São Paulo Overview - A data science approach to the biggest south american city

Carlos Barreto

January 12, 2021

1. Introduction

1.1. Background

São Paulo is the largest and most populous brazilian municipality, capital of a homonymous state and the most influential corporate and financial center of South America.
It is a city that welcomes internal immigrants (other cities and states in the country) in addition to foreign communities, thus printing a character multicultural and cosmopolitan city.
In addition, São Paulo is a city that annually receives around 15 million tourists, both brazilian and foreign, who are looking for different experiences in the city.

1.2. Problem

We will then explore the city and group the localities according to the venues categories most present in each neighborhood.

2. Data Sources

2.1. Postcode, City, Neighborhood

The 'Great São Paulo' is a vast metropolis, composed of the conurbation of a very large number of municipalities and with a population of approximately 20 million people. There are several levels of organization in the city and countless sub-municipalities, administrative regions and neighborhoods. There is compiled data that lists more than 1500 neighborhoods.
The list used in this work presents the 88 main neighborhoods in the city and was obtained from the website:

https://www.estadosecapitaisdobrasil.com/listas/lista-dos-bairros-de-sao-paulo/

2.2. Geospatial data

All geospatial coordinates were obtained using the geopy package and specifically the data provider ARCGIS, chosen specifically because it does not require the creation of an access key for simple data retrieval (latitude and longitude) of a location.

2.3. Foursquare API

The get method of the 'explore' endpoint will be used in particular, which returns the following information by default: Neighborhood, Neighborhood Latitude, Neighborhood Longitude, Venue, etc.

3. Methodology

3.1. Python libraries

All data were collected, processed and visualized using the libraries: Numpy, Pandas, Geocoder, Folium, Json, Requests and Sklearn.
The list of neighborhoods was the starting point of the work, these data were successively enriched and finally the neighborhoods were clustered with the application of the KMeans algorithm available in the Sklearn package.

3.2. Data gathering

The initial data was obtained on this website, which lists the 88 main neighborhoods in São Paulo, and these data were saved in a csv (link to github) to facilitate loading via pandas. The site has a semi-structured structure and the difficulty related to its scraping is outside the scope of this work.

```
df = pd.read_csv('bairros_puro.csv')
df.head()
```

	Bairro
0	Água Rasa
1	Alto de Pinheiros
2	Anhanguera
3	Aricanduva
4	Artur Alvim

3.3. Data pre processing

Like many languages of Latin origin, Portuguese uses a considerable set of special characters. All were removed to avoid treatment failures and subsequent visualization.

```
from unicodedata import normalize
def removeSpecialChar (text) :
        return normalize('NFKD', text).encode('ASCII', 'ignore').decode('ASCII')
```

```
df['Bairro'] = df['Bairro'].apply(removeSpecialChar)
df.rename(columns={'Bairro':'Neighborhood'}, inplace=True)
df.head()
```

	Neighborhood
0	Agua Rasa
1	Alto de Pinheiros
2	Anhanguera
3	Aricanduva
4	Artur Alvim

3.4. Adding geolocation data

Once the special characters have been removed, we can call the geolocation API available in the geocoder package, using the name of each neighborhood, city and respective country as an argument.

The API responses were stored in a dictionary, a structure that is easy to manipulate and transform into the tabular form of the pandas.

```
df_geolocation = pd.DataFrame.from_dict(aux_dict, orient = 'index', columns = ['Latitude', 'Longitude'])
df_geolocation.reset_index(inplace = True)
df_geolocation = df_geolocation.rename(columns = {'index':'Neighborhood'})
df_geolocation.head()
```

	Neighborhood	Latitude	Longitude
0	Agua Rasa	-23.55337	-46.58027
1	Alto de Pinheiros	-23.55273	-46.70916
2	Anhanguera	-23.42097	-46.78517
3	Aricanduva	-23.56771	-46.51025
4	Artur Alvim	-23.55105	-46.48000

```
def get_latlgn(Neighborhood):
    g = geocoder.arcgis('{}, Sao Paulo, BR'.format(Neighborhood))
    return g.latlng
```

```
aux_dict = {}
count = 1
df_size = len(df['Neighborhood'])
for elem in df['Neighborhood']:
    aux_dict[elem] = get_latlgn(elem)
    print('{} : done - {}/{}'.format(elem, count, df_size))
    count += 1
```

```
Agua Rasa : done - 1/88
Alto de Pinheiros : done - 2/88
Anhanguera : done - 3/88
Aricanduva : done - 4/88
Artur Alvim : done - 5/88
Barra Funda : done - 6/88
Bela Vista : done - 7/88
Belem : done - 8/88
```

