Applied Data Science Capstone - Fado, the sound of saudades.

Introduction and business problem

Fado is the sound of Lisbon. This characteristic style of Portuguese music is part of the UNESCO Intangible Cultural Heritage List, and no trip to Lisbon is complete without a visit to a Fado show. Traditionally, the neighbourhood of Alfama is home to most of the Fado Venues in Lisbon, and walking through its narrow streets one is often greeted by the haunting voice of the many Fadistas who grace the neighbourhood with their voices. However, while tradition is very important to the Portuguese, the recent COVID pandemic necessitates a rethinking of the commercial potential of Fado. Tourism has long been one of Portugal's main sources of income, and my client would like to open a cultural centre to capitalise on the tourism industry and to showcase some of the best talent on the Fado scene in a single location. Given Lisbon's very hilly topography, my client is looking for a cultural and touristic hotspot, where a potential client wouldn't have to walk more than 250 meters in any direction to find a choice of several Fado venues and hotels. In order to do this, I will map the hotels and Fado venues in the centre of Lisbon using their geographical coordinates, and use a clustering algorithm to find the cultural and tourist hotspots for my clients venture.

Data Section

The data I will use to solve this problem are:

Geolocation and names of the of the Fado venues in Lisbon.

Geolocation, names and types of the accommodation available in Libson.

I will source my data using the following:

Foursquare API for the accommodation data.

Google Places API for the Fado venues data.

Methodology

Firstly, I went about installing the libraries and resources I would use for the project, and I added more as I progressed.

The first datapoints I required were the geolocation coordinates of the centre of Lisbon, which I obtained using Nominatim.

```
In [2]: ## Getting the geolocation of Lisbon

address = 'Lisbon, Portugal'
geoloc = Nominatim(user_agent = "qferreiraclinpsych@gmail.com")
location = geoloc.geocode(address)
latitude = location.latitude
longitude = location.longitude

print("The coordinates of Lisbon are: ",latitude,",", longitude)


The coordinates of Lisbon are:  38.7077507 , -9.1365919
```

Once I had these, I executed a hotel query with the Foursquare API to return hotels near the city centre directly, and it returned a JSON file with several nested dictionaries and lists further nested within a dictionary. I decided the quickest way to create a dataframe would be to iterate through this data and extract the relevant values (Hotel_Name, Type_of_Accomodation, Latitude, Longitude) from the nested dictionaries/lists, create one list per column and finally merge them into a single dataframe. I wrangled this data to exclude hostels, boarding houses and an instance of a hotel pool which appeared during the search. I also converted the coordinates into the datatype “float64”, so that they could be mapped Once I was satisfied with the format of the data, I used a Folium map to plot the hotels onto a map of Lisbon.

The Foursquare API query:

```
In [3]: ## Foursquare API to find venues and hotels

        CLIENT_ID = [redacted]
        CLIENT_SECRET
        VERSION = '20180604'
        LIMIT = 1000
        RADIUS = 1000

        foursq1 = 'https://api.foursquare.com/v2/venues/search?categoryId=4bf58dd8d4898
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            latitude,
            longitude,
            RADIUS,
            LIMIT)

        foursq1

Out[3]: 'https://api.foursquare.com/v2/venues/search?categoryId=4bf58dd8d48988d1fa93
        1735&client_id=[redacted]&client_secre
        t=[redacted]v=20180604&ll=38.707750
        7,-9.1365919&radius=1000&limit=1000'

In [4]: ##loading the Json File

        response = requests.get(foursq1)
        fsqjs = json.loads(response.text)

        fsqjs
```

An example of the iteration to retrieve the latitude and longitude values:

```
In [9]: hotel_lat_list = []
        hotel_lon_list = []

        q1 = fsqjs['response']['venues']
        x = 0
        while x < len(q1):
            vn = q1[x]
            hotel_lat_list.append(vn['location']['lat'])
            hotel_lon_list.append(vn['location']['lng'])
            x += 1

        # Printing the lengths of the lists to make sure they are equal.
        print(len(hotel_lat_list),len(hotel_lon_list))

        48 48
```

Joining the lists into a dataframe:

```
In [10]: ## joining my lists into a dataframe.

hotel_df = pd.DataFrame(np.column_stack([hotel_list, venue_type_list, hotel_lat_list, hotel_lon_list]),
                                columns=['Hotel_Name', 'Type_of_Accomodation', 'Latitude', 'Longitude'])
```

