Proyecto Capstone IBM

September 8, 2021

Proyecto Final Certificación IBM

Este notebook contiene el proyecto final o proyecto capstone de la certificacion de ciencia de datos de IBM. Se hace uso de la api de geolocalizacion de FourSquare.

Descripción del Problema

El negocio de los restaurante es un mercado grande y consolidado, en Estados Unidos representa el 5% del PIB y genera mas de 16 millones de empleos [1]. Siendo ademas un negocio muy rentable, lo hace atractivo para los inversionistas. Al momento de abrir un restaurante los inversionistas deben preocuparse por multiples caracteristicas como el ambiente, la iluminacion, la distribucion de las mesas, el tipo de menu, el servicio, los accesos para discapacitados, entre otros. Adicionalmente uno de los factores mas importantes es la ubicacion. La localizacion tiene gran influencia en el exito de un restaurante, y no resulta ser una tarea nada facil como se podria creer a simple vista. Segun [2] la apertura de un restaurante requiere una gran inversion de dinero y es necesario alcanzar el punto de equilibrio rapidamente. Segun [3] la evaluacion de riesgo antes de montar un negocio es importante para los emprendedores.}

Hoy dia gracias a la masificacion de internet, la creciente cantidad de datos digitales disponibles y el incremento en el poder de computo, se ha intensificado el uso de modelos de machine learning para la solucion de problemas que anteriormente solo podian ser abordados por expertos o directamente no eran manejables.

Por ello se plantea la pregunta ¿Sera posible utilizar analisis de datos para obtener sugerencias de lugares donde abrir un restaurante?

Antecedentes

En [4] se señala una una serie de variables sociodemograficas, psicograficas y de comportamiento de los potenciales clientes. Tambien sugiere tener en cuenta factores como el trafico peatonal y vehicular de la zona, los restaurantes cercanos identificando los competidores y los complementarios, y por ultimo otros tipos de infraestructuras como colegios, centros comerciales, entre otros, puesto que son un indicativo del desarrollo general de la zona.

En [5] se propone un marco de trabajo para abrir el restaurante, haciendo usos de los datos de la pagina de reseñas yelp. La metodologia consiste en 3 tareas. Primero identificar las caracteristicas mas apreciadas por un cliente, siendo la tarjeta de credito la mas apreciada, seguida por acceso para discapacitados y un buen servicio por parte de los camareros. Segundo identificaron cual es el dia con mayor aforo, siendo el lunes, esto en busca de generar las mejores ofertas para ese dia de la semana. Por ultimo se utiliza el algoritmo kd-tree para encontrar los restaurantes mas cercanos a una determinada localizacion, posteriormente se procede a analizar su tipo de cocina y

caracteristicas, en busca de generar un concepto de restaurante nuevo segun las caracteristicas del area.

En [6] los autores realizan utilizan los modelos arbol de desiciones, regresion logistica y SVM no linear para predecir la calificacion de un restaurante segun sus caracteristicas. SVM es de lejos el mejor modelo obteniendo una presicion de mas del 95% pero con un tiempo de procesamiento seis veces mayor al de los otros algoritmos, para este caso el tiempo no resulta ser un factor a tener en cuenta, por ello los autores ven la implementacion como satisfactoria. Tambien se realiza un analisis de las mejores ciudades para abrir un restaurante teniendo en cuenta unicamente la calificacion por parte de los clientes de los restaurantes de las mismas.

Datos

Para este proyecto se utilizara la fuente de datos de codigos postales y barrios de Toronto que se ha trabajado previamente en el curso. Adicionalmente se obtuvo una base de datos oficial del gobierno de Toronto que contiene datos demograficos como edad, sexo, idioma, educacion, ingresos entre otros. Estos datos se usaran para enriquecer el analisis y seleccionar de una mejor forma los barrios.

Otro recurso que se usara es la librería de geopy para encontrar los valores de latitud y longitud para cada uno de los barrios. Y la API de foursquare para obtener los datos de los negocio cercanos y su categoria

Metodologia

Obteniendo los datos de codigo postal de wikipedia

Para obtener los codigos postales de la ciudad de toronto se utilizo una tabla de wikipedia, junto con la libreria de beautiful soup para la extracion de los datos por medio de webscraping. Primero se lee el html de la pagina para luego filtrar la tabla por medio de las etiquetas de celda, dado que algunos codigos no estan asignados se utiliza un condicional para omitir estos valores del dataframe.

