Projeto Foursquare

Cassino em Belo Horizonte - MG

Lucas Parreiras de Sousa
Instituto INFNET

Oultline

- Sumário
- Introdução
- Metodologia
- Resultados
- Conclusão

Sumário

Sumário de metodologias:

- Coleta de dados via Foursquare API
- Tratamento e transformação dos dados
- Visualização dos dados com gráfico e mapas
- Clusterização usando K-Means

Sumário dos resultados:

- Resultados da análise exploratória
- Resultados da clusterização

Introdução

- Contexto e motivações:
  - Na primeira quinzena de abril de 2022, foi aprovado um projeto de lei (PL 442/91) que propõe a legalizção de cassinos e jogos de azar no Brasil. Eu, como estudante de Data Science, analisarei alguns bairros que sei que são movimentados e atrativos, na cidade de Belo horizonte, e vou agrupá-los com uma técnica de clusterização para saber qual região da cidade seria interessante abrir um cassino. O agrupamento será feito com base no número de bares, restaurantes, boates e hotéis nas proximidades dos bairros.
- Problemas
  - Identificar quais são os bairros que possuem mais hotéis e vida noturna ativa (boates, bares e restaurantes)
  - Verificar se esses bairros também são geograficamente próximos
  - Plotar gráficos mostrando os lugares encontrados e os clusters criados

Metodologia

Coleta de dados

1. Foursquare API

Transformação de dados

1. Colunas criadas a partir de dicionário com categorias dos lugares próximos
2. Exclusão de lugares repetidos
3. Generalização de todas as categorias em apenas quatro (bares, restaurantes, boates e hotéis)
4. Criação de variáveis dummy a partir das 3 colunas de categorias, para contagem de lugares por bairro

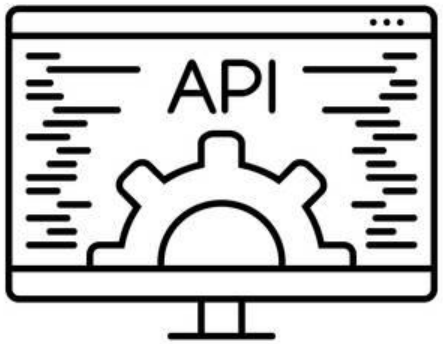
Plot de gráficos para visualização

1. Mapa de BH com lugares encontrados
2. Gráfico 'Stacked Bar' para mostrar quantidade de lugares encontrados por bairro

Criação do modelo de clusterização:

1. Método do cotovelo e silhueta para decidir número de K
2. Criação de modelo K Means com k=5

Coleta de dados



Fazer *request* usando API da Foursquare

API retorna dados em formato .json

Normaliza o arquivo .json e transforma em um *dataframe*

Foursquare API

Como a API do Foursquare só consegue buscar locais próximos a partir de uma dada coordenada (lat/long), foi necessário criar um *dataframe 'dados'* que possui os bairros escolhidos e suas coordenadas retiradas manualmente no Google Maps

Os bairros foram definidos a partir de uma lista encontrada na internet com os 10 melhores bairros para se morar e outros bairros, que sei que também são requisitados, foram incluídos para a base. A grande maioria desse bairros fica na região Centro-sul de Belo Horizonte

```
dados = pd.DataFrame(
        [('Anchieta', -19.949453323219245, -43.92700801773141),
         ('Belvedere', -19.972559218057427, -43.93718789026611),
         ('Centro', -19.919279099643592, -43.939315609652134),
         ('Cidade Jardim', -19.938641684001976, -43.950640230590984),
         ('Cidade Nova', -19.890068184185246, -43.924772018135),
         ('Castelo', -19.88213333487584, -43.999206085545126),
         ('Gutierrez', -19.93678462997389, -43.95771548553014),
         ('Buritis', -19.97563516763052, -43.96828714846663),
         ('Prado', -19.922907168023617, -43.960709744404966),
         ('Lourdes', -19.93049620714053, -43.94296904336832),
         ('Vila da Serra', -19.9546793, -43.9154242),
         ('Floresta', -19.916406188338616, -43.92885909887027),
         ('Funcionários', -19.931827768132536, -43.92856347797582),
         ('Ouro Preto', -19.8736478717947, -43.98513618425674),
         ('Santa Tereza', -19.91578247666307, -43.91675236069356),
         ('Savassi', -19.93614741061873, -43.93349802745276),
         ('Santo Antônio', -19.94462581113818, -43.94392639104507),
         ('Santo Agostinho', -19.92917086904201, -43.95068392066596),
         ('Santa Efigênia', -19.923409948392948, -43.92061174310462),
         ('Sion', -19.953112178597326, -43.934422374007475)], columns=['Bairro', 'Lat', 'Lng'])
```

