Proyecto Capstone - La Batalla de los Vecindarios (Semana 2)

R. Salaberry

11 de julio de 2021

Resumen

Como parte del curso IBM Data Science professional program Capstone Project, trabajaremos con un conjunto de datos reales con el fin de llevar a cabo un proyecto que abarque todas las instancias en las que un científico de datos se encontraría en la vida real al resolver un problema dado.

El principal objetivo de este proyecto es utilizar todas las herramientas vistas en los cursos previos para la resolución de un problema de negocios dado, tal como lo haría en la vida real un científico de datos.

Se aplicarán los conocimientos adquiridos en la búsqueda de datos, la preparación de los mismos, el análisis y las conclusiones que implican para el caso de negocios planteado. El caso de negocio será la apertura de un restaurante de comida sudamericana (cono sur, Argentina, Uruguay, Brasil, Chile y Paraguay) muy populares en la región y que se conocen con el nombre de parrillas o parilladas en la ciudad de Toronto, Canadá.

Índice

1. Definición del problema **2**
- 1.1. Planteo del problema 2
- 1.2. Perfil de interesados 3

2. Fuente de datos. Adquisición y transformación **3**

3. Metodología **4**

4. Desarrollo del trabajo **4**
- 4.1. Primera parte, los barrios y sus coordenadas 5
- 4.2. Segunda parte, datos demográficos de residentes sudamericanos . 8
- 4.3. Tercera parte. Análisis primario de los datos demográficos . . . 10
- 4.4. Cuarta parte. Obtención de datos de restaurantes sudamericanos 13

5. Resultados **17**

6. Conclusiones **17**

1. Definición del problema

1.1. Planteo del problema

Toronto, una ciudad muy cosmopolita, es la capital de la provincia de Ontario en Canadá. La ciudad ha albergado tradicionalmente muchas corrientes migratorias que se han establecido con el tiempo en sus barrios, de modo tal que hoy en día existen entre sus barrios, algunos que hacen referencias a la diversidad de culturas de dicha ciudad, como ser Chinatown, Corso Italia, Greektown, Kensington Market, Koreatown, Little India, Little Italy, Little Jamaica y Little Portugal.

Figura 1: Ciudad de Toronto, Canadá.

En este proyecto, seguiremos paso a paso el proceso para saber si es buena idea la decisión de abrir un restauran de estilo sudamericano en la ciudad de Toronto, cuales son los barrios mas indicados para ello, teniendo en cuenta la migración de origen sudamericano y también los ingresos para cada barrio a fin de crear para el restauran una clientela con un perfil económico aceptable.

Tomamos como definición de restaurante sudamericano (parrilla o parrillada) aquel en donde se cocina la carne de res a las brasas de la leña en una parilla. La comida típica así definida se denomina asado.

Los países definidos como sudamericanos son los países pertenecientes al denominado cono sur, es decir Argentina, Uruguay, Brasil, Chile y Paraguay. Estos serán la población objetivo a estudiar y relacionar con los barrios de Toronto.

Para ello nos planteamos las siguientes preguntas:

- ¿En qué barrios de Toronto vive la comunidad sudamericana del cono sur?
- ¿Hay restaurantes de estilo sudamericano en Toronto, donde están ubicados?
- ¿Cómo es la distribución de ingresos de cada barrio de Toronto?

Figura 2: Parrillada estilo sudamericano

- ¿Cuál es el barrio de Toronto mas apropiado para abrir un restaurante de estilo sudamericano (parrillada) ?

1.2. Perfil de interesados

El perfil de personas interesadas en este problema son las personas relacionadas a la comunidad sudamericana de Toronto, y los empresarios sudamericanos que residiendo o no en Toronto deseen invertir en un restaurante típico en la ciudad, así como también cualquier empresario canadiense que quiera invertir en el rubro de gastronomía.

También representa interés a la comunidad de científicos analistas de datos que deseen analizar los barrios de Toronto para un eventual modelo de negocio del rubro gastronómico.

2. Fuente de datos. Adquisición y transformación

Para este planteo vamos a precisar un detalle de los barrios de Toronto, que se puede obtener en Wikipedia desde el siguiente link: Datos código postal, barrios de Toronto. Tambien existe otra pagina con dicha información, es de Zipcodesonline del año 2020 y tiene los zip codes de muchas ciudades del mundo, entre ellas los de Toronto en este link: Zip code de Toronto

También el siguiente link de Coordenadas de códigos postales de Toronto, Canadá que proporciona la latitud y longitud de cada uno de los códigos postales que aparecen en la página de wikipedia anterior.

Finalmente vamos a necesitar información demográfica de los barrios de Toronto, esta información la obtenemos del portal de datos de la ciudad de Toronto, Datos de perfiles de barrios de Toronto, Canadá.

Para obtener la información sobre restaurantes en cada barrio de Toronto, usaremos la API de Foursquare. Usando la API obtendremos los nombres,

categorías, latitud y longitud de cada restauran existente en cada barrio de Toronto. En esta búsqueda se tratará de ubicar los restaurantes que cumplen con la definición de parrillas o parrilladas de estilo sudamericano.

