Francisco Antunes

Neighborhood recommendation, based on people and homes density, average wage and nearby services

Problem

I like my current neighborhood, but I need to move… It's the chance of my life!!!!
Well… **I have a suggestion:**

I will give you some options of neighborhood similar to the one you live, but that it's closer to the new job :)

Methodology

1. Define the data that I'll use
2. Look for data sources
3. Explore the data
4. Clean and Format data
5. Use clustering algorithm to group neighborhoods
6. Make some calculation
7. Return the recommended neighborhoods.

Define and take a look at the data

- **List of all neighborhoods**. I’ve chosen my city as example (Curitiba, Brazil). This data is available on Wikipedia:

  https://pt.wikipedia.org/wiki/Lista_de_bairros_de_Curitiba
- **Venues nearby each neighborhood:** I’ve used Foursquare API.
- **Coordinates (latitude and longitude) of each neighborhood:**

  geopy.geocoder python library.

Explore the data

- Many things to do

```
df_neigh
```

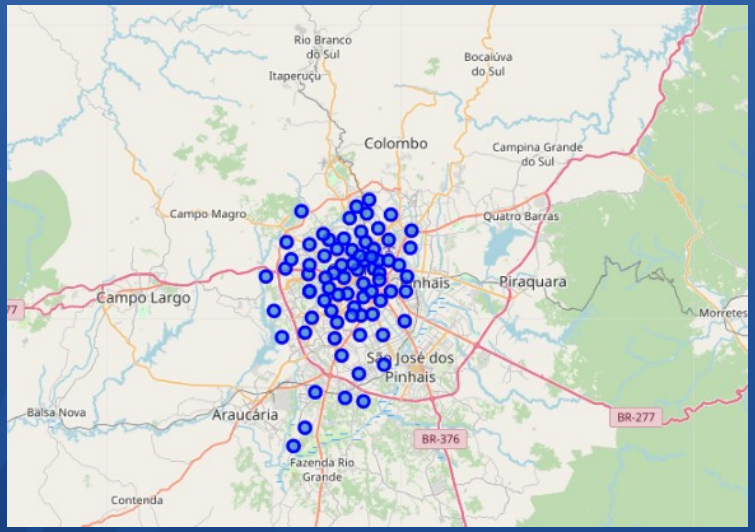
	0	1	2	3	4	5	6	vdeBairros de Curitiba	vdeBairros de Curitiba.1	vdeBairros de Curitiba.2
0	Ganchinho	1120	3667	3658	7325	1921	76735	NaN	NaN	NaN
1	Sitio Cercado	1112	50631	51779	102410	27914	93495	NaN	NaN	NaN
2	Umbará	2247	7280	7315	14595	17064	90870	NaN	NaN	NaN
3	Abranches	432	5463	5702	11165	3154	100967	NaN	NaN	NaN
4	Atuba	427	6156	6476	12632	3627	121160	NaN	NaN	NaN
5	Bacacheri	698	10762	12344	23106	7107	302900	NaN	NaN	NaN
6	Bairro Alto	702	20244	21789	42033	12071	121160	NaN	NaN	NaN
7	Barreirinha	373	8079	8942	17021	5024	127218	NaN	NaN	NaN
8	Boa Vista	514	13677	15714	29391	9212	181740	NaN	NaN	NaN
9	Cachoeira	307	3811	3927	7738	2091	90870	NaN	NaN	NaN
10	Pilarzinho	713	13358	14549	27907	7883	121160	NaN	NaN	NaN
11	Santa Cândida	1033	13504	14366	27870	7937	121160	NaN	NaN	NaN
12	São Lourenço	226	2611	2945	5556	1603	242320	NaN	NaN	NaN

Clean and Format data

- Remove NaN values (rows and columns)
- Define column names
- Format numbers for each column
- Adjust average wage values to have all in the same year. Observation: some data was from 2000 and others to 2010, so I decided to use rate of inflation to ajust all of them to 2010.
- Created two new columns: more important than population and number of homes, it’s the density in each area.
  - People_per_area
  - Home_per_area

Clean and Format data

- Get the 10th most frequent venue categories in each neighborhood, based on Foursquare API.