Foursquare API

Foi criada então uma função que passaria por todo o *dataframe* criado e retornaria os locais próximos. A função foi então aplicada ao *dataframe* 'dados'

```
[ ]  def getNearbyVenues(names, latitudes, longitudes):

       '''
         Função para conectar com a API do Foursquare, obter os dados
         em formato json e tranformá-los em um dataframe
         '''

       df = pd.DataFrame()
       bairros = []

       for name, lat, lng in zip(names, latitudes, longitudes):

         # Criando o Request
         url = f'https://api.foursquare.com/v3/places/search?ll={lat},{lng}&radius={radius}&categories={cat_casino}&limit={LIMIT}'

         headers = {'Accept': 'application/json',
                'Authorization': 'fsq3AWxWY/26wzcU9YHkerFRNQXlWkwHBUsh8md+NYR8AJE='}

         # Making get request
         response = requests.get(url, headers=headers)
         json = response.json()

         df_mult = pd.json_normalize(json['results'])
         df_mult['Bairro'] = name
         df = pd.concat([df, df_mult])

       df.reset_index(inplace=True, drop=True)
       return(df)
```

Aplicando a função nos nossos dados dos bairros

```
[ ]  df = getNearbyVenues(dados['Bairro'], dados['Lat'], dados['Lng'])
```

Tratamento e transformação

- Os dados .json recebidos retornavam as categorias dos lugares em um formato de dicionário, em que cada lugar pode ter até 3 categorias diferentes
- Foi criada uma coluna para categorias que cada registro recebia uma lista com todas as categorias que fazia parte e, a partir dessa listas, foram criadas 3 colunas para as 3 categorias. Lugares que não possuíam 3 categorias associadas recebem um NaN nas colunas 2 e/ou 3
- Os lugares podem aparecer mais de uma vez devido a proximidade dos bairros e, como o fator geográfico já vais ser levado em consideração no final, vamos excluir os duplicados.
- Das categorias que a API retornou, agrupamos todas em 4 grupos e transformamos seus valores para: bar, hotel, restaurante ou boate
- A partir das 3 colunas criadas para as categorias, foi feito manualmente a transformação para variáveis dummy e sem repetir categorias dentro do mesmo lugar

Visualização dos dados

- Criaçaõ do mapa de Belo Horizonte usando o Folium
  - Marcação no mapa de todos estabelecimentos encontrados
- Stacked Bar Chart
  - Gráfico para mostrar, por bairros, o número total de cada tipo de estabelecimento
- Criação de gráficos usando yellowbrick para definir qual melhor K para a técnica do K Means

```
def plotMap(lat, lng, label, map, color, radius):
  for lat, lng, label in zip(lat, lng, label):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=radius,
        color=color,
        fill=True,
        fill_opacity=0.5,
        parse_html=False).add_to(map)

  return map
```

Clusterização com K Means

Criando o modelo

- Depois de definir o k (=5) com os gráficos na etapa anterior, é criado o modelo e aplicado no nosso *dataframe* agrupado por bairros