3.5. Data unification and Visualization

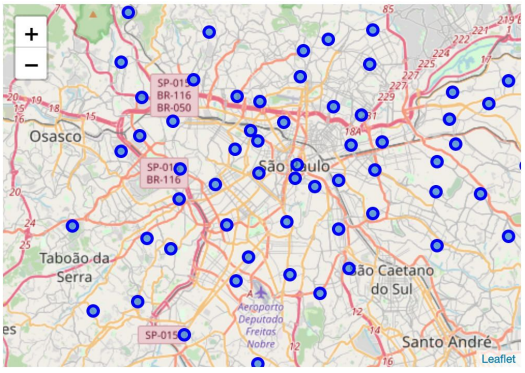
Once a new dataframe is created from the dictionary with geolocation data, we can then unify it with the original dataframe with the name of the neighborhoods.

We now have enough information to create a first view of the city using the folium package.

```
df_sp = pd.concat([df,df_geolocation[['Latitude','Longitude']]], axis = 1)
df_sp.head()
```

	Neighborhood	Latitude	Longitude
0	Agua Rasa	-23.55337	-46.58027
1	Alto de Pinheiros	-23.55273	-46.70916
2	Anhanguera	-23.42097	-46.78517
3	Aricanduva	-23.56771	-46.51025
4	Artur Alvim	-23.55105	-46.48000

```
map_sp = folium.Map(width=500, height=350, location=[sp_geolocation[0], sp_geolocation[1]], zoom_start = 11)

for lat, lng, neighborhood in zip(df_sp['Latitude'], df_sp['Longitude'], df_sp['Neighborhood']):
    label = '{}'.format(neighborhood)
    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_sp)
map_sp
```



3.6. Exploring the city using Foursquare

Having the neighborhoods defined by their names and geographical coordinates, we have the possibility to use the Foursquare API to retrieve the respective venues for each neighborhood.

The version of the API used is ‘20180605’ in addition to a limit of 100 venues per response within a radius of 500 meters around the coordinate employed.

```
import json
import requests
from pandas.io.json import json_normalize # Convert JSON into pandas dataframe

from foursquare_credentials import CLIENT_ID, CLIENT_SECRET

VERSION = '20180605' # API version
LIMIT = 100 # Default API limite value
RADIUS = 500
df_sp['Neighborhood'].unique()
```

```
print(sp_venues.shape)
sp_venues.head()
```

```
(2330, 7)
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Agua Rasa	-23.55337	-46.58027	Padaria Carillo	-23.553214	-46.578554	Bakery
1	Agua Rasa	-23.55337	-46.58027	Portuga Bar e Restaurante	-23.553846	-46.579573	Brazilian Restaurant
2	Agua Rasa	-23.55337	-46.58027	Chama Supermercados	-23.554178	-46.581120	Market
3	Agua Rasa	-23.55337	-46.58027	Bona's Carnes	-23.552434	-46.583091	Steakhouse
4	Agua Rasa	-23.55337	-46.58027	Padaria Santa Branca	-23.553953	-46.583706	Bakery

3.7. Prepare the dataset for the clustering algorithm

With all the relevant data collected, we started processing for the application of the clustering algorithm, which will allow us to find similarities between different neighborhoods in the city.

```
# Applying one hot encoding
sp_onehot = pd.get_dummies(sp_venues[['Venue Category']], prefix='', prefix_sep='')

# add neighbourdhood back to dataframe
sp_onehot['Neighborhood'] = sp_venues['Neighborhood']

# move neighborhood column to first column
fixed_columns = ['Neighborhood'] +[col for col in sp_onehot.columns if col != 'Neighborhood' ]
sp_onehot = sp_onehot[fixed_columns]

sp_onehot.head()
```

	Neighborhood	Acai House	Accessories Store	American Restaurant	Arcade	Argentinian Restaurant	Art Gallery	Art Museum	Art Studio	Arts & Crafts Store	...
0	Agua Rasa	0	0	0	0	0	0	0	0	0	...
1	Agua Rasa	0	0	0	0	0	0	0	0	0	...
2	Agua Rasa	0	0	0	0	0	0	0	0	0	...
3	Agua Rasa	0	0	0	0	0	0	0	0	0	...
4	Agua Rasa	0	0	0	0	0	0	0	0	0	...

5 rows × 269 columns

3.8. Clustering application

The KMeans algorithm available in the SKlean python package was applied to the treated data set.