3. Metodología

Se trabaja con un Notebook Jupyter, realizando el código Python con un kernel Python 3.0. Los pasos a seguir serán los siguientes:

- Pasar a un data frame los datos de los barrios de Toronto de la página de wikipedia.
- Relacionar el data frame con las coordenadas latitud y longitud de archivo de coordenadas geográficas trabajado anteriormente.
- Obtener la información demográfica relevante de los barrios de Toronto y de la población emigrante de Sudamérica desde la base de datos del portal de datos de la ciudad de Toronto.
- Combinar los datos demográficos con la base anterior.
- Realizar el análisis de los datos para saber donde se ubica la población objetivo.
- Obtener datos de restaurantes relacionados (parrilladas) en los barrios de Toronto mediante consultas con la API de Foursquare.
- Relacionar la ubicación de dichos restaurantes con la ubicación de la población objetivo.
- Analizar los resultados obtenidos y realizar las recomendaciones necesarias.

4. Desarrollo del trabajo

De acuerdo al portal de datos de la ciudad de Toronto, existen formalmente en la ciudad 140 barrios,tal como se explica en dicho portal. Por lo tanto a efectos de este trabajo, tomaremos como base los 140 barrios detallados en el archivo neighbourhood-profiles-2016-csv que se usa en este trabajo para tomar los datos de migración como se explicó en la sección 2.

También en dicho portal se listan los 140 barrios reconocidos de Toronto que coinciden con los explicados en el párrafo anterior. Por tal motivo, en este trabajo tomamos como barrios de Toronto los 140 que se encuentran en la pagina oficial de datos de Toronto.

Primero se cargan los paquetes y librerías necesarios.

```
#Importo las librerías paquetes que necesito. Imported the needed library\n",
import re
import numpy as np # librería para manejar datos vectorizados

import pandas as pd # librería para análisis de datos
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

```
import json # librería para manejar archivos JSON

#!conda install -c conda-forge geopy --yes # retirar el comentario de esta línea si no ha
↪  completado el laboratorio de la API de FourSquare
!pip install geopy
from geopy.geocoders import Nominatim # convertir una dirección en valores de latitud y
↪  longitud

import requests # librería para manejar solicitudes
from pandas.io.json import json_normalize # librería para convertir un archivo json en un
↪  dataframe pandas

# Matplotlib y módulos asociados para graficar
import matplotlib.cm as cm
import matplotlib.colors as colors

# importar k-means desde la fase de agrupación
from sklearn.cluster import KMeans

#!conda install -c conda-forge folium=0.5.0 --yes # retirar el comentario de esta línea si
↪  no ha completado el laboratorio de la API de FourSquare
! pip install folium==0.5.0
import folium # librería para graficar mapas

# se carga beautiful soup para el trabajo con HTML
! pip install beautifulsoup4
from bs4 import BeautifulSoup

print('Libraries imported.')
```

4.1. Primera parte, los barrios y sus coordenadas

Luego de cargar las librería a utilizar, se cargan las tablas en donde esta la información de barrios de Toronto, el postal code y los 140 barrios formales de con el objetivo de obtener un data frame que incluya los 140 barrios con sus respectivas coordenadas de latitud y longitud.

El primer problema surge de ver inconsistencias entre los barrios de la página de wikipedia, los barrios de la página de zipcodesonline y los 140 barrios formales del portal de datos de Toronto, en el cual también se incluye la información demográfica de dichos 140 barrios, por lo tanto, hay que buscar la forma de obtener las coordenadas de los 140 barrios por otro camino.

Se descarta entonces el uso de la información de las paginas de wikipedia y zipcodesonline para buscar las coordenadas a traves del servicio de "Geolocator" usado anteriormente para ubicar coordenadas de direcciones.

Carga de los barrios de la página del portal de datos de la ciudad de Toronto.

```
with open('Neighbourhood Profiles - City of Toronto.html', encoding="utf8") as fp3:
soup = BeautifulSoup(fp3, "html.parser")
table3 = soup.find_all('table')
df3 = pd.read_html(str(table3))[1]
col3 = ['Hood #','Neighborhood']
df3.columns=col3
df3 = df3.reset_index(drop=True)
df3.sort_values(by='Neighborhood')
df3.head()
```

Obtenemos el data frame con los 140 barrios. Luego buscamos las coordenadas con el servicio "Geolocator".

```
barrio=[]
lat=[]
long=[]
barrio_num=[]
```

```
for i in range(len(df3)):
try:
address  = df3.loc[i,'Neighborhood']+','+'Toronto'
geolocator = Nominatim(user_agent="Tor_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
barrio.append(df3.loc[i,'Neighborhood'])
lat.append(latitude)
long.append(longitude)
barrio_num.append(df3.loc[i,'Hood #'])
except:
barrio_num.append(df3.loc[i,'Hood #'])
barrio.append(df3.loc[i,'Neighborhood'])
lat.append(None)
long.append(None)
col32=['Hood #','Neighborhood', 'Latitude', 'Longitude']
df32=pd.DataFrame(columns=col32)
df32['Hood #']=barrio_num
df32['Neighborhood']=barrio
df32['Latitude']=lat
df32['Longitude']=long
```

Se recuperan 108 coordenadas de 140 barrios.

Out[201]:

	Hood #	Neighborhood	Latitude	Longitude
0	1	West Humber-Clairville	43.722563	-79.597039
1	2	Mount Olive-Silverstone-Jamestown	NaN	NaN
2	3	Thistletown-Beaumond Heights	NaN	NaN
3	4	Rexdale-Kipling	43.722114	-79.572292
4	5	Elms-Old Rexdale	43.721770	-79.552173

Figura 3: Data frame con coordenadas

Paso los registros nulos a otro data frame para analizarlos separadamente. Hay 32 valores Nan correspondientes a barrios de los cuales no se encontraron las coordenadas. Formo otro Data Frame con los barrios que faltan completar y los estudio aparte.

Viendo los vecindarios que quedaron como NaN, la gran mayoría son de dos palabras separadas por "-". Luego hay una de una sola palabra (*"Humbermede"* antiguamente *"Emery"*), dos que tienen parentesis (*"Taylor-Massey (formerly Crescent Town)"* y *"Mimico (includes Humber Bay Shores)"* y otros que esta separado por espacios (*"Lambton Baby Point"*, *"Woodbine Corridor"* y *"Parkwoods Donalda"*). Un caso particular es *"Thistletown-Beaumond Heights"*, el cual es mas conocido solo como *"Thistletown"* y por lo tanto se cambia a ese nombre para poder obtener sus coordenadas.

Separo los registros con Nan para estudiarlos separadamente, luego separo los nombres de los barrios para usar su segunda denominación y poder encontrar las coordenadas. También arreglo los casos particulares presentados anteriormente. Luego utilizo de nuevo el servicio "Geolocator" para obtener las coordenadas de los restantes barrios. El resultado es un data frame con 32 filas de los barrios que antes eran Nan y ahora tienen su respectivas coordenadas.