Identificamos a melhor zona para se abrir um cassino em Belo Horizonte

Resultados

- Resultados da análise exploratória
- Resultados da clusterização

Análise Exploratória e transformação de Dados

Pre processamento dos dados

Extraindo as categorias do dicionário entes de salvar a base em um csv

As categorias são um dado importante, precisamos extrair da coluna 'categories'.
A coluna está em um formato de dicionário e cada registro pode ter até 3 categorias. Vou criar um lista em que cada registro possui uma lista com sua categorias.

```
[ ]  items=list()
     for venue in df['categories']:
       cat=list()
       for c in range(len(venue)):
         cat.append(venue[c]['name'])
       items.append(cat)

     items
```

Criando colunas para as categorias

Serão criadas 3 novas colunas que serão preenchidas com as categorias de cada registro. Os locais que possuem 1 ou 2 categorias apenas, receberão um 'None' no lugar.

```
[ ]  categ = pd.DataFrame(items, index= df.index, columns=['Cat1','Cat2','Cat3'])
     categ.head()
```

	Cat1	Cat2	Cat3
0	Arts and Entertainment	Cocktail Bar	Buffet
1	Beer Bar	Gastropub	None
2	Hotel	None	None
3	Pub	None	None
4	Arts and Entertainment	Pub	Restaurant

Análise Exploratória e transformação de Dados

```
categories = []
for col in ['Cat1', 'Cat2', 'Cat3']:
  var=[]
  for row in df[col]:
    if pd.isnull(row):
      var.append(row)
    else:
      for k, v in cats.items():
        if row in v:
          var.append(k)

  categories.append(var)
```

```
[ ] df_categ = pd.DataFrame(data=categories, index=['gcat_1', 'gcat_2', 'gcat_3']).T
```

```
[ ] df_categ
```

	gcat_1	gcat_2	gcat_3
0	bar	bar	restaurant
1	bar	bar	NaN
2	hotel	NaN	NaN
3	bar	NaN	NaN
4	bar	bar	restaurant
...	...	...	...
459	bar	NaN	NaN
460	nightclub	NaN	NaN
461	hotel	hotel	NaN
462	hotel	hotel	NaN
463	hotel	hotel	NaN

464 rows × 3 columns

Transformação de todas as categorias em apenas 4 categorias. Nota-se que alguns registros ficam com categorias repetidas

```
for index, row in df_categ.iterrows():
  test = []
  for val in row:
    test.append(val)
  if test[0] == test[1]:
    row[1] = None
  elif test[0] == test[2]:
    row[2] == None
  elif test[1] == test[2]:
    row[2] == None

df_categ.head(3)
```

	gcat_1	gcat_2	gcat_3
0	bar	None	restaurant
1	bar	None	NaN
2	hotel	NaN	NaN

Retirando as repetidas

Análise Exploratória e transformação de Dados

Vamos criar 4 colunas, uma para cada categoria, e colocar o valor 1 nas colunas que categorizam o local (dummy)

```
df_categ['bar'] = np.nan
df_categ['hotel'] = np.nan
df_categ['restaurant'] = np.nan
df_categ['nightclub'] = np.nan

for idx, row in df_categ.iterrows():
  for c in range(3):
    for v in ['bar', 'hotel', 'restaurant', 'nightclub']:
      if row[c] == v:
        df_categ[v].iloc[idx] = 1
```

```
new_df = pd.concat([df[['name', 'Bairro', 'geocodes.main.latitude', 'geocodes.main.longitude']], df_categ[['bar','hotel', 'restaurant', 'nightclub']]], axis=1).fillna(0)
new_df.tail()
```

	name	Bairro	geocodes.main.latitude	geocodes.main.longitude	bar	hotel	restaurant	nightclub
459	Growler Station Wals	Sion	-19.954426	-43.940194	1.0	0.0	0.0	0.0
460	DJ Babi	Sion	-19.957421	-43.940044	0.0	0.0	0.0	1.0
461	Hotel Rural Canto das Cachoeiras Ltda	Sion	-19.953583	-43.939009	0.0	1.0	0.0	0.0
462	Apartamento do Escultor	Sion	-19.956793	-43.937526	0.0	1.0	0.0	0.0
463	Pousada Vila do Beco	Sion	-19.946842	-43.940655	0.0	1.0	0.0	0.0

