Capstone Project Report

IBM Data Science Specialisation

Introduction

The proposed software will have the following main functionality. Provided an input location and a destination, it will investigate various characteristics of the input location, and suggest the most similar locations within a predetermined area of the destination.

This functionality is expected to address the following two use cases:

A.) Suppose a user would like to move from his/her current location to a new city, perhaps due to professional reasons, climate change or religious beliefs. It is difficult to find an area that has similar characteristics to the location the user is moving away from, and this software could help

B.) Suppose a user has a “dream” location in mind and would like to see if something similar is available closer to his/her current location.

The Minimal Viable Product (MVP) of this software will do the following:

- Accept 2 input coordinates
- Retrieve information on various characteristics of around the 2 input coordinates
- Perform hierarchical clustering
- Display the top 10 areas within a predefined range of input coordinate 2 most similar to input coordinate 1

Optional features:

- Allow users to set the range parameter for the 2 input coordinates
    - E.g. Find the top 10 most similar locations within a 10 miles range of input coordinate 2
- Allow users to set the limit for displaying the top *n* most similar locations

- Allow users to specify characteristics which should be considered
    - E.g. There needs to be a school in the are and a Mexican restaurant
- Allow users to provide city and country names as input instead of coordinates

Data

This software will use publicly accessible location data from Foursquare - As prescribed by the capstone project assignment.
For the MVP, no additional data source is required. The user will provide 2 sets of coordinates and the software will retrieve location characteristics data for both locations using the Foursquare API. The data will be parsed, cleaned, trimmed and combined into a data frame. Hierarchical clustering algorithms will be deployed on the data frame, centred around the input coordinate. The output will be a map, using the folium package, that will highlight the top 10 locations around the second input coordinate which are highly similar to the first input coordinate.

A high-level overview of the workflow:

1. The user provides input parameters (***location**, **destination***)
2. The software calls the Foursquare API and gets data within a 1-km range of ***location*** and within a ***5x5 grid of 2x2 km sized cells*** centred on the ***destination***
3. Both JSON data will be cleaned, selecting useful features and discarding others
4. The 2 data frames will be combined
    a. ***n*** columns (selected features)
    b. ***m*** rows (1 row for ***location*** and m-1 rows for ***destination***)
5. A hierarchical clustering algorithm will be called, centred around the row of ***location***
6. The results will be sorted by similarity to the row of ***location***
7. The ***best_hit*** will be printed to the user (with venues)
8. The ***best_hit*** number of similar rows will be plotted on a map

Examples of the data used:

```
get_similar_locations(location=(43.753259, -79.329656),
                      destination=(43.746143, -79.324630),
                      range_limit=10,
                      top_hits=10)
```

A Foursquare API endpoint is then called to get venue information for both ***location*** and ***destination*** using the following URL:

```
venues_url +
'/explore?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit
={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    lat,
    lng,
    radius,
    range_limit)
```

Example (fragment) of the returned data:

```
{'reasons': {'count': 0,
       'items': [{'summary': 'This spot is popular',
         'type': 'general',
         'reasonName': 'globalInteractionReason'}]},
      'venue': {'id': '51819802498ee3c6834b8e0b',
       'name': 'De Mello Palheta Coffee Roasters',
       'location': {'address': '2489 Yonge St',
        'crossStreet': 'Erksine Ave',
        'lat': 43.71179137179029,
        'lng': -79.39940300399607,
        'labeledLatLngs': [{'label': 'display',
          'lat': 43.71179137179029,
          'lng': -79.39940300399607}],
        'distance': 679,
        'postalCode': 'M4P 2H6',
        'cc': 'CA',
        'city': 'Toronto',
        'state': 'ON',
        'country': 'Canada',
        'formattedAddress': ['2489 Yonge St (Erksine Ave)',
         'Toronto ON M4P 2H6',
         'Canada']},
       'categories': [{'id': '4bf58dd8d48988d1e0931735',
         'name': 'Coffee Shop',
         'pluralName': 'Coffee Shops',
         'shortName': 'Coffee Shop',
         'icon': {'prefix':
'https://ss3.4sqi.net/img/categories_v2/food/coffeeshop_',
          'suffix': '.png'},
         'primary': True}],
       'photos': {'count': 0, 'groups': []}},
      'referralId': 'e-0-51819802498ee3c6834b8e0b-1'}
```