Obteniendo Datos Latitud y Longitud por codigo postal

La latitud y longitud de cada codigo postal se tiene en un archivo aparte, por lo cual se lee ese .csv y luego se hace un join utilizando el codigo postal como llave.

Graficando Resultado

Teniendo las coordenadas geograficas procedemos a graficar cada una de las ubicaciones, añadiendole ademas una etiqueta con el nombre del municipio y los barrios que pertenecen a cada codigo postal, en total obtenemos 103 localizaciones.

Definiendo datos para conexion con Foursquare

Se utilizara la api de foursquare para obtener los restaurante de cada codigo postal. Es necesario obtener un id y un secreto de cliente para ello se crea una cuenta en la aplicacion. Luego se obtienen los datos por medio de una api REST, la peticion devuelve una lista con los negocios a un determinado radio de la ubicacion dada.

Luego se procedio a filtrar los sitios que tenian en su categoria la palabra restaurante. Para luego calcular la frecuencia de cada sitio y ordenar los codigos postales segun la frecuencia, obteniendo asi las 10 categorias de restaurantes mas comunes.

Ahora se utilizara el algoritmo de k means para agrupar los barrios en 5 categorias.

Podemos observar que el primer cluster integrado por 9 codigos postales se compone de restaurantes mayoritariamente de comida rapida y comida vietnamita. El segundo cluster integrado por 5 codigos postales se compone principalmente de restaurantes de restaurantes italianos, restaurantes doner y restaurantes sin tematica. El tercer cluster compuestos por 3 ubicaciones se caracteriza por tener restaurantes de comida asiatica con cocinas vietnamita,china y de medio oriente. el cuarto cluster con 2 codigos postales se compone principalmente de restaurantes vietnamitas, italianos y japoneses, por ultimo el cluster 5 es de lejos el mas grande contiene 46 restaurantes pricipalmente de comida italiana, griega y sin tematica.

Conclusiones

Durante el analisis se identifico que la cocina mas comun en la ciudad de toronto es la vietnamita, la italiana y la doner. Para el cluster 5 que fue de lejos el mas numeroso vemos una tendencia a estar ubicados en el centro de la ciudad o un poco hacia el norte. Analizando el numero de codigos postales de cada cluster podriamos dividir a grandes rasgos a la ciudad de Toronto en dos, donde uno seria el cluster 5 caracterizado por comida italiana y el otro cluster serian los demas siendo 19 restaurantes en total caraterizado por comida vietnamita y donde sus ubicaciones son mas hacia la periferia de la ciudad.

[ ]:
```
#Importando la libreria para web
from bs4 import BeautifulSoup
import requests
#Importando libreria manejo de datos
import pandas as pd
import numpy as np
#Importando libreria graficar mapas
import folium
#Importar k-means desde la fase de agrupación
from sklearn.cluster import KMeans
#Modulos matplotlib grafica final
import matplotlib.cm as cm
import matplotlib.colors as colors
```

[ ]:
```
#Definimos la url de wikipedia
url = "https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M"
#Leemos el html como texto
data = requests.get(url).text
#Leemos el html con BS
soup = BeautifulSoup(data,"html.parser")
#Leyendo la tabla
table = soup.find("table")
dtBarToronto = pd.DataFrame(columns=['postalCode', 'borough', 'neighborhood'])
#Contador casillas no asignadas
contNA = 0
#Ciclo para las filas
for row in table.find_all("tr"):
    #Ciclo para las columnas
    for col in row.find_all("td"):
```

```
        #Condicional ignorar celdas no asignadas
        if col.find(string="Not assigned") == None:
            #Obteniendo lista de strings en la celda
            colStrings = col.p.strings
            neighborhood = ""
            #Ciclo para recorrer la lista de strings
            for i,colString in enumerate(colStrings):
                uniString = str(colString)
                #Primera posicion es el codigo postal
                if i == 0:
                    postalCode = uniString
                #Segunda posicion es el municipio
                elif i == 1:
                    borough = uniString
                else:
                    neighborhood = neighborhood + uniString
            #Cambiando el separados de los barrios
            neighborhood = neighborhood.strip().replace("(","").replace(")","").
 ↪replace(" /",",")
            #Generando el diccionario con los datos de la fila
            nuevaFila = {"postalCode":postalCode,"borough":
 ↪borough,"neighborhood":neighborhood}
            #Agregando una nueva fila
            dtBarToronto = dtBarToronto.append(nuevaFila,ignore_index=True)
        else:
            contNA += 1
print(contNA,"Casillas No Asignadas")
```

```
77 Casillas No Asignadas
```