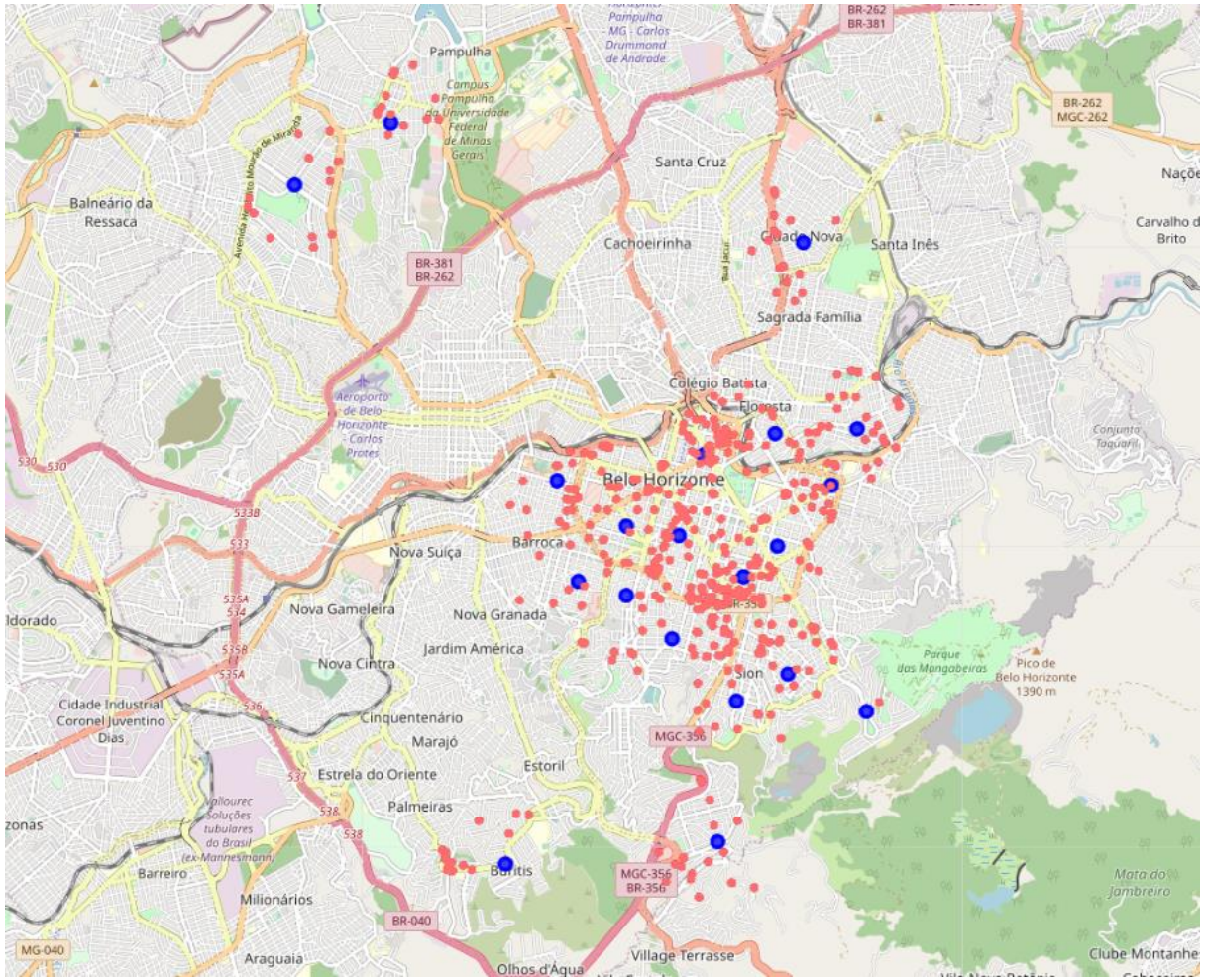
Análise Exploratória e transformação de Dados

Utilizando as colunas *dummy* fazemos a contagem dos locais encontrados agrupado pelos bairros