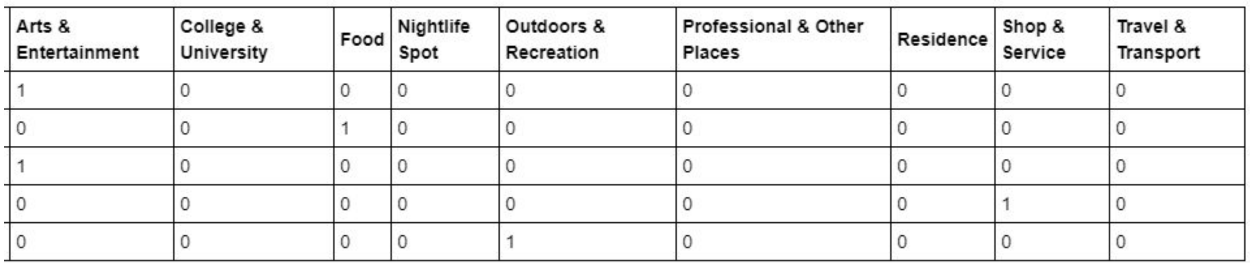
The resulting low-level categories (e.g. Cafe, Caffee Shop, etc) will then be mapped to higher-level categories, provided by another FourSquare API endpoint at URL:

```
venues_url +
"/categories?&client_id={}&client_secret={}&v={}".format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION
)
```

Example (fragment) of the returned data:

```
{'id': '4bf58dd8d48988d182941735',
    'name': 'Theme Park',
    'pluralName': 'Theme Parks',
    'shortName': 'Theme Park',
    'icon': {'prefix':
'https://ss3.4sqi.net/img/categories_v2/arts_entertainment/themepark_
',
     'suffix': '.png'},
    'categories': [{'id': '5109983191d435c0d71c2bb1',
      'name': 'Theme Park Ride / Attraction',
      'pluralName': 'Theme Park Rides/Attractions',
      'shortName': 'Theme Park',
      'icon': {'prefix':
'https://ss3.4sqi.net/img/categories_v2/arts_entertainment/themepark_
',
       'suffix': '.png'},
      'categories': []}]},
```

All the low-level categories will be merged based on the higher-level category (i.e. parent category) they belong to and grouped by the area they belong to (1-by-1 mile grid around the coordinates). These merged and grouped categories will be the features used for the hierarchical clustering algorithm. The features would look something like this:



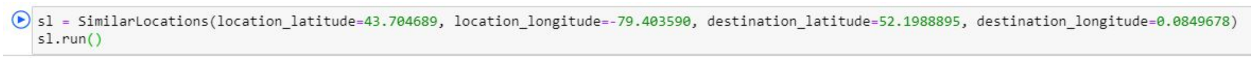
Arts & Entertainment	College & University	Food	Nightlife Spot	Outdoors & Recreation	Professional & Other Places	Residence	Shop & Service	Travel & Transport
1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0

Methods

The script performs the following steps:

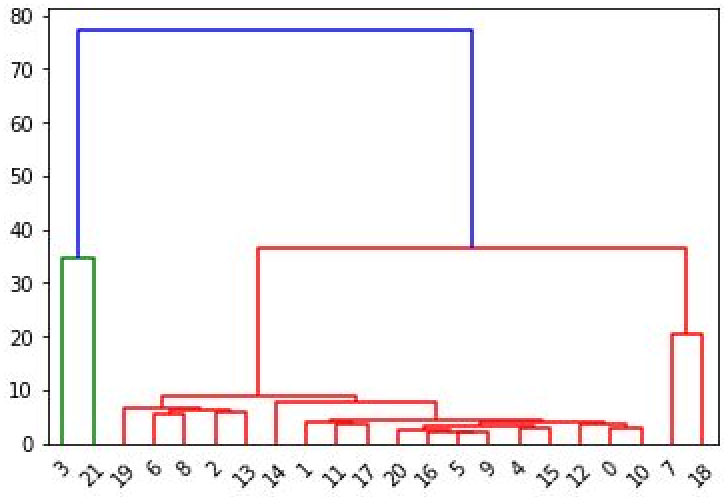
1. Initialise an instance of the Python object with user input
    a. sl = SimilarLocations(location_latitude=43.704689, location_longitude=-79.403590, destination_latitude=52.1988895, destination_longitude=0.0849678)
2. Start running the process
    a. sl.run()
3. Create data frames for the “location” and the “distance”
    a. The “destination” consists of a grid of 5x5 cells, each spanning 2x2 km and centred around the “destination” coordinates
4. Call the Places API of Foursquare to get the venues data for the “location” coordinates and the 25 “destination” area coordinates
5. Merge the data frames
6. Encode the venue categories into 0 and 1
7. Sum up the venue categories per location
    a. This yields a data frame with 26 rows and *n* number of columns where *n* is the number of venue categories
8. Perform hierarchical clustering of the data frame
9. Extract the “destination” grid cell which is most similar to the “location”
10. Render a map which highlights the best matching area
11. Display the venues for both the “location” and the best matching “destination”

