, X)
users = df['address'].values
for row in closest:
    print(users[row])

4300 24th Ave, Fort Gratiot, MI 48059-3806
60 Longview Dr, Bangor, ME 04401-3629
2200 Dallas Pkwy, Plano, TX 75093-4300
```

```
1465 E Mall Dr, Holland, OH 43528-9490
```

```
mappita(kmeans_rnd_init2.cluster_centers_)
```



```
# Predicting the clusters
labels = kmeans_rnd_init2.predict(X)
# Getting the cluster centers
C = kmeans_rnd_init2.cluster_centers_
colores = ['Almacen 1', 'Almacen 2 ', 'Almacen 3', 'Almacen 4 ', 'Almacen 5 ']
asignar = []

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

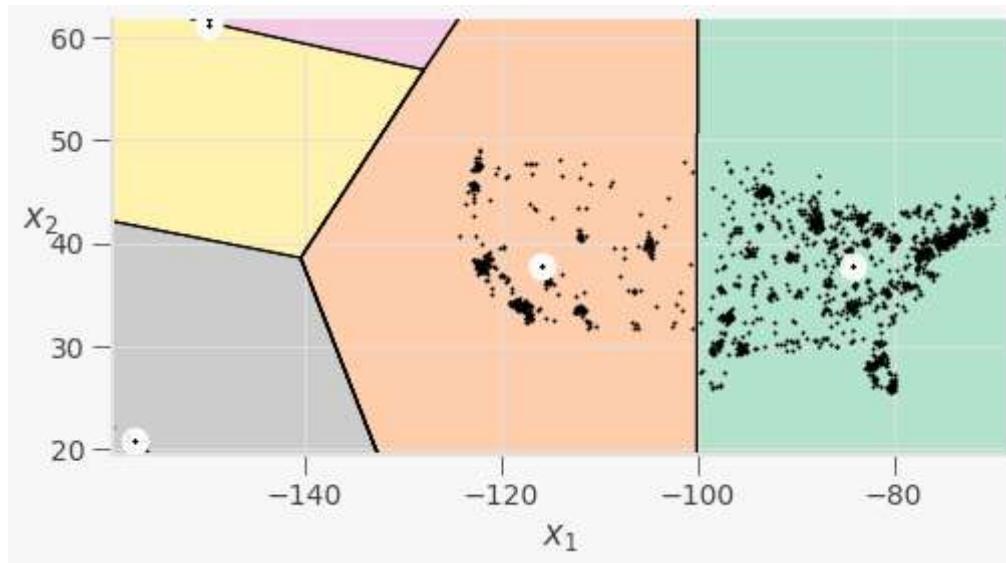
```

good_init = np.array([[-3, 3], [-3, 2], [-3, 1], [-1, 2], [0, 2]])
kmeans = KMeans(n_clusters=5, init=good_init, n_init=1, random_state=42)
kmeans.fit(X)

KMeans(init=array([[-3, 3],
                  [-3, 2],
                  [-3, 1],
                  [-1, 2],
                  [0, 2]]),
      n_clusters=5, n_init=1, random_state=42)

# extra code
plt.figure(figsize=(8, 4))
plot_decision_boundaries(kmeans, X)

```



```
kmeans.inertia_
```

```
150762.9945866068
```

```
kmeans_rnd_init1.inertia_ # extra code
```

```
62084.34548589636
```

```
kmeans_rnd_init2.inertia_ # extra code
```

```
73145.48229260262
```

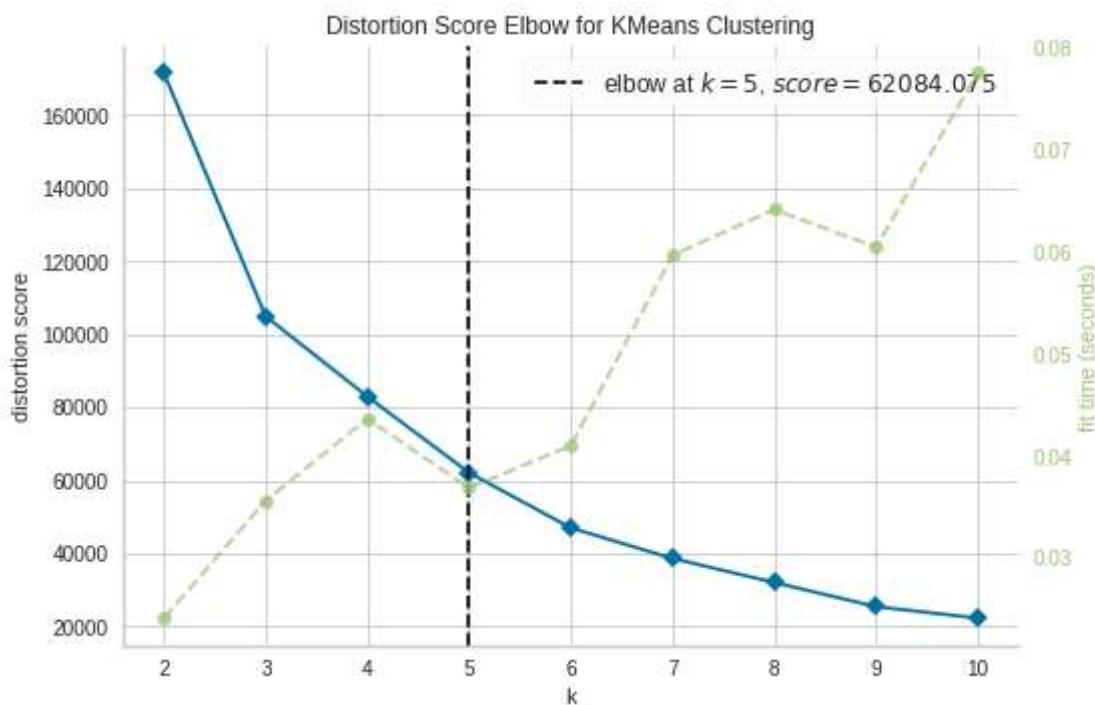
```
X_dist = kmeans.transform(X)
(X_dist[np.arange(len(X_dist)), kmeans.labels_] ** 2).sum()
```

```
150762.99458660622
```

```
kmeans.score(X)
```

```
-150762.9945866068
```

```
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
model = KElbowVisualizer(KMeans(), k=10)
model.fit(X)
model.show()
```