Bairro	bar	restaurant	hotel	nightclub
Anchieta	26.0	7.0	13.0	5.0
Belvedere	9.0	4.0	6.0	4.0
Buritis	13.0	9.0	1.0	3.0
Castelo	10.0	3.0	3.0	0.0
Centro	15.0	6.0	33.0	3.0
Cidade Jardim	17.0	6.0	22.0	10.0
Cidade Nova	10.0	1.0	5.0	2.0
Floresta	16.0	8.0	18.0	13.0
Funcionários	22.0	10.0	15.0	10.0
Gutierrez	3.0	0.0	4.0	2.0
Lourdes	6.0	3.0	19.0	4.0
Ouro Preto	11.0	5.0	4.0	1.0
Prado	10.0	3.0	23.0	5.0
Santa Efigênia	7.0	3.0	5.0	4.0
Santa Tereza	18.0	9.0	3.0	2.0
Santo Agostinho	2.0	0.0	7.0	3.0
Santo Antônio	11.0	3.0	9.0	2.0
Savassi	12.0	9.0	4.0	3.0
Sion	1.0	0.0	4.0	1.0
Vila da Serra	0.0	0.0	1.0	0.0

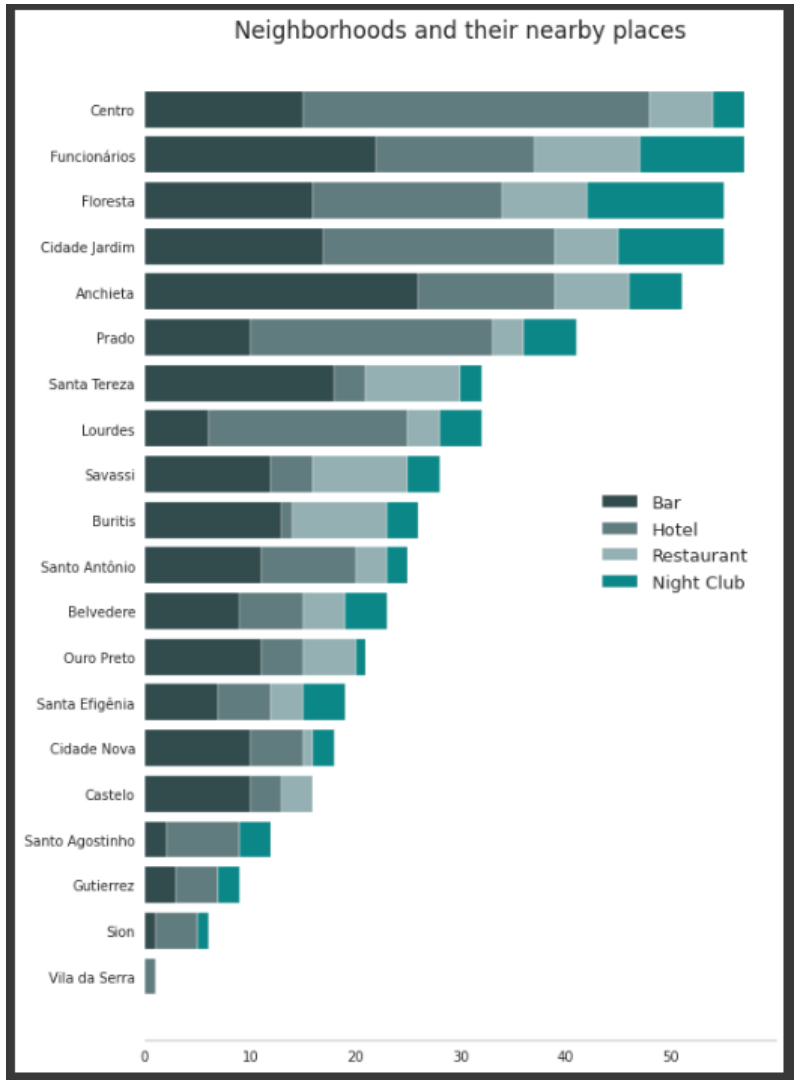
Já podemos ver que vários bairros da região centro-sul possuem muitos estabelecimentos, como era esperado.

