Presentado por:

Giovanni Acuña Morales ***A01794007***

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-/maps.ipynb#scrollTo=uo2oPtSCeAOz

Encuentra el numero ideal de almacenes, justifica tu respuesta:

Encuentra las latitudes y longitudes de los almacenes, ¿qué ciudad es?, ¿a cuantas tiendas va surtir?, ¿sabes a que distancia estará? ¿Cómo elegiste el número de almacenes?, justifica tu respuesta técnicamente. Adicionalmente, en el notebook notaras que al inicio exploramos los datos y los graficamos de manera simple, después nos auxiliamos de una librería de datos geográficos.

¿qué librerías nos pueden ayudar a graficar este tipo de datos? ¿Consideras importante que se grafique en un mapa?, ¿por qué? Agrega las conclusiones

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

```
Requirement already satisfied: folium in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pyLDAvis in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: descartes in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: pyarrow in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pandas-datareader in /usr/local/lib/python3.7/d
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: quandl in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: quantecon in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: openpyxl in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: munch in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.7/
Requirement already satisfied: pyproj>=2.2.0 in /usr/local/lib/python3.7/dist-
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
Requirement already satisfied: sklearn in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: funcy in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-package
                               d: numexpr in /usr/local/lib/python3.7/dist-packag
                               d: cycler>=0.10 in /usr/local/lib/python3.7/dist-p
                               d: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /us
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: 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: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/d
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
Requirement already satisfied: inflection>=0.3.1 in /usr/local/lib/python3.7/d
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: sympy in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/py
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
```

```
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.sample(10)
```

Se guardó correctamente

	name	latitude	longitude	address	phone
				2490 N Fairfield Rd	937

```
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.

```
latlong.plot.scatter( "longitude","latitude")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5976d0e350>
```


```
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
import qeds
qeds.themes.mpl_style();
```

```
df["Coordinates"] = list(zip(df.longitude, df.latitude))
df["Coordinates"] = df["Coordinates"].apply(Point)
df.sample(10)
```

	name	latitude	longitude	address	phone	
701	Jonesboro	35.823099	-90.664557	3000 E Highland Dr, Jonesboro, AR 72401-6321	870-934-9661	https://www.target.com/sl/jonesb

```
gdf = gpd.GeoDataFrame(df, geometry="Coordinates")
gdf.sample(10)
```

	name	latitude	longitude	address	phone	we
1147	Huntington	40.827611	-73.407155	124 E Jericho Tpke, Huntington Station, NY 117...	631-760-3271	https://www.target.com/sl/huntingto
608	Wood Dale	41.967564	-87.992684	401 W Irving Park Rd, Wood Dale, IL 60191-1338	630-594-5510	https://www.target.com/sl/wood-d
1183	Westbury	40.741667	-73.600818	999 Corporate Dr, Westbury, NY 11590-	516-222-1003	https://www.target.com/sl/westbu

```
world = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))
world = world.set_index("iso_a3")
world.sample(10)
```

	pop_est	continent	name	gdp_md_est	geometry	iso_a3