```
# import k-means from clustering stage
from sklearn.cluster import KMeans
```

```
sp_grouped_clustering_aux = sp_grouped.copy(deep=True)
sp_grouped_clustering = sp_grouped.drop('Neighborhood',1)
sp_grouped_clustering.head()
```

	Acai House	Accessories Store	American Restaurant	Arcade	Argentinian Restaurant	Art Gallery	Art Museum	Art Studio	Arts & Crafts Store	Arts & Entertainment	...
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.041667	0.0	...
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...

5 rows × 268 columns

```
# set number of clusters
kclusters = 4

# run k-mens clustering
kmeans = KMeans(n_clusters = kclusters, random_state=0).fit(sp_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:30]
```

```
array([1, 1, 0, 1, 2, 1, 1, 1, 1, 1, 0, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,
       2, 1, 1, 1, 1, 2, 1, 1], dtype=int32)
```

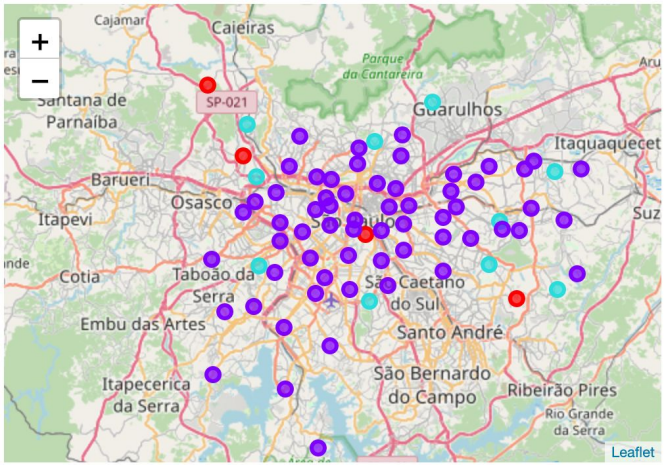
3.9. What are the most common locations in each neighborhood and which cluster is in each neighborhood?

```
# Identifying the top 10 most common venues
num_top_venues = 10

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

# create columns according to number of top venues
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))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = sp_grouped['Neighborhood']

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

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th M Com Ve
0	Agua Rasa	-23.55337	-46.58027	1.0	Bakery	Brazilian Restaurant	Furniture / Home Store	Farmers Market	Gym / Fitness Center	Soccer Field	Pharmacy	Garden Center	Café	Steakho
1	Alto de Pinheiros	-23.55273	-46.70916	1.0	Plaza	Convenience Store	Pharmacy	Bagel Shop	Dog Run	Restaurant	Fast Food Restaurant	Flea Market	Café	Busin Serv
2	Anhanguera	-23.42097	-46.78517	0.0	Brazilian Restaurant	Lake	Restaurant	Zoo	Flower Shop	Farm	Farmers Market	Fast Food Restaurant	Fish Market	F Mar
3	Aricanduva	-23.56771	-46.51025	1.0	Gym / Fitness Center	Farmers Market	Food & Drink Shop	Fast Food Restaurant	Grocery Store	Bank	Bakery	Brazilian Restaurant	Dessert Shop	Pet St
4	Artur Alvim	-23.55105	-46.48000	2.0	Bakery	Pharmacy	Ice Cream Shop	Farmers Market	Flower Shop	Supermarket	Tea Room	Pizza Place	History Museum	Food Tr

3.10. Viewing clustered neighborhoods

Once the cluster identification tags for each neighborhood have been defined, we can explore this new information using the folium.