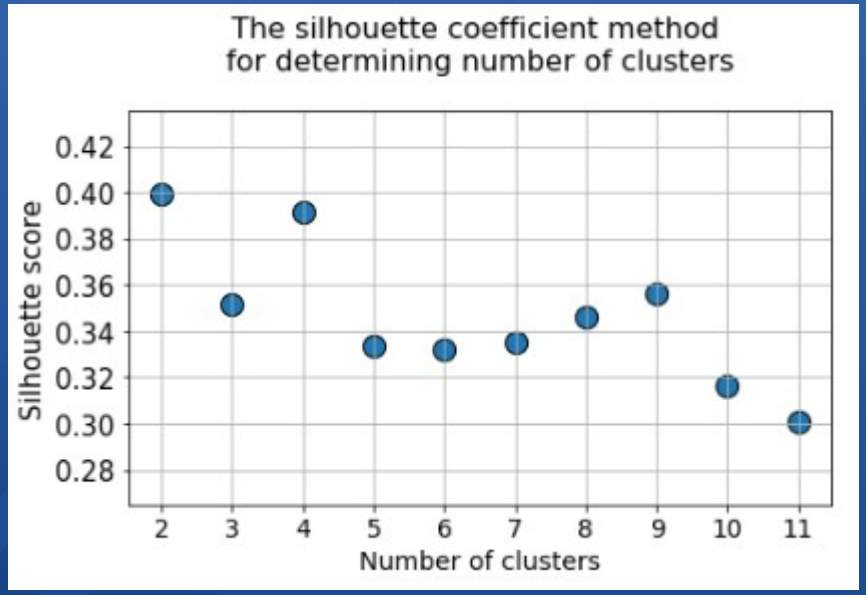



```
neighborhoods_venues_sorted.head()
```

	Neighborhood	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 Most Common Venue
0	Abranches	Soccer Field	Grocery Store	Park	Gym / Fitness Center	Juice Bar	Bakery	Fishing Spot	Fish Market	Fish & Chips Shop	Fast Food Restaurant
1	Ahú	Soccer Field	Pizza Place	Brazilian Restaurant	Food Truck	Restaurant	Dessert Shop	Pet Store	Gym / Fitness Center	Italian Restaurant	Steakhouse
2	Alto Boqueirão	Pizza Place	Bakery	Grocery Store	Market	Diner	Gym	Electronics Store	Lottery Retailer	Supermarket	Brazilian Restaurant
3	Alto da Glória	Brazilian Restaurant	Pizza Place	Italian Restaurant	Coffee Shop	Middle Eastern Restaurant	Café	Gym / Fitness Center	Hot Dog Joint	Bar	Asian Restaurant
4	Alto da Rua XV	Bar	Restaurant	Italian Restaurant	Dessert Shop	Coffee Shop	Café	Brazilian Restaurant	Ice Cream Shop	Gym / Fitness Center	Gym