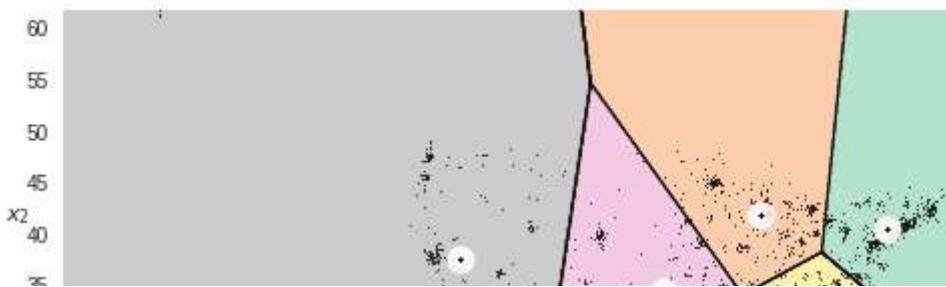
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f95c301abd0>
```

```
kmeans_rnd_10_inits = KMeans(n_clusters=5, init="random", n_init=50,
                               random_state=10)
```

```
kmeans_rnd_10_inits.fit(X)
```

```
KMeans(init='random', n_clusters=5, n_init=50, random_state=10)
```

```
plt.figure(figsize=(8, 4))
plot_decision_boundaries(kmeans_rnd_10_inits, X)
plt.show()
```



```

name = df["address"].str.split(',',expand=True)
name
name.columns = ['direccion', 'Ciudad','Estado','nada']
name
ciudad=name["Estado"].str.split(expand=True)
ciudad=ciudad[0]

from sklearn.metrics import pairwise_distances_argmin_min

closest, _ = pairwise_distances_argmin_min(kmeans_rnd_10_inits.cluster_centers_, X)

users = ciudad.values
ciudades= list()
for row in closest:
    ciudades.append(users[row])

print(ciudades)

['WI', 'CA', 'TX', 'IL', 'CA']

def unique(list1):

    # initialize a null list
    unique_list = []

    # traverse for all elements
    for x in list1:
        # check if exists in unique_list or not
        if x not in unique_list:
            unique_list.append(x)
    # print list
    for x in unique_list:
        print (x)