Visualização dos dados



Mapa de Belo Horizonte com os bairros marcados de azul e todos os lugares encontrados de vermelho

Visualização dos dados



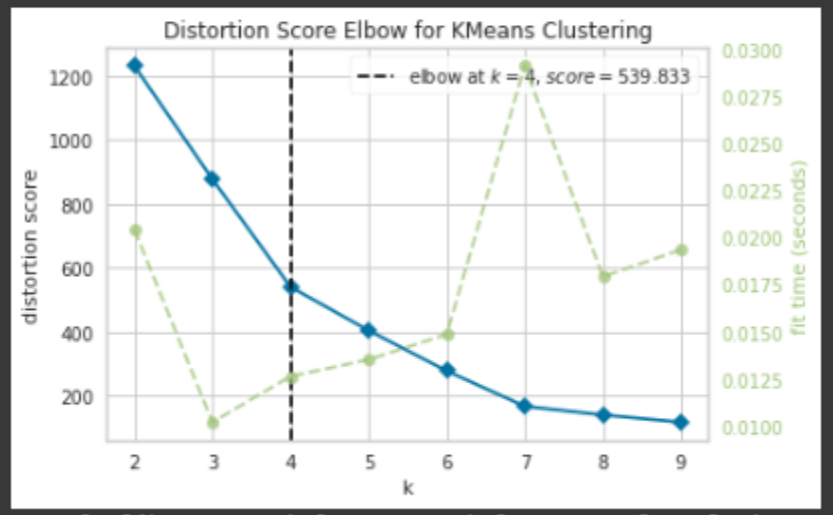
Stacked bar dos bairros e estabelecimentos perto

Criação do modelo

Definindo valor de K

Método do cotovelo:

```
kmeans = KMeans()
yellow_visualizer = KElbowVisualizer(kmeans, k=(2, 10))
yellow_visualizer.fit(bairros_dummy)
yellow_visualizer.show()
```



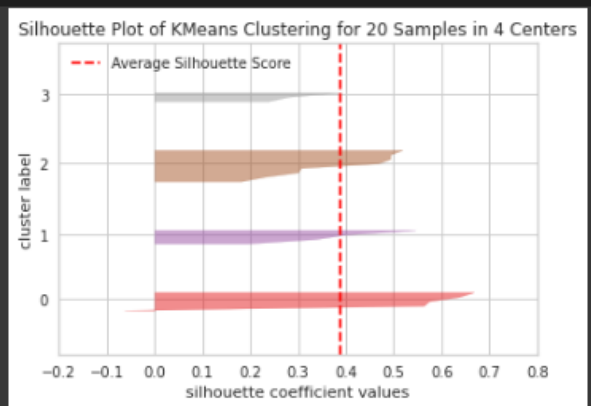

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c10fe0d90>
```

Criação do modelo

Definindo valor de K

Método da silhueta:

```
[ ]  for i in range(2,10):
       kmeans = KMeans(n_clusters=i)
       yellow_visualizer = SilhouetteVisualizer(kmeans)
       yellow_visualizer.fit(bairros_dummy)
       yellow_visualizer.show()
```



O valor de k foi escolhido para ser 5 e o grupo 1 será considerado um outlier

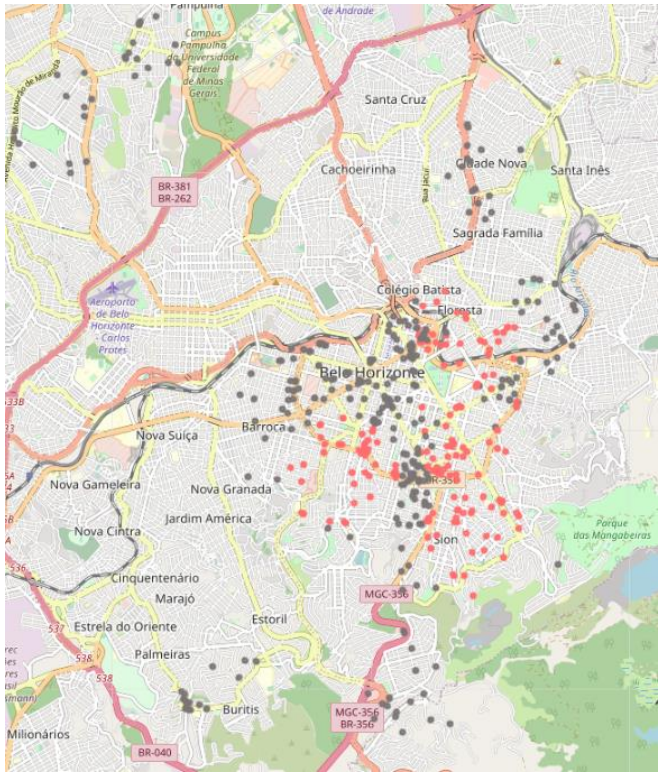
Dataframe com uma coluna identificando os clusters