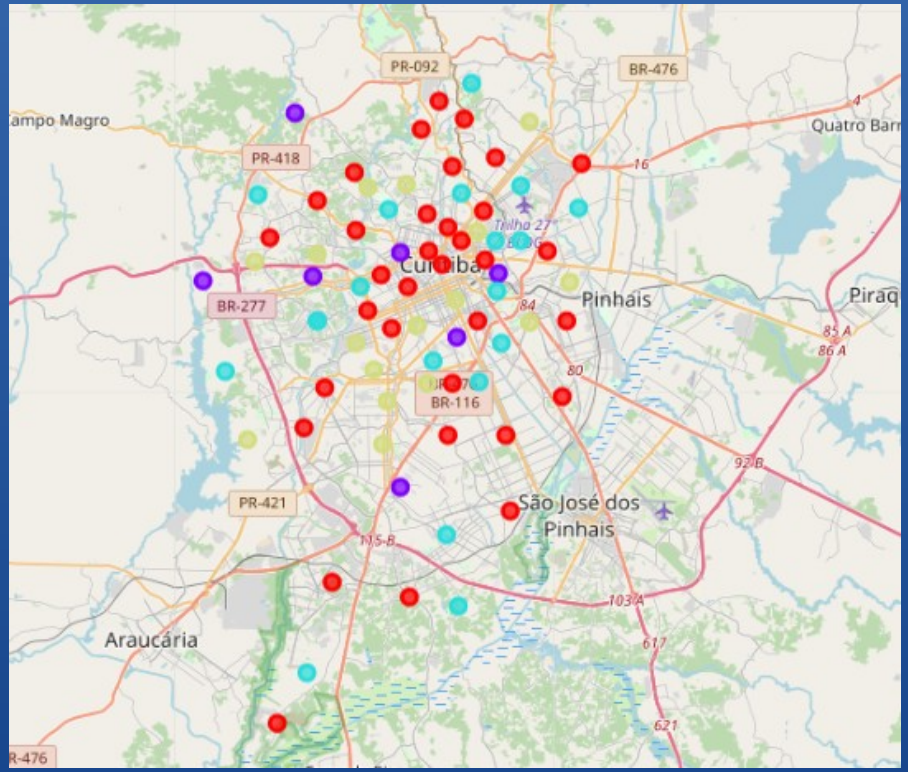
Clustering Algorithm - k-means

1. Choose the best k. It should be k=2, but for the purpose of the business, I've chosen k=4.



Clustering Algorithm - k-means

2. Visualize the generatde clusters.



Calculate the distance between neighborhoods

```
origin = (lat_new_job, long_new_job)

list_dist = []

lat_list = city_merged['latitude']
long_list = city_merged['longitude']

for i in range(len(lat_list)):
    dest = (lat_list[i], long_list[i])
    list_dist.append(geodesic(origin, dest).meters)

city_merged['distance'] = list_dist
city_merged
```

Get the recommended neighborhoods

INPUT

2. Define location variables

Here, we define the coordinates (latitude and

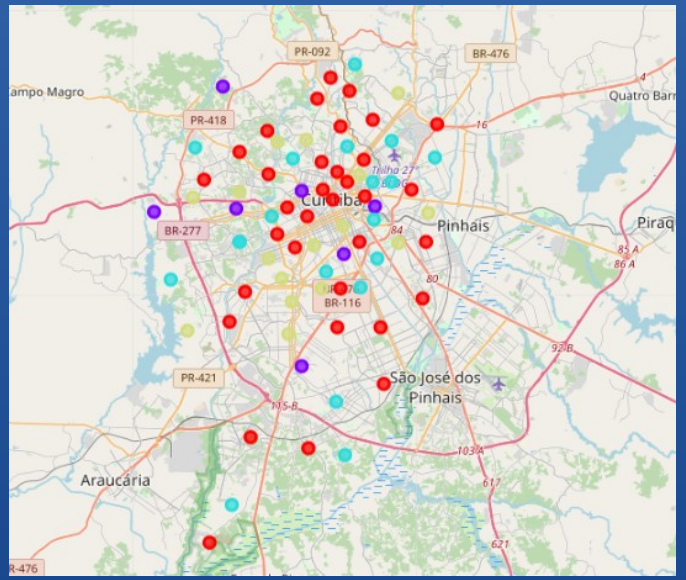
```
# Coordinates from the city
lat_city = -25.480876
lon_city = -49.304425

# Coordinates from my current home
neigh_current_home = 'Cristo Rei'
lat_current_home = -25.433222
lon_current_home = -49.2519034

# Coordinates from my new job
neigh_new_job = 'Água Verde'
lat_new_job = -25.4598405
long_new_job = -49.2739856

# The city and state string is impor
rary.
city_state_str = 'Curitiba-PR'
```

OUTPUT




```
0             Parolin
1         Cristo Rei
2              Mercês
3          Mossunguê
4        Pinheirinho
5             Riviera
6    Lamenha Pequena
Name: neighborhood, dtype: object
```