```
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)
```

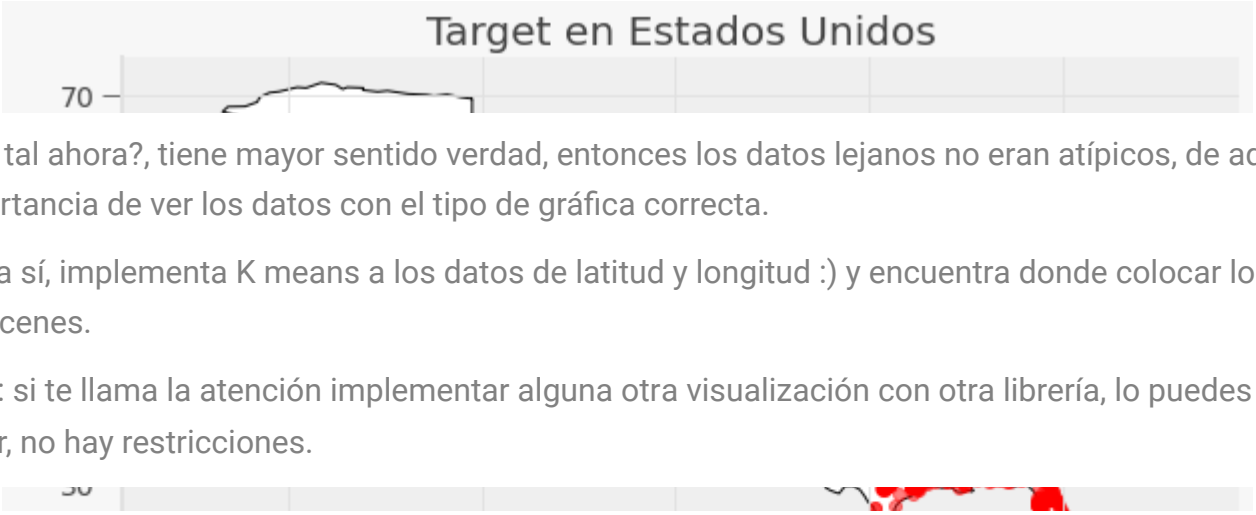
```
fig, gax = plt.subplots(figsize=(10,10))
world.query("name == 'United States of America'").plot(ax=gax, edgecolor='black',colo
gax.set_xlabel('longitude')
gax.set_ylabel('latitude')
gax.spines['top'].set_visible(False)
gax.spines['right'].set_visible(False)
```

Encuentra las latitudes y longitudes de los almacenes

longitude

```
fig, gax = plt.subplots(figsize=(10,10))
world.query("name == 'United States of America'").plot(ax = gax, edgecolor='black', c
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()
```

Se guardó correctamente



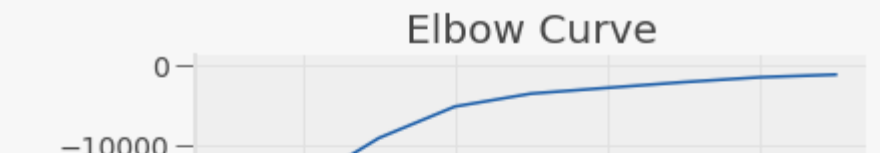
¿qué tal ahora?, tiene mayor sentido verdad, entonces los datos lejanos no eran atípicos, de aquí la importancia de ver los datos con el tipo de gráfica correcta.

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

#Encuentra las latitudes y longitudes de los almacenes, --------------------RESPONDID
# ¿qué ciudad es?,
# ¿a cuantas tiendas va surtir?,
# ¿sabes a que distancia estará?
# ¿Cómo elegiste el número de almacenes?, ----------------------------------RESPONDI
# justifica tu respuesta técnicamente.
# Adicionalmente, en el notebook notaras que al inicio exploramos los datos y los gra
#pueden ayudar a graficar este tipo de datos? ¿Consideras importante que se grafique


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()
```

Se guardó correctamente



PCA es para reducir dimensiones, la variabilidad de tus datos, para trabajar con menos variables de entrada y KMeans es para hacer clusteres, aqui solo los agrupas y la variabilidad no nos importa