```
barrio2=[]
for f in range(len(df32nan)):
n=df32nan.iloc[f,1]
nsplit_aux=re.split('[(]',n)
n2=nsplit_aux[0]
nsplit=re.split('-',n2)
```

```
if len(nsplit)>1:
barrio2.append(nsplit[1])
else:
barrio2.append(nsplit[0])
df32nan['Neighborhood2']=barrio2
df32nan
```

```
barrio32=[]
lat32=[]
long32=[]
barrio32_num=[]
for i in range(len(df32nan)):
try:
address  = df32nan.loc[i,'Neighborhood2']+','+'Toronto'
geolocator = Nominatim(user_agent="Tor_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
barrio32_num.append(df32nan.loc[i,'Hood #'])
barrio32.append(df32nan.loc[i,'Neighborhood2'])
lat32.append(latitude)
long32.append(longitude)
except:
barrio32_num.append(df32nan.loc[i,'Hood #'])
barrio32.append(df32nan.loc[i,'Neighborhood2'])
lat32.append(None)
long32.append(None)
df32nan['Hood #']=barrio32_num
df32nan['Neighborhood2']=barrio32
df32nan['Latitude']=lat32
df32nan['Longitude']=long32
df32nan.head()
```

Obtengo el data frame con los 32 registros que faltan de los barrios y sus coordenadas. Ahora hay que concatenarlo al otro data frame y obtener así el data frame buscado de los 140 barrios y sus coordenadas.

Out[206]:

	Hood #	Neighborhood	Latitude	Longitude	Neighborhood2
0	2	Mount Olive-Silverstone-Jamestown	43.749751	-79.599116	Silverstone
1	3	Thistletown-Beaumond Heights	43.737266	-79.565317	Thistletown
2	6	Kingsview Village-The Westway	43.683437	-79.566514	The Westway
3	7	Willowridge-Martingrove-Richview	43.673954	-79.560480	Martingrove
4	10	Princess-Rosethorn	43.658474	-79.539097	Rosethorn

Figura 4: Data frame con coordenadas restantes.

Finalmente elimino la columna de segunda denominación , elimino los Nan del primer data frame y concateno los dos obteniendo el resultado buscado.

```
df32nan=df32nan.drop(['Neighborhood2'], axis=1)
df32=df32.dropna()
df32 = pd.concat([df32, df32nan])
df32 = df32.reset_index(drop=True)
print(df32.shape)
df32.head()
```

Obtengo el siguiente data frame:

Finalmente creamos un mapa para ver los resultados de los 140 barrios de Toronto.

```
# crear un mapa de Toronto utilizando los valores de latitud y longitud
map_toronto = folium.Map(location=[latitude, longitude], zoom_start=10)
# añadir marcadores al mapa
```

Out[214]:

	Hood #	Neighborhood	Latitude	Longitude
0	1	West Humber-Clairville	43.722563	-79.597039
108	2	Mount Olive-Silverstone-Jamestown	43.749751	-79.599116
109	3	Thistletown-Beaumond Heights	43.737266	-79.565317
1	4	Rexdale-Kipling	43.722114	-79.572292
2	5	Elms-Old Rexdale	43.721770	-79.552173

Figura 5: Data frame final con los 140 barrios y sus coordenadas.

```
for lat, lng, neighborhood, hood in zip(df32['Latitude'], df32['Longitude'],
↪  df32['Neighborhood'],df32['Hood #'] ):
label = '{}, hood n° {}'.format(neighborhood, hood)
label = folium.Popup(label, parse_html=True)
folium.CircleMarker(
[lat, lng],
radius=5,
popup=label,
color='blue',
fill=True,
fill_color='#3186cc',
fill_opacity=0.7,
parse_html=False).add_to(map_toronto)