```
[ ]  bairros_dummy.reset_index(inplace=True)
     bairros_dummy.head()
```

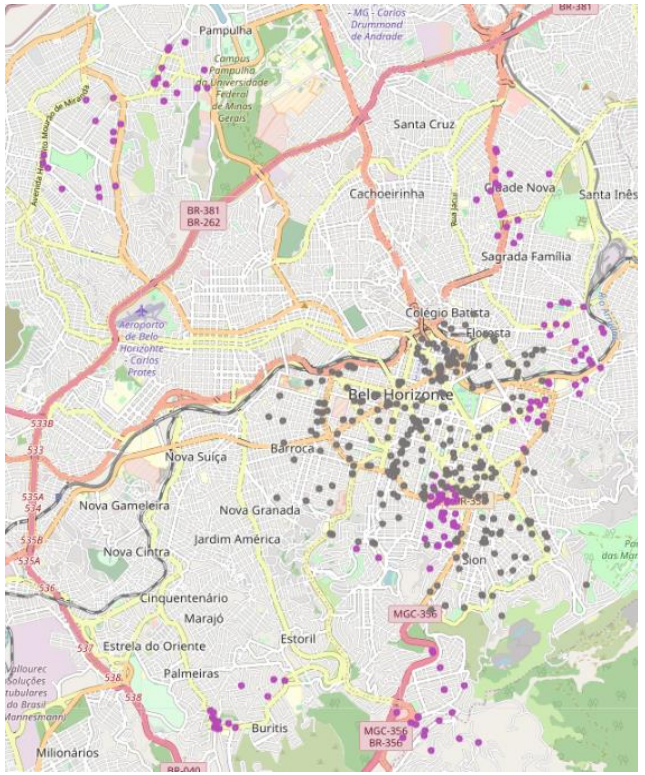
	Bairro	bar	restaurant	hotel	nightclub	Cluster
0	Anchieta	26.0	7.0	13.0	5.0	1
1	Belvedere	9.0	4.0	6.0	4.0	0
2	Buritis	13.0	9.0	1.0	3.0	0
3	Castelo	10.0	3.0	3.0	0.0	0
4	Centro	15.0	6.0	33.0	3.0	4

Resultados

Cluster 1



Cluster 2



Conclusão

- Temos 2 clusters que mostraram ter uma boa vida noturna e hotéis nas proximidades, foram eles:
- Cluster 1: Anchieta, Cidade Jardim, Floresta, Funcionários
- Cluster 2: Buritis, Castelo, Cidade Nova, Ouro Preto, Santa Efigênia, Santa Tereza, Santo Antônio, Savassi
- Como o **Cluster 1** tem uma média muito alta e é composto por bairros bem próximos, **foi decidido que naquela região seria o melhor local para se abrir um cassino em Belo Horizonte**. Além disso, o Cluster 2 possui alguns bairros que estão próximos do Cluster 1, assim como o Centro (nosso outlier, mas que possui um número bem altos de estabelecimentos perto), o que ainda contribuiria no elevado movimento do local/região.

Obrigado!