Anexos

[ ]:
```
#Tamaño del dataframe resultante
dtBarToronto.shape
```

[ ]:
```
(103, 3)
```

[ ]:
```
#Tamaño del dataframe resultante
dtBarToronto.head()
```

[ ]:

	postalCode	borough	neighborhood
0	M3A	North York	Parkwoods
1	M4A	North York	Victoria Village
2	M5A	Downtown Toronto	Regent Park, Harbourfront
3	M6A	North York	Lawrence Manor, Lawrence Heights
4	M7A	Queen's Park	Ontario Provincial Government

[ ]:
```
#Leyendo el archivo csv con las coordenadas geoespaciales
df_spacial = pd.read_csv("Geospatial_Coordinates.csv")
df_spacial.head()
```

[ ]:
```
  Postal Code   Latitude  Longitude
0         M1B  43.806686 -79.194353
1         M1C  43.784535 -79.160497
2         M1E  43.763573 -79.188711
3         M1G  43.770992 -79.216917
4         M1H  43.773136 -79.239476
```

[ ]:
```
#Haciendo Join con el codigo postal
dtToronto = dtBarToronto.set_index("postalCode").join(df_spacial.
 ↪set_index("Postal Code"))
#Volviendo el codigo postal una columna
dtToronto.reset_index(inplace=True)
dtToronto.head()
```

[ ]:
```
  postalCode           borough                     neighborhood   Latitude  \
0        M3A        North York                        Parkwoods  43.753259
1        M4A        North York                  Victoria Village  43.725882
2        M5A  Downtown Toronto         Regent Park, Harbourfront  43.654260
3        M6A        North York  Lawrence Manor, Lawrence Heights  43.718518
4        M7A      Queen's Park     Ontario Provincial Government  43.662301

   Longitude
0 -79.329656
1 -79.315572
2 -79.360636
3 -79.464763
4 -79.389494
```

[ ]:
```
#Generando mapa Toronto
map_toronto = folium.Map(location = [43.6532,-79.3832], zoom_start = 11)
#Añadiendo marcadores para cada barrio
for lat,lon,nei,bor in␣
 ↪zip(dtToronto["Latitude"],dtToronto["Longitude"],dtToronto["neighborhood"],dtToronto["borou
 ↪
    label = '{},{}'.format(nei,bor)
    label = folium.Popup(label,parse_html=True)
    folium.
 ↪CircleMarker(location=[lat,lon],radius=5,popup=label,color="green",fill=True,fill_color="#9
 ↪add_to(map_toronto)
#Graficando mapa
map_toronto
```

[ ]:
```
<folium.folium.Map at 0x1b19e748>
```

[ ]:
```
#Configuracion Foursquare
#CLIENT_ID = 'VL21BT4LPJFFGOUHO5TJWRWOKXOUI5ZNOOC2T24OI544KJTU' # su ID de␣
↪Foursquare
#CLIENT_SECRET = 'EDWOK1HJXYY2TCSL5DNA1OQOBG1PVBVO1ZWJCDGFUFBSAOUC' # Secreto␣
↪de Foursquare
CLIENT_ID = '3KQ23TNDRE4U545FXH421FR5OEUAJOUU4PKVNC4XNHRE3LKM' # su ID de␣
↪Foursquare
CLIENT_SECRET = 'L1MSXE30SXDOQ4CNBTSMIRK2ZAXX5NZ1S3S2IRU1OTMMEBEA' # Secreto de␣
↪Foursquare
VERSION = '20180605' # versión de la API de Foursquare
LIMIT = 100 # Un valor límite para la API de Foursquare
```