map_toronto
```

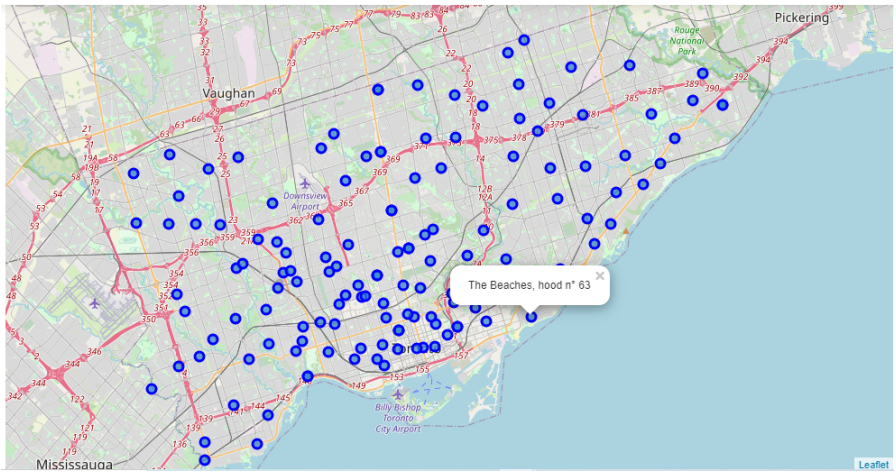

Figura 6: Barrios de Toronto.

4.2. Segunda parte, datos demográficos de residentes sud-americanos

Comenzamos cargando el archivo csv bajado del portal de datos de la ciudad de Toronto. Despues de un estudio de sus características, buscamos con las palabras claves *Neighbourhood Number* , *Total income: Average amount ($)*, *Brazil*, *Brazilian*, *Argentinian*, *Chilean*, *Paraguayan*, *y Uruguayan* para filtrar los valores que nos interesan.

```
df_demo=pd.read_csv('neighbourhood-profiles-2016-csv.csv',decimal=",")
valores_a_buscar=['Neighbourhood Number',
```

```
'Total income: Average amount ($)',
'Brazil',
'Brazilian',
'Argentinian',
'Chilean',
'Paraguayan',
'Uruguayan']
[True if item in df_demo.values else False for item in valores_a_buscar]
```

Como los datos son todos interpretados como strings y ademas los numeros tiene comas, limpiamos la base para poder analizarla

```
demo2 = df_demo2.reset_index(drop=True)
columns=df_demo2.columns
for i in range(1,len(columns),1):
df_demo2.iloc[8,i] = df_demo2.iloc[8,i].replace(",", "")
val=df_demo2.iloc[8,i]
df_demo2.iloc[8,i] = pd.to_numeric(val, errors='coerce')
df_demo2.iloc[1,i] = pd.to_numeric(df_demo2.iloc[1,i], errors='coerce')
df_demo2.iloc[2,i] = pd.to_numeric(df_demo2.iloc[2,i], errors='coerce')
df_demo2.iloc[1,:]=df_demo2.iloc[1,:]+df_demo2.iloc[2,:]
df_demo2=df_demo2.drop(2, axis=0)
df_demo2 = df_demo2.reset_index(drop=True)
for f in range(len(df_demo2.index)):
for c in range(1,len(columns),1):
df_demo.iloc[f,c]=pd.to_numeric(df_demo.iloc[f,c], downcast='integer',errors='coerce')
df_demo2
```

Luego trasponemos el data frame y hacemos lo necesario para calcular la suma de sudamericanos por barrio y poder trabajarlos como un todo.

```
df_demo22=df_demo2.transpose()
df_demo22 = df_demo22.rename_axis('index').reset_index()
fila0=df_demo22.loc[0,:]
df_demo22.columns=fila0
df_demo22=df_demo22.drop(0,axis=0)
df_demo22 = df_demo22.reset_index(drop=True)
df_demo22 = df_demo22.rename(columns={'Characteristic':'Neighborhood',
'Neighbourhood Number':'Hood #',
'BrazilBrazil':'Brazil'})
df_demo22.head()
```

```
filas=df_demo22.index
colu=df_demo22.columns
len(filas), len(colu)
for f in range(len(filas)):
for c in range(1, len(colu)-2,1):
df_demo22.iloc[f,c]=pd.to_numeric(df_demo22.iloc[f,c], downcast='integer',errors='coerce')
df_demo22['Total migrantes sudamericanos']=df_demo22.iloc[:,-7:-2].sum(axis=1)
df_demo22 = df_demo22.reset_index(drop=True)
```

Obtenemos el siguiente data frame: Finalmente hacemos un merge de ambos

Out[85]:

	Neighborhood	Hood #	Total income: Average amount ($)	Total migrantes sudamericanos	Total Normalizado
0	Agincourt North	129	30414	30.0	0.030457
1	Agincourt South-Malvern West	128	31825	45.0	0.045685
2	Alderwood	20	47709	120.0	0.121827
3	Annex	95	112766	240.0	0.243655
4	Banbury-Don Mills	42	67757	135.0	0.137056

Figura 7: Data frame final de datos demográficos.

data frames y tenemos la información necesaria con la ubicación geográfica para cada barrio. En total quedaron 140 barrios.

```
df_demo=df_demo.merge(df_coor,how='inner',on=['Hood #'])
df_demo.head()
```

Out[42]:

	Neighborhood_x	Hood #	Total income: Average amount ($)	Total migrantes sudamericanos	Total Normalizado	Total Income Norm	Neighborhood_y	Latitude	Longitude
0	Agincourt North	129	30414	30.0	0.030457	0.098744	Agincourt North	43.808038	-79.266439
1	Agincourt South-Malvern West	128	31825	45.0	0.045685	0.103325	Agincourt South-Malvern West	43.781969	-79.257689
2	Alderwood	20	47709	120.0	0.121827	0.154894	Alderwood	43.601717	-79.545232
3	Annex	95	112766	240.0	0.243655	0.366111	Annex	43.670338	-79.407117
4	Banbury-Don Mills	42	67757	135.0	0.137056	0.219983	Banbury-Don Mills	43.752683	-79.365270

Figura 8: Data frame final de datos demográficos y coordenadas.

4.3. Tercera parte. Análisis primario de los datos demográficos

Para visualizar los datos hacemos un mapa que sea proporcional a la cantidad de migrantes sudamericanos en color rojo, ajustando el radio para tal motivo, y en azul proporcional a los ingresos de cada barrio.