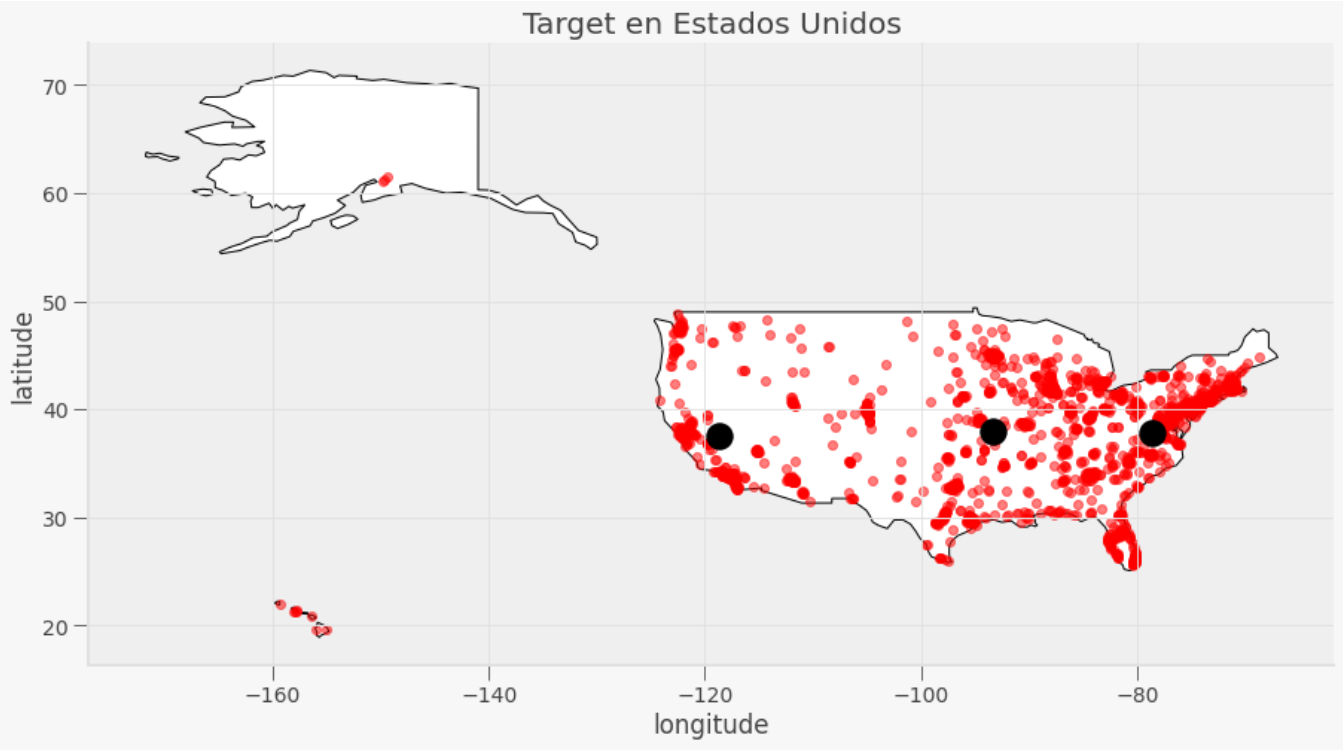
```
kmeans = KMeans(n_clusters = 3, init ='k-means++')
kmeans.fit(latlong[latlong.columns[0:2]])
labels = kmeans.labels_
labels
```

```
array([0, 0, 0, ..., 1, 0, 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	-78.534390	37.782609	POINT (-78.53439 37.78261)
1	-118.624473	37.487342	POINT (-118.62447 37.48734)
2	-93.279950	37.987914	POINT (-93.27995 37.98791)

```
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 en Estados Unidos')
                                        se)
                                        alse)
plt.show()
```



¿A cuantas tiendas va surtir?

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

```
/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	1	Coordinates
0	-78.534390	37.782609	POINT (-78.53439 37.78261)
1	-118.624473	37.487342	POINT (-118.62447 37.48734)
2	-93.279950	37.987914	POINT (-93.27995 37.98791)

Se guardó correctamente

```
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)
```

```
37.78260864094776, -78.53438980340219
37.48734203064935, -118.62447331844157
37.98791363565769, -93.27994961093502
```

Con esta función obtenemos la longitud y latitud de las 3 tiendas que cumple con la mejor geolocalización.

¿Sabes a que distancia estará?

```
from geopy.geocoders.yandex import Location
from geopy.geocoders import Nominatim
from geopy.distance import geodesic

geolocator = Nominatim(user_agent="my-application")
Locations = [Location1, Location2, Location3]

for i in Locations:
  location = geolocator.reverse(i)
  print(f'La tienda :{location.address}\n')
```

```
La tienda :James River Road, Scottsville, Albemarle County, Virginia, 24590, Uni

La tienda :Paradise Estates, Mono County, California, United States

La tienda :State Highway U, Hickory County, Missouri, 65668, United States
```

Obtenemos las distancias entre los 3 almacenes

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

Se guardó correctamente

```
Distancia entre el primer y segundo almacén :  2181.65568564248  ft2

Distancia entre el segundo y tercer almacén :  1384.2605690747557  ml

1879.0086206896553
1190.8793103448277
```