[ ]:
```
#Definimos funcion para obtener categoria de los lugares
def obtener_categoria(row):
    try:
        lista_categoria = row["categories"]
    except:
        lista_categoria = row["venue.categories"]
    if len(lista_categoria) == 0:
        return None
    else:
        return lista_categoria[0]["name"]
```

[ ]:
```
#Definimos funcion para obtener los lugares para cada barrio
def obtener_lugares(names,latitudes,longitudes,radius=500):
    lista_lugares = []
    for name,lat,lon in zip(names,latitudes,longitudes):
        #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,lon,radius,LIMIT)
        #solicitud GET
        results = requests.get(url).json()["response"]['groups'][0]['items']
        #regresa solo información relevante de cada sitio cercano
        lista_lugares.
↪append([(name,lat,lon,v['venue']['name'],v['venue']['location']['lat'],v['venue']['location
↪for v in results])
        lugares_cercanos = pd.DataFrame([item for lugar in lista_lugares for␣
↪item in lugar])
        lugares_cercanos.columns = ['Neighborhood','Neighborhood␣
↪Latitude','Neighborhood Longitude','Venue','Venue Latitude','Venue␣
↪Longitude','Venue Category']

    return(lugares_cercanos)
```

[ ]:
```
#Llamamos a la funcion anterior
lugares_toronto =␣
↪obtener_lugares(dtToronto["neighborhood"],dtToronto["Latitude"],dtToronto["Longitude"])
lugares_toronto.shape
```

[ ]:
```
(2138, 7)
```

[ ]:
```
lugares_toronto.head()
```

[ ]:
```
        Neighborhood  Neighborhood Latitude  Neighborhood Longitude  \
0          Parkwoods              43.753259              -79.329656
1          Parkwoods              43.753259              -79.329656
2          Parkwoods              43.753259              -79.329656
3  Victoria Village              43.725882              -79.315572
4  Victoria Village              43.725882              -79.315572

                    Venue  Venue Latitude  Venue Longitude  \
0         Brookbanks Park       43.751976       -79.332140
1                     KFC       43.754387       -79.333021
2           Variety Store       43.751974       -79.333114
3  Victoria Village Arena       43.723481       -79.315635
4            Tim Hortons       43.725517       -79.313103

         Venue Category
0                  Park
1  Fast Food Restaurant
2     Food & Drink Shop
3          Hockey Arena
4           Coffee Shop
```

[ ]:
```
restaurantes_toronto = lugares_toronto[lugares_toronto["Venue Category"].str.
↪contains("Restaurant")].reset_index(drop=True)
restaurantes_toronto.shape
```

[ ]:
```
(497, 7)
```

[ ]:
```
# generando una columna para cada categoria
toronto_onehot = pd.get_dummies(restaurantes_toronto[['Venue Category']],␣
↪prefix="", prefix_sep="")

# añadir la columna de barrio de regreso al dataframe
toronto_onehot['Neighborhood'] = restaurantes_toronto['Neighborhood']

# mover la columna de barrio a la primer columna
fixed_columns = [toronto_onehot.columns[-1]] + list(toronto_onehot.columns[:-1])
toronto_onehot= toronto_onehot[fixed_columns]
```

```
toronto_onehot.head()
```

[ ]:

	Neighborhood	Afghan Restaurant	American Restaurant	Asian Restaurant	Belgian Restaurant	Brazilian Restaurant	Cajun / Creole Restaurant	Caribbean Restaurant	Chinese Restaurant	Colombian Restaurant	…	Restaurant	Seafood Restaurant	Sushi Restaurant	Taiwanese Restaurant	Thai Restaurant	Theme Restaurant	Tibetan Restaurant	Turkish Restaurant	Vegetarian / Vegan Restaurant	Vietnamese Restaurant
0	Parkwoods	0	0	0	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0
1	Victoria Village	0	0	0	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0
2	Regent Park, Harbourfront	0	0	0	0	0	0	0	0	0	…	1	0	0	0	0	0	0	0	0	0
3	Regent Park, Harbourfront	0	0	0	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0
4	Regent Park, Harbourfront	0	0	0	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0

```
2                       0
3                       0
4                       0

[5 rows x 52 columns]
```

[ ]:
```
#obteniendo la frecuencia de cada categoria para cada barrio
toronto_grouped = toronto_onehot.groupby('Neighborhood').mean().reset_index()
toronto_grouped.head()
```