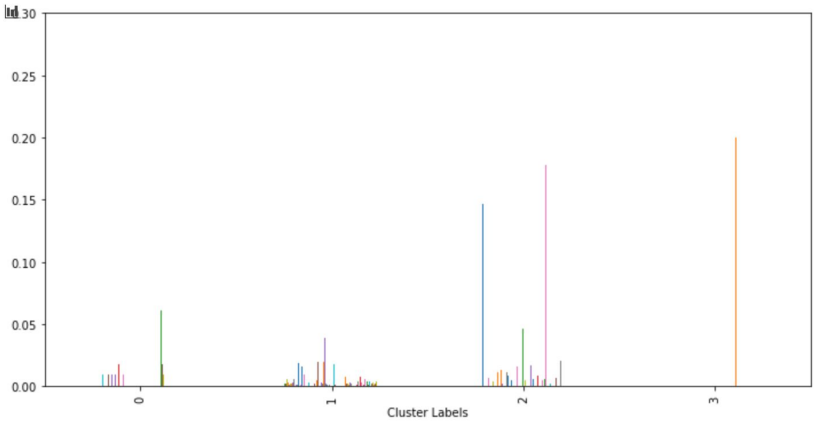


3.11. Clusters profile

In addition to the neighborhoods on the map, we can also view the profile of the clusters according to the average occurrence of types of establishments in each cluster.

```
table = pd.pivot_table(sp_grouped_clustering_aux, aggfunc='mean', index='Cluster Labels')
table.fillna(0)
table.plot(kind='bar', figsize=(12,6), legend=None)
import matplotlib.pyplot as plt
plt.ylim(ymax = 0.3, ymin = 0)
```

```
(0.0, 0.3)
```



3.12. Checking the composition of each cluster

Finally, we can also check the precise composition of each cluster. Bearing in mind that this classification is directly dependent on the venues present in each neighborhood.

All clusters can be viewed in the notebook available on github (link)

```
def printClusterComposition(df):
    qtd_clusters = df['Cluster Labels'].max() + 1
    for cluster in range(qtd_clusters):
        temp = df[df['Cluster Labels'] == cluster].mean().T
        temp.drop(['Cluster Labels'],inplace=True)
        with pd.option_context('display.max_rows', 500):
            print('Cluster {} :'.format(cluster))
            print(temp[temp>0])
            print()
```

```
Cluster 0 :
Bagel Shop              0.009615
Bakery                  0.054945
Bistro                  0.009615
Brazilian Restaurant    0.277930
Burger Joint            0.009615
Café                    0.009615
Chocolate Shop          0.009615
Clothing Store          0.009615
Convenience Store       0.017857
Cosmetics Shop          0.009615
Dessert Shop            0.041667
Diner                   0.017857
Farmers Market          0.009615
Gym / Fitness Center    0.041667
Ice Cream Shop          0.017857
Italian Restaurant      0.009615
Lake                    0.083333
Pet Store               0.060897
Pharmacy                0.045330
Pie Shop                0.017857
Pizza Place             0.037088
Plaza                   0.009615
Restaurant              0.120421
Spa                     0.009615
Supermarket             0.059524
dtype: float64
```

```
Cluster 3 :
Bakery            0.2
Burger Joint      0.2
Ice Cream Shop    0.2
Pet Service       0.2
Spa               0.2
dtype: float64
```

4. Results and Discussion

São Paulo is a vast and multicultural city, with numerous options for places to visit. It is possible to find all kinds of restaurants, from local to Asian style, passing by European classics (Italian, Greek and etc.)

In a quick segmentation, we can find wealth in the movement from central neighborhoods, with all kinds of services available (cluster 0) to more remote neighborhoods with purely residential characteristics (cluster 3), in addition to intermediate segments that deserve a more detailed exploration, composing thus an excellent theme for future work.

For anyone wishing to visit São Paulo, you will undoubtedly find an incredible city always ready to welcome people who are interested in exploring its nightlife, its gastronomic scene and its art circuit.

5. Conclusion

This work aims to explore the city of São Paulo by leveraging intelligence about free data available on the web about the city in addition to the application of python technology, apis and data analysis.

I had not yet found a job that dealt with Brazilian cities, so I decided to explore the city in which I currently reside and which I am completely passionate about.

This concludes a mandatory component of the IBM Data Science Professional Certificate course, the complete code of which can be found on the author's github (link).