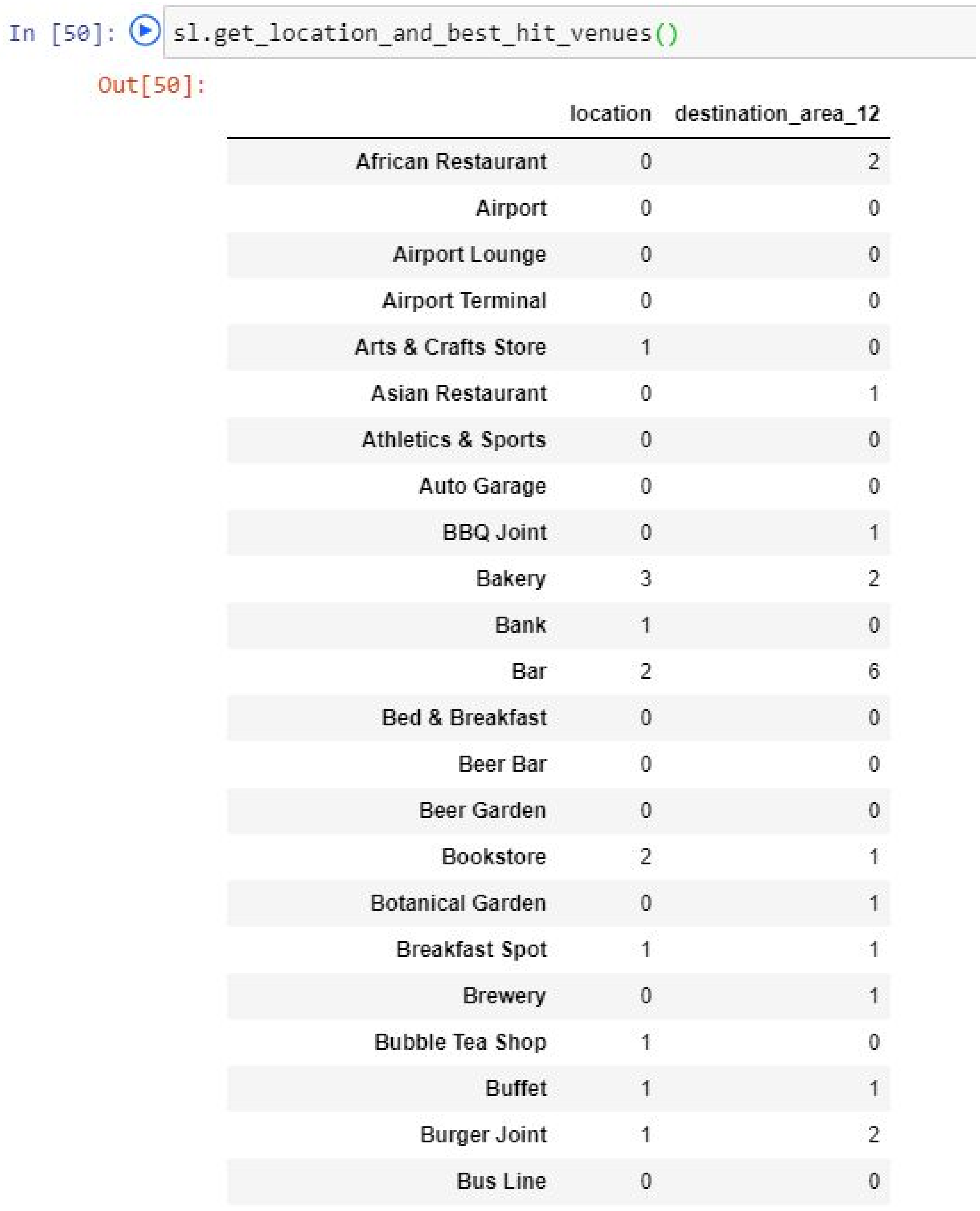
Example



```
sl = SimilarLocations(location_latitude=43.704689, location_longitude=-79.403590, destination_latitude=52.1988895, destination_longitude=0.0849678)
sl.run()
```

```
In [31]: # Display the dendrogram
         sl.render_dendrogram()
```