Part of the hotel dataframe:

	Hotel_Name	Type_of_Accomodation	Latitude	Longitude
2	Rossio Boutique Hotel	Hotel	38.714950	-9.138443
3	Porta Do Mar	Residential Building (Apartment / Condo)	38.709100	-9.134064
4	My Story Hotel Figueira	Hotel	38.713847	-9.137228
5	Pensão Estação Central	Hotel	38.713466	-9.140095
6	The Lift Boutique Hotel	Hotel	38.712335	-9.139303

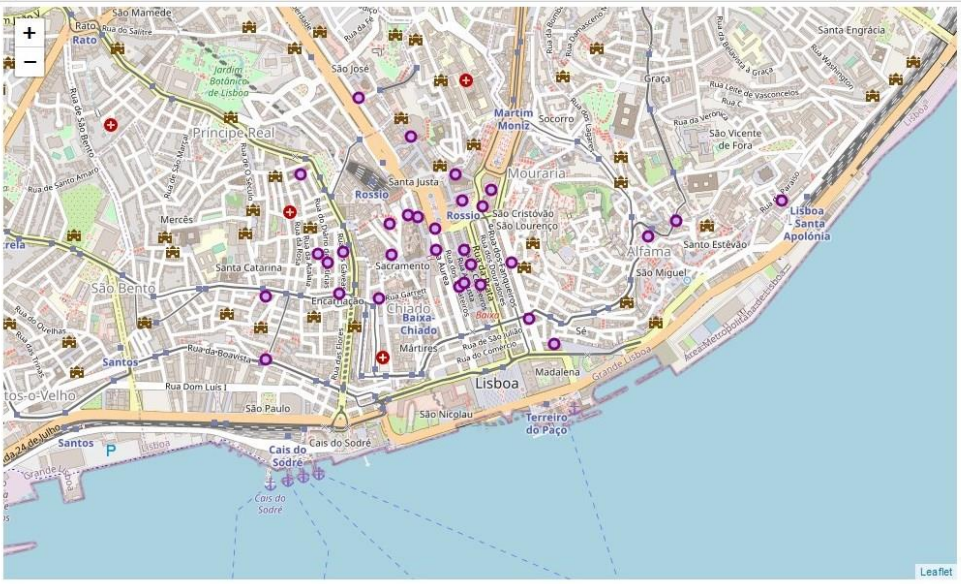
:

Mapping the hotels:

```
In [14]: ## mapping the hotels in Lisbon

Lis_hotel_map = folium.Map(location=[latitude, longitude], zoom_start=12)

for lat, lng, hotel, hot_type in zip(hotel_df['Latitude'], hotel_df['Longitude'], hotel_df['Hotel_Name'], hotel_df['Type_of_Ac
    label = '{}, {}'.format(hotel, hot_type)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='purple',
        fill=True,
        fill_color='#c8b1e5',
        fill_opacity=1.5,
        ).add_to(Lis_hotel_map)

Lis_hotel_map
```

The map of hotels:

Out[14]:



I then executed a query on the Google Places API to find the Fado venues in Lisbon, and it returned a JSON file with nested lists of the data. As before, I iterated through the lists to retrieve the venue name, latitude and longitude, and combined these lists into a single dataframe as shown below:

In [19]:
```
## Creating lists of the name, latitude and longitude of the Fado venues

fado_name_l = []
fado_lat_l = []
fado_lon_l = []
i = 0
while i < len(res_g):
    fado_l = res_g[i]
    fado_name_l.append(fado_l['name'])
    fado_lat_l.append(fado_l['geometry']['location']['lat'])
    fado_lon_l.append(fado_l['geometry']['location']['lng'])
    i += 1
```

In [20]:
```
# Printing the lengths of the lists to make sure they are equal.

print(len(fado_name_l),len(fado_lat_l),len(fado_lon_l))
```

```
20 20 20
```