```
# crear un mapa de Toronto utilizando los valores de latitud y longitud
map_toronto2 = folium.Map(location=[latitude, longitude], zoom_start=10)
# añadir marcadores al mapa
for lat, lng, neighborhood, hood, radio, pop, radioincome, income in
↪  zip(df_demo['Latitude'], df_demo['Longitude'],
df_demo['Neighborhood_x'],df_demo['Hood #'],
df_demo['Total Normalizado'],df_demo['Total migrantes sudamericanos'],
df_demo['Total Income Norm'],df_demo['Total income: Average amount ($)']):
label1 = '{}, hood n° {}, pop. {} '.format(neighborhood, hood, pop)
label1 = folium.Popup(label1, parse_html=True)
label2 = '{}, hood n° {}, Income. {} '.format(neighborhood, hood, income)
label2 = folium.Popup(label2, parse_html=True)

folium.CircleMarker(
[lat, lng],
radius=1,
#popup=label,
color='red',
fill=True,
fill_color='#3186cc',
fill_opacity=0.7,
parse_html=False).add_to(map_toronto2)
folium.Circle(
radius=radio*500,
popup=label1,
location=[lat,lng],
color='red',
fill=True,
fill_color='red'
).add_to(map_toronto2)
folium.CircleMarker(
[lat, lng],
radius=1,
#popup=label,
color='blue',
fill=True,
fill_color='#3186cc',
fill_opacity=0.7,
parse_html=False).add_to(map_toronto2)
folium.Circle(
radius=radioincome*500,
popup=label2,
location=[lat,lng],
color='blue',
```

```
fill=True,
fill_color='blue'
).add_to(map_toronto2)
```

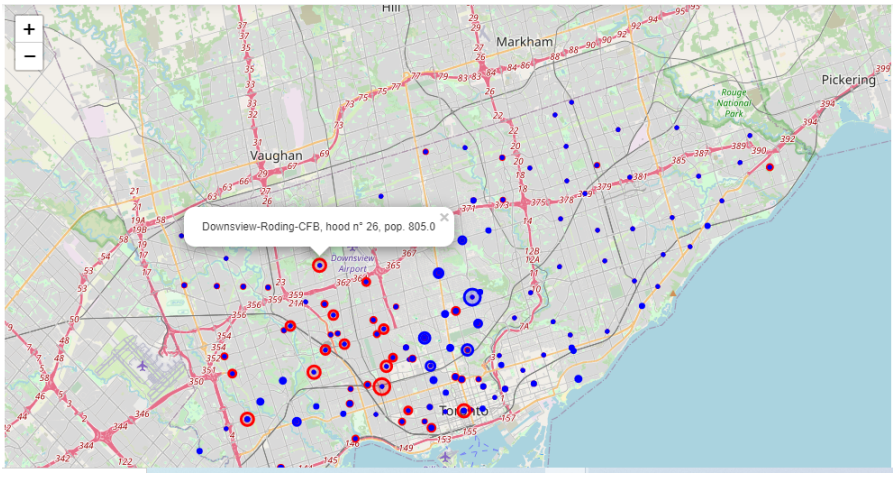

Figura 9: Barrios de Toronto y población sudamericana.

Los círculos rojos mas grandes muestra la mayor concentración de migrantes sudamericanos, y los círculos azules mas grandes la mayor concentración de ingresos.

Para poder hacer una agrupación adecuada, es preciso estudiar cual será el número de klusters ideal, hacemos este estudio usando el método Elbow.

```
df_demo2=df_demo
toronto_grouped_clustering = df_demo2.drop(['Neighborhood_x','Neighborhood_y', 'Hood #',
'Total Normalizado','Total Income Norm'], 1)
from sklearn import metrics
from scipy.spatial.distance import cdist

kmeans_kwargs = {"init": "k-means++","n_init": 10,"max_iter": 300,"random_state": 0,}
# A list holds the SSE values for each k
sse = []
for k in range(1, 11):
kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
kmeans.fit(toronto_grouped_clustering)
sse.append(kmeans.inertia_)
plt.style.use("fivethirtyeight")
plt.plot(range(1, 11), sse)
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
```

La gráfica obtenida muestra que el número ideal de kluster es 3.

```
In [52]: plt.style.use("fivethirtyeight")
         plt.plot(range(1, 11), sse)
         plt.xticks(range(1, 11))
         plt.xlabel("Number of Clusters")
         plt.ylabel("SSE")
         plt.show()
```

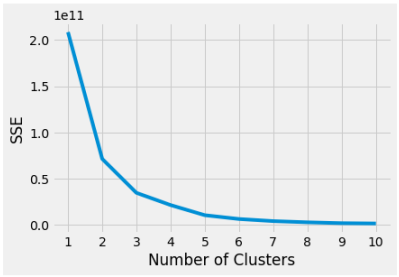

Figura 10: Número ideal de Klusters.

Vamos a probar con 3 klusters para visualizar distinto, incluyendo el ingreso medio por barrio.