[ ]:
```
                                  Neighborhood  Afghan Restaurant  \
0                                    Agincourt                0.0
1  Bathurst Manor, Wilson Heights, Downsview North                0.0
2                              Bayview Village                0.0
3           Bedford Park, Lawrence Manor East                0.0
4                                  Berczy Park                0.0

   American Restaurant  Asian Restaurant  Belgian Restaurant  \
0                  0.0               0.0            0.000000
1                  0.0               0.0            0.000000
2                  0.0               0.0            0.000000
3                  0.1               0.0            0.000000
4                  0.0               0.0            0.066667

   Brazilian Restaurant  Cajun / Creole Restaurant  Caribbean Restaurant  \
0                   0.0                        0.0                   0.0
1                   0.0                        0.0                   0.0
2                   0.0                        0.0                   0.0
3                   0.0                        0.0                   0.0
4                   0.0                        0.0                   0.0

   Chinese Restaurant  Colombian Restaurant  …  Restaurant  \
0                 0.0                   0.0  …    0.000000
1                 0.0                   0.0  …    0.333333
2                 0.5                   0.0  …    0.000000
3                 0.0                   0.0  …    0.100000
4                 0.0                   0.0  …    0.133333

   Seafood Restaurant  Sushi Restaurant  Taiwanese Restaurant  \
0            0.000000          0.000000                   0.0
1            0.000000          0.333333                   0.0
2            0.000000          0.000000                   0.0
3            0.000000          0.100000                   0.0
4            0.133333          0.000000                   0.0

   Thai Restaurant  Theme Restaurant  Tibetan Restaurant  Turkish Restaurant  \
0         0.000000               0.0                 0.0                 0.0
```

```
1         0.000000                0.0                 0.0                 0.0
2         0.000000                0.0                 0.0                 0.0
3         0.100000                0.0                 0.0                 0.0
4         0.066667                0.0                 0.0                 0.0

   Vegetarian / Vegan Restaurant  Vietnamese Restaurant
0                       0.000000                    0.0
1                       0.000000                    0.0
2                       0.000000                    0.0
3                       0.000000                    0.0
4                       0.066667                    0.0

[5 rows x 52 columns]
```

[ ]:
```
#Funcion para retornar lugares mas comunes
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

[ ]:
```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# crear las columnas acorde al numero de sitios populares
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# crear un nuevo dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = toronto_grouped['Neighborhood']

for ind in np.arange(toronto_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] =␣
 ↪return_most_common_venues(toronto_grouped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted.shape
```

[ ]:
```
(65, 11)
```

[ ]:
```
# establecer el número de agrupaciones
kclusters = 5
```

```
toronto_grouped_clustering = toronto_grouped.drop('Neighborhood', 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
type(kmeans.labels_)
unique, counts = np.unique(kmeans.labels_, return_counts=True)
dict(zip(unique, counts))
```

[ ]:
```
{0: 9, 1: 5, 2: 3, 3: 2, 4: 46}
```

[ ]:
```
# añadir etiquetas
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

toronto_merged = dtToronto

# juntar manhattan_grouped con manhattan_data
toronto_merged = toronto_merged.join(neighborhoods_venues_sorted.
 ↪set_index('Neighborhood'), on='neighborhood', how="right")

toronto_merged['Cluster Labels'].value_counts

#toronto_merged.head() # revisar las ultimas columnas
```

[ ]:
```
<bound method IndexOpsMixin.value_counts of 78    4
28    4
39    4
55    4
20    4
     ..
11    4
70    4
71    2
59    4
22    4
Name: Cluster Labels, Length: 65, dtype: int32>
```

[ ]:
```
# crear mapa
map_clusters = folium.Map(location=[43.6532,-79.3832], 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 in zip(toronto_merged['Latitude'],␣
 ↪toronto_merged['Longitude'], toronto_merged['neighborhood'],␣
 ↪toronto_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)
map_clusters
```

[ ]: <folium.folium.Map at 0x5964bb0>

[ ]:
```
group1=toronto_merged.loc[toronto_merged['Cluster Labels'] == 0, toronto_merged.
 ↪columns[[0] + list(range(5, toronto_merged.shape[1]))]]
print("First",group1["1st Most Common Venue"].mode())
print("Second",group1["2nd Most Common Venue"].mode())
print("Third",group1["3rd Most Common Venue"].mode())
```