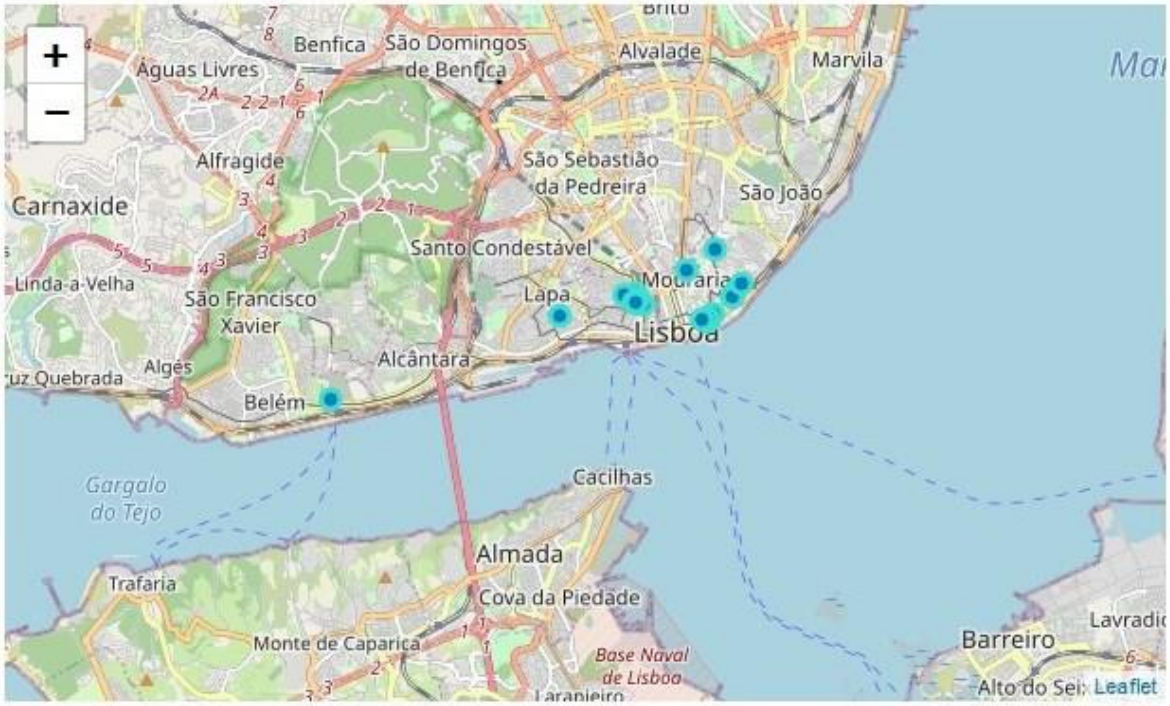
In [21]:
```
## creating a dataframe of Fado venues.

fado_df = pd.DataFrame(np.column_stack([fado_name_l, fado_lat_l, fado_lon_l]),
                                columns=['Fado_Venue_Name', 'Latitude', 'Longitu

fado_df.head(20)
```

The Fado Dataframe:

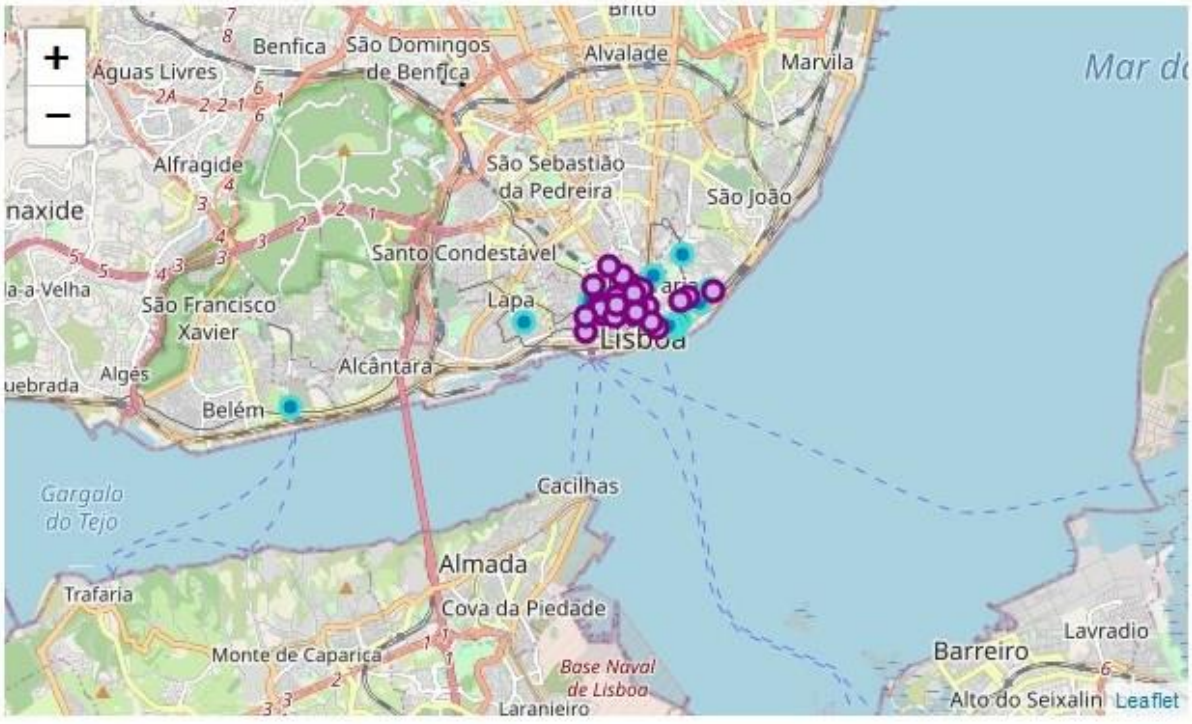
Out[21]:

	Fado_Venue_Name	Latitude	Longitude
0	Fado in Chiado | Fado in Lisbon	38.711422	-9.1428429
1	Adega Machado - Fado & Food Group	38.712294	-9.143813
2	Mesa De Frades	38.7131006	-9.1260757
3	Páteo de Alfama	38.7102667	-9.129925
4	Café Luso - Fado & Food Group	38.7130556	-9.1438889
5	Tasca do Chico	38.7116451	-9.1441389
6	Live Fado	38.7122435	-9.1437096
7	Casa de Linhares - FADO	38.7091595	-9.130906
8	Clube de Fado - Fado & Food Group	38.7097222	-9.1311111
9	Duetos da Sé	38.7093017	-9.132198899999999
10	Fado in Inglesinhos	38.71284259999999	-9.1462571
11	A Tasca do Jaime	38.7191667	-9.1297222
12	A Viela do Fado	38.712649	-9.126433
13	Maria da Mouraria - Casa de Fados	38.7162605	-9.134675999999999
14	Real Fado	38.6981757	-9.1993681
15	Sr.Vinho	38.71	-9.1577778
16	O Faia	38.7117375	-9.1444649
17	Retiro Dos Sentidos	38.711672	-9.144063
18	Adega dos Fadistas	38.7124759	-9.126415699999999
19	O CORRIDO, restaurante - Casa de Fado	38.7144729	-9.124735300000001

Once again, I converted the latitude and longitude datapoints into the datatype “float64” and mapped them onto the map of Lisbon:



I then produced a map with the hotels and Fado venues shown alongside:



Because I am looking for clusters based on distance, I decided to use the DBSCAN algorithm as this allows one to input latitude and longitude data and use the haversine distance between points to form the clusters.

Firstly, I had to combine the latitude and longitudes within my dataframes into a 2D array, as this is the required format for applying the DBSCAN algorithm to the type of data I am inputting:

```
In [25]: ## Now I am going to convert my latitudes and longitudes to a 2D array, so we c

         ha1 = hotel_df[['Latitude', 'Longitude']]
         fa1 = fado_df[['Latitude','Longitude']]

         merged_dfs = pd.concat([ha1, fa1])

         merged_dfs.reset_index(drop=True, inplace = True)
         merged_dfs

         hotfad1 = merged_dfs['Latitude']
         hotfad2 = merged_dfs['Longitude']

         # I need a 2D array to work with DBSCAN, so I stack the columns

         hotfad = np.column_stack((hotfad1,hotfad2))

         hotfad
```

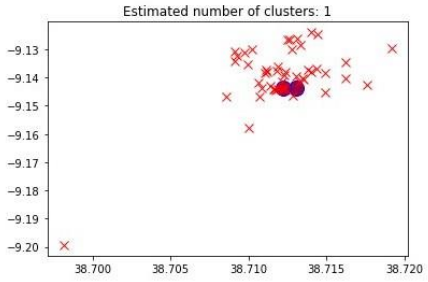
Then I used DBSCAN on these datapoints, to find which venues could form part of a cluster from where you could reach 14 other venues (hence min_samples = 15.)