```
# establecer el número de agrupaciones
df_demo2=df_demo
kclusters = 3

toronto_grouped_clustering = df_demo2.drop(['Neighborhood_x','Neighborhood_y', 'Hood #',
'Total Normalizado','Total Income Norm'], 1)

# ejecutar k-means
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(toronto_grouped_clustering)

# revisar las etiquetas de las agrupaciones generadas para cada fila del dataframe
kmeans.labels_[0:10]
```

Luego llevamos los klusters al mapa.

```
# crear mapa
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# establecer el esquema de color para las agrupaciones
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# añadir marcadores al mapa
markers_colors = []
for lat, lon, poi, cluster, radio in zip(df_demo2['Latitude'], df_demo2['Longitude'],
df_demo2['Neighborhood_x'], df_demo2['Cluster Labels'], df_demo2['Total Normalizado']):
label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
folium.CircleMarker(
[lat, lon],
radius=10*radio,
popup=label,
color=rainbow[cluster-1],
fill=True,
fill_color=rainbow[cluster-1],
fill_opacity=0.7).add_to(map_clusters)
```

Obtenemos el siguiente mapa:

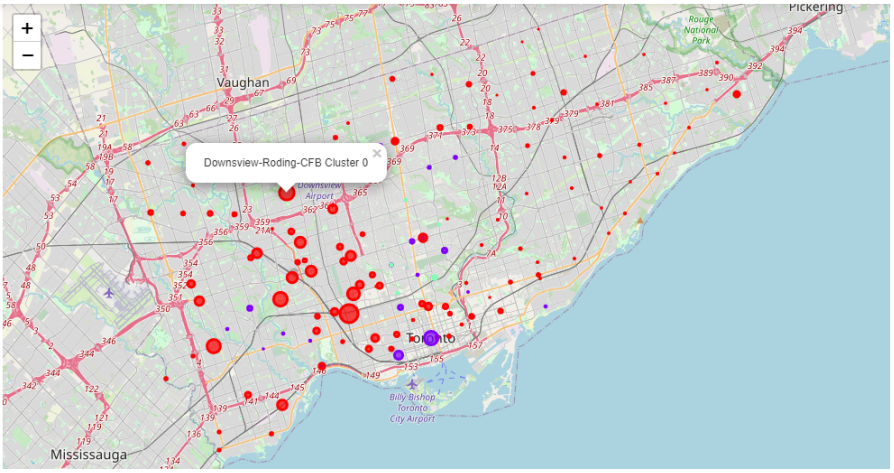

Figura 11: Klusters de barrios por densidad de migrantes sudamericanos.

Miramos la información obtenida en forma de tabla.

```
for i in range(0,3,1):
print('Cluster {}, ingreso medio {}, población sudamericana media {}, cantidad barrios {}
↪  '.format(
df_demo2.loc[df_demo2['Cluster Labels'] == i].mean()[0], df_demo2.loc[df_demo2['Cluster
↪  Labels'] == i].mean()[2],
df_demo2.loc[df_demo2['Cluster Labels'] == i].mean()[3], df_demo2.loc[df_demo2['Cluster
↪  Labels'] == i].count()[1]))
```

Cluster	Ingreso medio	Población sudamericana media	cantidad barrios
0.0	41146.23	206.32	113
1.0	92299.09	165.22	22
2.0	210936.8	113.0	5

Una primera conclusión es que en el kluster donde mayor es el ingreso promedio, es menor la concentración de sudamericano, y donde la concentración es mayor, el ingreso promedio es el mínimo.

4.4. Cuarta parte. Obtención de datos de restaurantes sudamericanos

En esta parte trabajamos con la API de Foursquare para obtener datos sobre negocios de Toronto, pero particularmente de restaurantes de comida típica sudamericana.

Primero fijamos los datos para la API.

```
CLIENT_ID = 'TKDEL1UDPMTSHP2JODTMAMEJAXWSZOT1EHMVGYQHE5Z1XYXG' # su ID de Foursquare
CLIENT_SECRET = 'BZSEZJINU52JACXRQMH5AOPZBZMMV2HFF4XP1F2GOTKOJ32M' # Secreto de Foursquare
VERSION = '20210709' # versión de la API de Foursquare
LIMIT = 100 # Un valor límite para la API de Foursquare

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

Con la función *getNearbyVenues* obtenemos todos los datos de negocios de cada barrio.

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

venues_list=[]
for name, lat, lng in zip(names, latitudes, longitudes):
print(name)

# crear la URL de solicitud de API
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&
v={}&ll={},{}&radius={}&limit={}'.format(
CLIENT_ID,
CLIENT_SECRET,
VERSION,
lat,
lng,
radius,
LIMIT)

# solicitud GET
results = requests.get(url).json()["response"]['groups'][0]['items']

# regresa solo información relevante de cada sitio cercano
venues_list.append([(
name,
lat,
lng,
v['venue']['name'],
v['venue']['location']['lat'],
v['venue']['location']['lng'],
v['venue']['categories'][0]['name']) for v in results])

nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
nearby_venues.columns = ['Neighborhood',
'Neighborhood Latitude',
'Neighborhood Longitude',
'Venue',
'Venue Latitude',
'Venue Longitude',
'Venue Category']

return(nearby_venues)
```

(2796, 7)

Out[40]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Agincourt North	43.808038	-79.266439	Menchie's	43.808338	-79.268288	Frozen Yogurt Shop
1	Agincourt North	43.808038	-79.266439	Saravanaa Bhavan South Indian Restaurant	43.810117	-79.269275	Indian Restaurant
2	Agincourt North	43.808038	-79.266439	Congee Town 太皇名粥	43.809035	-79.267634	Chinese Restaurant
3	Agincourt North	43.808038	-79.266439	Booster Juice	43.809915	-79.269382	Juice Bar
4	Agincourt North	43.808038	-79.266439	Shoppers Drug Mart	43.808894	-79.269854	Pharmacy

Figura 12: Datos de API de Foursquare.

Hago una lista con las categorías de negocios encontradas y busco con el criterio de denominación sudamericana para un restaurante.