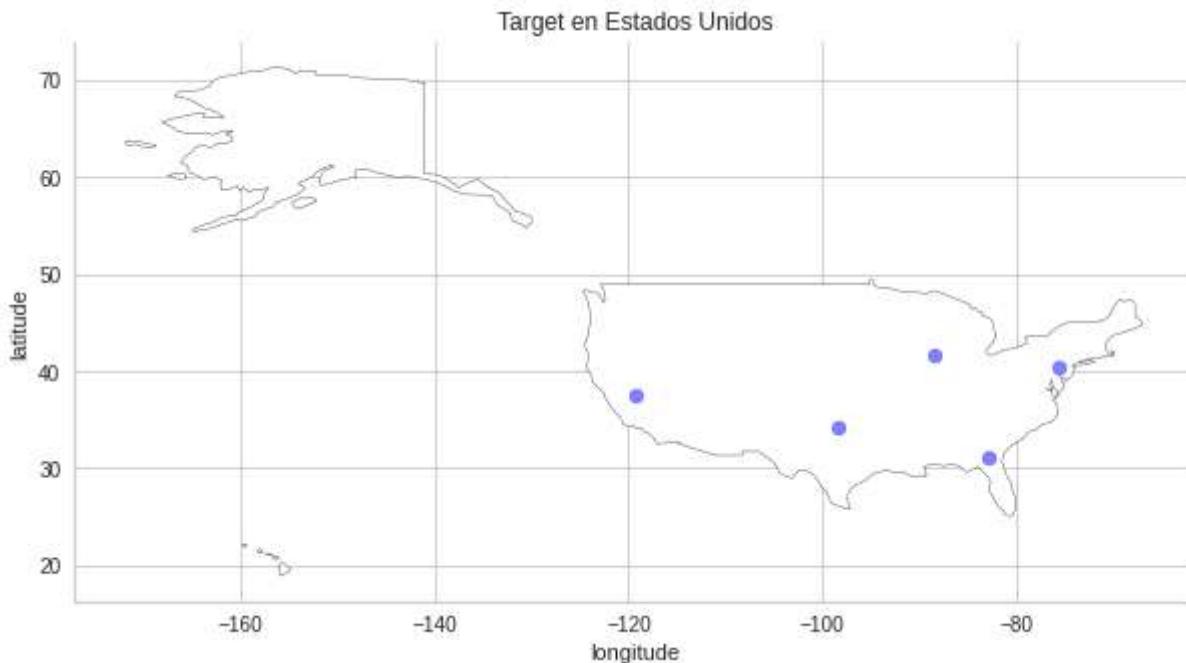
dfcities=pd.DataFrame(ciudades)

```

```
unique(ciudades)
```

```
WI  
CA  
TX  
IL
```

```
mappita(kmeans_rnd_10_inits.cluster_centers_)
```



▼ 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 mejores valores se encontro que las ciudades donde conviene poner un almacén 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 almacén a cada tienda

```
# Predicting the clusters
labels = kmeans_rnd_10_inits.predict(X)
# Getting the cluster centers
C = kmeans_rnd_10_inits.cluster_centers_
colores = dfcitis
asignar = []
print('Tabla 1 Ciudad y Cuntas Tiendas Cubre')
copy = pd.DataFrame()
copy['usuario'] = df['latitude'].values
copy['categoria'] = df['longitude'].values
copy['label'] = labels;
```

```
cantidadGrupo = pd.DataFrame()
cantidadGrupo['Tienda'] = colores
cantidadGrupo['cantidad'] = copy.groupby('label').size()
cantidadGrupo
```

Tabla 1 Ciudad y Cuantas Tiendas Cubre

Tienda cantidad 

	Tienda	cantidad
0	WI	481
1	CA	429
2	TX	289
3	IL	272
4	CA	368

```
print('Tabla 2 Que tan lejos esta de cada punto cada tienda')
Distancia=pd.DataFrame(kmeans.transform(X).round(2))
Distancia=Distancia.set_axis(dfcitis, axis=1)
Distancia=Distancia.set_axis(ciudad, axis=0)
Distancia
```

Tabla 2 Que tan lejos esta de cada punto cada tienda

(WI,) (CA,) (TX,) (IL,) (CA,) 

	0	1	2	3	4
AL	6.33	34.90	74.35	74.24	76.63
AL	2.95	29.41	68.16	68.06	72.33
AL	39.74	10.70	30.97	30.79	42.22
AL	10.88	42.20	78.63	78.60	85.86
AL	12.34	37.74	78.19	78.05	77.45
...
WI	33.79	4.74	42.63	42.36	41.52
WI	4.77	30.99	67.24	67.20	75.19
WY	39.36	10.42	31.31	31.13	42.50
WY	13.10	44.37	80.50	80.48	88.08
WY	10.97	42.31	78.76	78.73	85.96

1839 rows × 5 columns

```
kmeans_rnd_10_inits.inertia_
```

```
62066.14730281925
```

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

Inicialmente se decido analizar la opcion de tener el mismo numero de almacenes que una 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 encontro 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 de puerto rico, para este punto es interesante poder visualizar otras grandes empresas como amazon y alli express que tienen casi la misma distribucion de almacenes que lo calculado y de la misma forma dejando de lado Alaska y puerto rico , ya que Alaska es un mercado temporal y puerto rico recibe los pedidos desde los estados unidos de los almacenes principales

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

Existen otras librerias como: 1. ArcPy 2. Cartopy 3. Folium 4. GDAL/OGR 5. Geemap 6. GeoPandas 7. GeoPy 8. LiDAR

Todas estas librerias son enfocadas en datos gis, los cuales son conocidos como datos geograficos, su importancia dentro del medio es muy necesaria ya que nos ayudan a entender mejor los datos podriamos pensar que cierto tipo de datos son atipicos o estan mal posicionados sin embargo como en este caso no es que fueran atipicos si no que estaban en un lugar diferente geograficamente

¿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 y en este caso nos ayuda a entender donde poner el almacen y ubicarlo de forma facil dentro del sistema

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 donde tenemos una gran cantidad de datos y queremos tomar una muestra

Productos de pago de Colab - Cancelar contratos

✓ 0 s completado a las 23:00