Because I was using latitude and longitude coordinates with real world distance in metres, I used numpy radians when I fit the model and set the eps parameter to 0.25/6371 which produced the distance of 250m in radians. I also used the ball-tree algorithm because I am working within a 2D space.

```
In [26]: ## running the DBSCAN algorithm to cluster based on 15 venues within 250 meters of each other.

         db = DBSCAN(eps=0.25/6371., min_samples=15, algorithm='ball_tree', metric='haversine').fit(np.radians(hotfad))

In [27]: cluster_labels = db.labels_
         num_clusters = len(set(cluster_labels))
         clusters = pd.Series([hotfad[cluster_labels == n] for n in range(num_clusters)])
         print('Number of clusters: {}'.format(num_clusters))

         Number of clusters: 2

In [28]: cluster_labels

Out[28]: array([-1, -1, -1, -1, -1,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0, -1,
                 0,  0, -1, -1, -1,  0,  0, -1, -1, -1, -1,  0, -1, -1,  0,  0,  0, -1,
                -1,  0,  0,  0, -1, -1, -1,  0, -1, -1, -1, -1, -1,  0,  0, -1, -1],
               dtype=int64)
```

I then plotted my results, you will see the outliers represented as ‘x’ below:

In [29]:
```
## plotting the clusters, and checking for outliers

core_samples_mask = np.zeros_like(cluster_labels, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
n_clusters_ = len(set(cluster_labels)) - (1 if -1 in cluster_labels else 0)
n_noise_ = list(cluster_labels).count(-1)

unique_labels = set(cluster_labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]

    class_member_mask = (cluster_labels == k)

    xy = hotfad[class_member_mask & core_samples_mask] ## These are the clustered datapoints
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='b', markersize=14)

    xx = hotfad[class_member_mask & ~core_samples_mask] ## These are the outliers which will be excluded
    plt.plot(xx[:, 0], xx[:, 1], 'x', markerfacecolor=tuple(col),
             markeredgecolor='r', markersize=8)

plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```



I then created a dataframe with the coordinates of my outliers so that I could exclude them from the final dataframe, from which I would plot the heatmap showing the cultural and tourism hotspot.

In [30]:
```
## this will be my dataframe of outliers for exclusion in the heatmap

excl_df = pd.DataFrame(xx, columns=['Latitude', 'Longitude'])

excl_df
```

Out[30]:

Using an outer join to exclude the outlier latitude and longitudes from a dataframe of all the latitudes and longitudes examined, and making a list for the heatmap:

In [31]:
```
# A dataframe of the latitude and longitudes of my merged Fado and Hotel Dataframes

merged_lat_lon = merged_dfs[['Latitude','Longitude']]
```

In [38]:
```
## The dataframe excluding the outlying venues

final_df = merged_lat_lon.merge(excl_df, how = 'outer', indicator =True).loc[lambda x: x['_merge']=='left_only']
```

In [39]:
```
## a list for my heatmap

hm_data = [[row['Latitude'],row['Longitude']] for index, row in final_df.iterrows()]
```

Lastly, I plotted all the hotels and Fado venues, onto a Folium heatmap showing the venues that fall into the cultural and tourism hotspot in Lisbon:

In [40]:
```
## lastly, creating a heatmap where the heat signifies the venues that fall into a density cluster of 10 venues per 250m

center = latitude,longitude
hm_lisbon = folium.Map(location = center, zoom_start=15)

for lat, lng, fado_n, in zip(fado_df['Latitude'], fado_df['Longitude'], fado_df['Fado_Venue_Name']):
    label = fado_n
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='turquoise',
        fill=True,
        fill_color='#0082c4',
        fill_opacity=1.5,
        ).add_to(hm_lisbon)
for lat, lng, hotel, hot_type in zip(hotel_df['Latitude'], hotel_df['Longitude'], hotel_df['Hotel_Name'], hotel_df['Type_of_Ac
    label = '{}, {}'.format(hotel, hot_type)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='purple',
        fill=True,
        fill_color='#c8b1e5',
        fill_opacity=1.5,
        ).add_to(hm_lisbon)

folium.TileLayer('cartodbpositron').add_to(hm_lisbon)
folium.Marker(center, popup='Lisbon Centre').add_to(hm_lisbon)
HeatMap(hm_data).add_to(hm_lisbon)