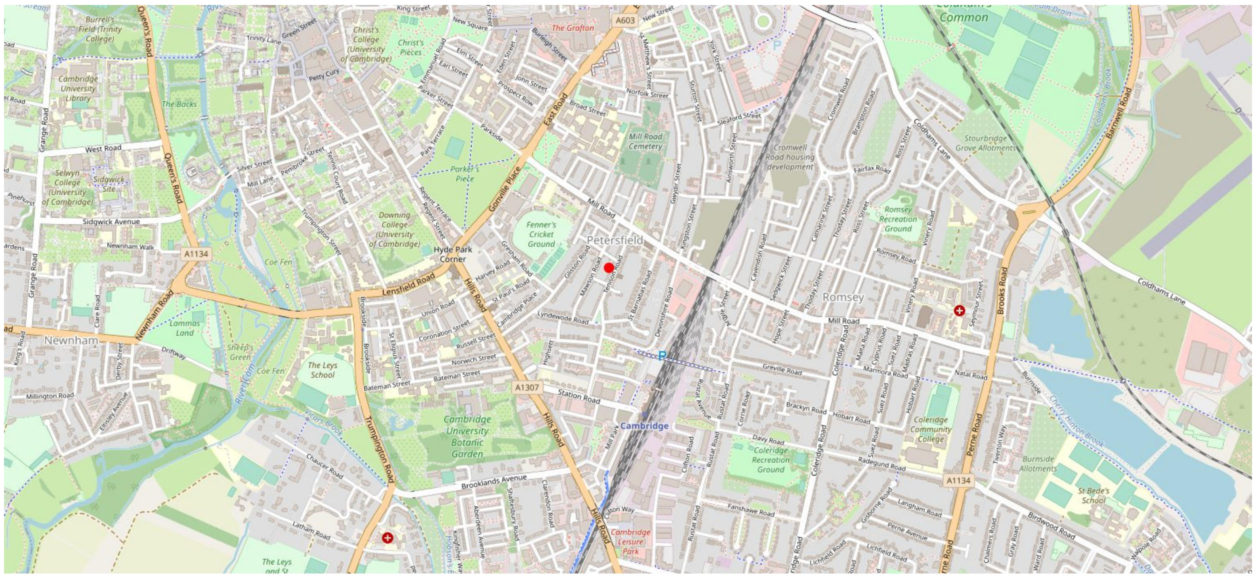




```
In [50]: sl.get_location_and_best_hit_venues()
```

Out[50]:

	location	destination_area_12
African Restaurant	0	2
Airport	0	0
Airport Lounge	0	0
Airport Terminal	0	0
Arts & Crafts Store	1	0
Asian Restaurant	0	1
Athletics & Sports	0	0
Auto Garage	0	0
BBQ Joint	0	1
Bakery	3	2
Bank	1	0
Bar	2	6
Bed & Breakfast	0	0
Beer Bar	0	0
Beer Garden	0	0
Bookstore	2	1
Botanical Garden	0	1
Breakfast Spot	1	1
Brewery	0	1
Bubble Tea Shop	1	0
Buffet	1	1
Burger Joint	1	2
Bus Line	0	0



Results

The tool described here is aiming to answer the question “Which area of A is most similar to areas around B?” which would normally involve much manual work, compiling data from various sources and then comparing the available data to identify the best match.

For example, suppose that John Doe is living in Toronto, Canada, but he got an exciting job offer in Cambridge, UK. He likes the area of Toronto he currently lives in and would like to have a similar setting in terms of the venues available to him within a short distance. The software presented here can help John to find the area in Cambridge which is most similar to his current location in Toronto.

This tool is a lightweight software that generally aims to address the following 2 main use cases:

- A user is considering to move to a new location and would like to get assistance in terms of finding an area that is most similar to the user’s current location
- A user thinks that it would be great to live at a certain location, they can get an idea of which area around their current location is most similar to the dream location in terms of available venues

It solves this problem by retrieving publicly available data on venues using the Places API of Foursquare and then parses, transforms and analyses the data, resulting in a hierarchical cluster that can be used to identify the area which is most similar to the input area of the user.

Discussion

This tool would benefit from a number of further improvements, such as:

- Support using city & country as input instead of decimal degree coordinates
- Support changing the range of the venues the API returns
- Support changing the limit of the API
- Provide not only the best hit but also a sorted list of all the locations from the destination grid
- Support larger grids (not only 5x5)
- Allow the user to filter what venue categories are irrelevant and/or emphasise venue categories that should weight more

However, given the scope of this project, i.e. that it is a 2-weeks long course project, the implemented functionality should serve its purpose and can be used as a Proof of Concept (PoC).

Conclusion

The tool presented here can help answer the question “Which area around coordinates A is most similar to my area of B, in terms of available venues within a short distance?”.

There are a number of potential improvements, but as a PoC this implementation is useful in showing how to use data from a publicly accessible API, how to parse it using Python packages such as pandas and how to perform various machine learning calculations, such as hierarchical clustering.