```
venues_cat=toronto_venues['Venue Category'].tolist()
search_list = [ 'Argentinian', 'Brazilian', 'Uruguayan', 'Chilean', 'Paraguayan',
'Parrilla', 'Parrillada', 'Asado', 'Asador']
len(search_list), type(search_list)
for i in range(len(venues_cat)):
for j in range(len(search_list)):
if search_list[j] in venues_cat[i]:
print(search_list[j], search_list.index(search_list[j]), venues_cat[i],
↪  venues_cat.index(venues_cat[i]) )
else:
pass
toronto_venues2=toronto_venues[(toronto_venues['Venue Category'] =='Argentinian
↪  Restaurant')
```

```
| (toronto_venues['Venue Category'] =='Brazilian Restaurant')]
toronto_venues2
```

Se encuentra solo dos restaurantes de estilo sudamericano. Finalmente grafico

Out[137]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
471	Corso Italia-Davenport	43.677954	-79.443083	Rio 40 Graus	43.677329	-79.446298	Brazilian Restaurant
2786	Yorkdale-Glen Park	43.703434	-79.452918	Sky Ranch Restaurant	43.700730	-79.451684	Argentinian Restaurant

Figura 13: Restaurantes de estilo sudamericano.

sobre el mapa de klusters para ver la ubicación de los restaurantes.

```
# crear mapa
map_clusters2 = folium.Map(location=[latitude, longitude], zoom_start=11)

# establecer el esquema de color para las agrupaciones
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# añadir marcadores al mapa
markers_colors = []
for lat, lon, poi, cluster, radio in zip(df_demo2['Latitude'], df_demo2['Longitude'],
df_demo2['Neighborhood_x'], df_demo2['Cluster Labels'], df_demo2['Total Normalizado']):
label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
folium.CircleMarker(
[lat, lon],
radius=10*radio,
popup=label,
color=rainbow[cluster-1],
fill=True,
fill_color=rainbow[cluster-1],
fill_opacity=0.7).add_to(map_clusters2)