hm_lisbon
```

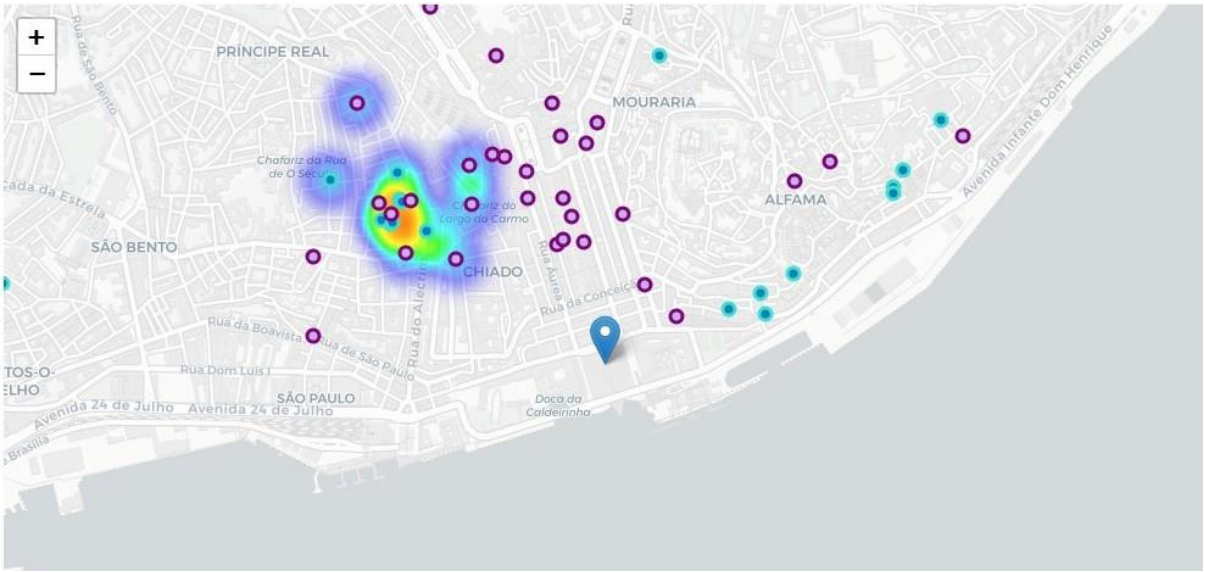
Results

Based on the heatmap produced and shown below, several interesting observations can be made.

Firstly, being familiar with Lisbon I expected to see a large concentration of Fado districts in the Alfama neighbourhood, but indeed there are few hotels located there as it is mostly a residential neighbourhood. Furthermore, it was unsurprising that many of the hotels are located just North of the centre of Lisbon (known as the Praça do Comércio.)

What is clear from the analysis performed is that the cultural and touristic hotspot for Fado lies neither in the most famous Fado district, nor in the centre of the city itself where a majority of the hotels are located. Instead, it lies at the junction between the

neighbourhoods of Santa Catarina, Encarnaçao and Sacramento, where a person can find numerous Fado venues and Hotels within a short walk from the centre of the hotspot (shown in red/orange).



Discussion

Based on the geolocation data for hotels and Fado venues obtained from the Foursquare and Google APIs respectively, and having applied a DBSCAN unsupervised classification algorithm to cluster hotels and Fado venues based on a distance of 250m, I recommend that my client considers the cultural and touristic hotspot located at the juncture of the Santa Catarina, Encarnaçao and Sacramento neighbourhoods for their venture.

It is worth noting that while the Alfama district is home to most of the Fado venues, and most of the hotels are found just north of the Praça do Comércio, these sites don't really offer the "best of both worlds" for someone looking to avoid walking over the many hills of Lisbon in order to benefit from both tourism and the appeal of Fado venues.

One caveat is in order, however. Given that this study relied on data derived from the free versions of Foursquare and Google APIs, and only returned one hotspot, future

researchers could benefit from applying a similar method to larger datasets, and could always adjust the parameters of the DBSCAN algorithm for more/fewer venues within a greater/shorter distance based on their particular use case.

Conclusion

In conclusion, this study has demonstrated the use of a clustering algorithm to identify a hotspot of activity between two types of venues within a city. In this case, I identified that the nexus of the Santa Catarina, Encarnaçao and Sacramento neighbourhoods in Lisbon represents a viable location for someone wishing to open a cultural centre to bring the beauty of Fado to people visiting the city on seven hills, all within a comfortable walking distance of numerous hotels and Fado venues.