```
First 0    Fast Food Restaurant
dtype: object
Second 0    Vietnamese Restaurant
dtype: object
Third 0         Doner Restaurant
1    Vietnamese Restaurant
dtype: object
```

[ ]:
```
group1=toronto_merged.loc[toronto_merged['Cluster Labels'] == 1, toronto_merged.
 ↪columns[[0] + list(range(5, toronto_merged.shape[1]))]]
print("First",group1["1st Most Common Venue"].mode())
print("Second",group1["2nd Most Common Venue"].mode())
print("Third",group1["3rd Most Common Venue"].mode())
```

```
First 0    Italian Restaurant
dtype: object
Second 0    Restaurant
dtype: object
Third 0    Doner Restaurant
dtype: object
```

[ ]:
```
group1=toronto_merged.loc[toronto_merged['Cluster Labels'] == 2, toronto_merged.
 ↪columns[[0] + list(range(5, toronto_merged.shape[1]))]]
print("First",group1["1st Most Common Venue"].mode())
print("Second",group1["2nd Most Common Venue"].mode())
print("Third",group1["3rd Most Common Venue"].mode())
```

```
First 0    Vietnamese Restaurant
dtype: object
Second 0              Doner Restaurant
1    Middle Eastern Restaurant
2        Vietnamese Restaurant
dtype: object
Third 0    Chinese Restaurant
1      Greek Restaurant
2     Indian Restaurant
dtype: object
```

[ ]:
```
group1=toronto_merged.loc[toronto_merged['Cluster Labels'] == 3, toronto_merged.
 ↪columns[[0] + list(range(5, toronto_merged.shape[1]))]]
print("First",group1["1st Most Common Venue"].mode())
print("Second",group1["2nd Most Common Venue"].mode())
print("Third",group1["3rd Most Common Venue"].mode())
```

```
First 0    Caribbean Restaurant
1     Japanese Restaurant
dtype: object
Second 0     Caribbean Restaurant
1    Vietnamese Restaurant
dtype: object
Third 0         Doner Restaurant
1    Vietnamese Restaurant
dtype: object
```

[ ]:
```
group1=toronto_merged.loc[toronto_merged['Cluster Labels'] == 4, toronto_merged.
 ↪columns[[0] + list(range(5, toronto_merged.shape[1]))]]
print("First",group1["1st Most Common Venue"].mode())
print("Second",group1["2nd Most Common Venue"].mode())
print("Third",group1["3rd Most Common Venue"].mode())
print("Fourth",group1["4th Most Common Venue"].mode())
print("Fifth",group1["5th Most Common Venue"].mode())
```

```
First 0    Italian Restaurant
dtype: object
Second 0    Italian Restaurant
dtype: object
Third 0    Restaurant
dtype: object
Fourth 0    Greek Restaurant
dtype: object
```

```
Fifth 0          Doner Restaurant
1    Gluten-free Restaurant
dtype: object
```

Bibliografia

[1] “National restaurent association,” http://www.restaurant.org/Downloads/PDFs/News-Research/2017 Restaurant outlook summary-FINAL.pdf, 2017.

[2] Raul, N., Shah, Y., Devganiya, M.: Restaurant revenue prediction using machine learning. Int. J. Eng. Sci. 6(4), 91–94 (2016)

[3] S.Khatwani and M. Chandak, “Building Personalized and Non Personalized recommendation systems”, in 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), Pune, India, 2018, pp. 623-628.

[4] Das Baksi, B., Rao, V. and Anitha, C., 2018. A Survey on Local Market Analysis for a Successful Restaurant Yield. Advances in Intelligent Systems and Computing, pp.249-257.

[5] Hegde, S., Satyappanavar, S. and Setty, S., 2017. Restaurant setup business analysis using yelp dataset. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI),.

[6] Shihab, I., Oishi, M., Islam, S., Banik, K. and Arif, H., 2018. A Machine Learning Approach to Suggest Ideal Geographical Location for New Restaurant Establishment. 2018 IEEE Region 10 Humanitarian Technology Conference (R10-HTC),.