for venlat, venlon, vencat, venue, nei in zip(toronto_venues2['Venue Latitude'],
↪  toronto_venues2['Venue Longitude'],
toronto_venues2['Venue Category'], toronto_venues2['Venue'],
toronto_venues2['Neighborhood']):
label = folium.Popup(venue + ' ' + vencat , parse_html=True)
folium.Marker([venlat, venlon],popup=label).add_to(map_clusters2)
```

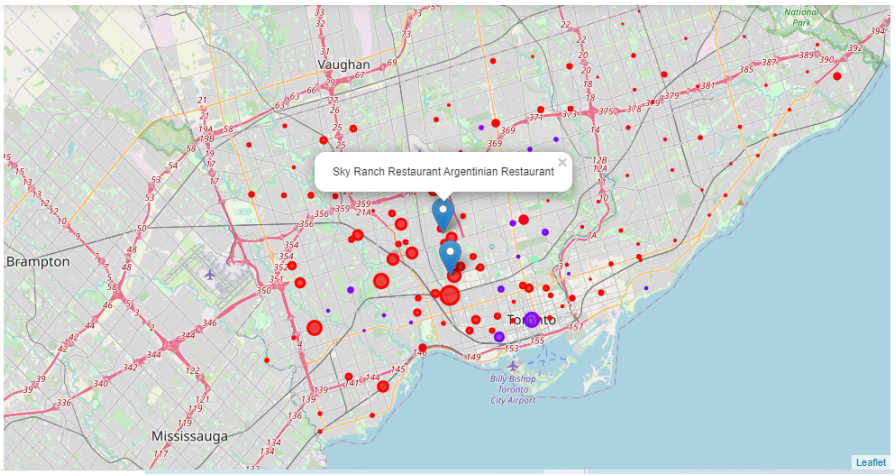

Figura 14: Klusters de barrios por densidad de migrantes sudamericanos y restaurantes de estilo.

Finalmente analizamos cada kluster para ver la información de población sudamericana e ingresos junto al mapa que tiene la ubicación de los restaurantes.

Out[78]:

	Cluster Labels	Hood #	Total income: Average amount ($)	Total migrantes sudamericanos	Total Normalizado	Total Income Norm	Latitude	Longitude
count	113.0	113.000000	113.000000	113.000000	113.000000	113.000000	113.000000	113.000000
mean	0.0	70.238938	41146.238938	206.327434	0.209469	0.133587	43.713215	-79.395150
std	0.0	42.461805	9981.387720	189.888381	0.192780	0.032406	0.053930	0.112252
min	0.0	1.000000	25989.000000	0.000000	0.000000	0.084377	43.592005	-79.599116
25%	0.0	32.000000	32724.000000	65.000000	0.065990	0.106243	43.670862	-79.480246
50%	0.0	69.000000	38527.000000	150.000000	0.152284	0.125084	43.706162	-79.409985
75%	0.0	111.000000	48511.000000	295.000000	0.299492	0.157498	43.759051	-79.310230
max	0.0	140.000000	65274.000000	985.000000	1.000000	0.211922	43.823174	-79.150768

Figura 15: Características del Klusters 0.

Out[79]:

	Cluster Labels	Hood #	Total income: Average amount ($)	Total migrantes sudamericanos	Total Normalizado	Total Income Norm	Latitude	Longitude
count	22.0	22.000000	22.000000	22.000000	22.000000	22.000000	22.000000	22.000000
mean	1.0	67.909091	92299.090909	165.227273	0.167743	0.299663	42.518891	-79.299054
std	0.0	32.586747	22090.128021	159.988670	0.162425	0.071719	5.471008	0.557705
min	1.0	9.000000	67757.000000	30.000000	0.030457	0.219983	18.024544	-79.539097
25%	1.0	40.500000	72008.750000	77.500000	0.078680	0.233787	43.654080	-79.455577
50%	1.0	72.500000	87335.000000	117.500000	0.119289	0.283546	43.670681	-79.400790
75%	1.0	96.500000	109185.250000	197.500000	0.200508	0.354486	43.705431	-79.375851
max	1.0	114.000000	144642.000000	745.000000	0.756345	0.469602	43.758950	-76.817072

Figura 16: Características del Klusters 1.

Out[80]:

	Cluster Labels	Hood #	Total income: Average amount ($)	Total migrantes sudamericanos	Total Normalizado	Total Income Norm	Latitude	Longitude
count	5.0	5.00000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000
mean	2.0	87.80000	210936.800000	113.000000	0.114721	0.684838	43.700763	-79.397081
std	0.0	26.30019	57708.657056	63.796552	0.064768	0.187360	0.021195	0.016614
min	2.0	41.00000	165047.000000	65.000000	0.065990	0.535849	43.678102	-79.413902
25%	2.0	96.00000	169203.000000	85.000000	0.086294	0.549343	43.686954	-79.409416
50%	2.0	98.00000	204521.000000	95.000000	0.096447	0.664008	43.693559	-79.403253
75%	2.0	101.00000	207903.000000	95.000000	0.096447	0.674988	43.716002	-79.381280
max	2.0	103.00000	308010.000000	225.000000	0.228426	1.000000	43.729199	-79.377554

Figura 17: Características del Klusters 2.

5. Resultados

Luego del detallado análisis hecho sobre la comunidad migrante sudamericana de Toronto y los restaurantes de comida típica de dicha comunidad, podemos deducir lo siguiente:

- La población migrante sudamericada está distribuida mayoritariamente dentro del kluster 0 de 113 barrios de Toronto.
- Existen solo dos restaurantes típicos sudamericanos en el área de Toronto, los mismos están ubicados en una área de mayor concentración de migrantes.Es el área de kluster 0 e incluye 113 barrios.
- Dicha área es la de menor ingreso promedio ($41146.23) de todas las agrupaciones obtenidas en el estudio.
- Las área de mayor ingreso son las de menor migración sudamericana promedio y las de mayor concentración son las de menores ingresos.

6. Conclusiones

Podemos concluir del análisis realizado lo siguiente:

- Dado la escasa cantidad de restaurantes de comida típica sudamericana, recomendamos la ubicación de dicho tipo de restaurantes en las áreas del kluster 2, que se encuentra contigua a la del kluster 0 (con mayor proporción de migrantes sudamericanos) pero con un ingreso medio bastante mayor.
- De todos modos, dada que solo existen dos restaurantes de este tipo, la ventana de oportunidad para la apertura de dicho tipo de restaurantes es bastante grande, con posibilidades de éxito mayor si se ubica en la zona recomendada en el punto anterior.
- Este análisis presenta algunos punto débiles que son:

  Los kluster se basan en la información del censo 2016, y por lo tanto hay una brecha de actualización de información de 5 años.

  Los datos de restaurantes típicos sudamericanos se basan unicamente en los datos de obtenidos de Foursquare API, lo cual podrían no ser ni actualizados ni completos.

- A pesar de que se pueden mejorar en los aspectos anteriores, este análisis ciertamente nos ha brindado una primera imagen del problema planteado, pudiendo contestar las preguntas planteadas inicialmente, aunque puede mejorarse el mismo con mayor y mejor información.
- En este proyecto tuvimos la oportunidad de abordar un problema empresarial como lo haría un científico de datos real. Usamos muchas bibliotecas de Python para obtener los datos, manipular los contenidos y analizar y visualizar esos conjuntos de datos. Usamos la API de Foursquare para explorar los negocios en los vecindarios de Toronto y buscar aquellos que nos interesaban. Usamos también datos del portal de datos de la Ciudad de Toronto, usamos las librerías seaborn, matplotlib y Folium para visualizar los datos en un mapa. También pudimos usar algoritmos de clasificación como K-means para agrupar los barrios con criterios apropiados.
- Hemos hecho uso de todas las herramientas aprendidas para resolver un problema de ciencia de datos real, el mismo puede extenderse a otros problemas similares como abrir gimnasios, tiendas de artesanías, copra de inmuebles, etc. Este proyecto constituye así una guía inicial para abordar mayores desafíos de ciencia de datos en la vida real.