Data Science Capstone
IBM Coursera Course

Joseph Iles

I. Introduction/Business Problem

Bristol City Council are proposing a new initiative for Low-Traffic Neighbourhoods (LTNs). LTNs are an area in a city in which all through-traffic is banned, meaning that the area becomes highly pedestrianised. As traffic and pollution levels have reached exorbitant levels in recent years, LTNs are designed to promote the use of greener alternative means of transport such as cycling or electric scooters. While business owners are often hostile to the notion of an LTN, research has shown time-and-time again that businesses actually benefit from the increased foot traffic.

The paper outlines an approach to identify a candidate area for a new LTN in Bristol. It is postulated that neighbourhoods which contain a larger number of hospitality businesses, as opposed to industrial or office spaces, will benefit most from the introduction of an LTN, because these types of businesses will benefit from the increased footfall.

II. Data

Bristol is divided into 116 informal neighbourhoods, names of which are found in the following Wikipedia article: https://en.wikipedia.org/wiki/Subdivisions_of_Bristol. The names of these neighbourhoods will be used to form part of a Foursquare API request that will return a JSON file of local businesses.

The Foursquare API can return venues close to a specified latitude and longitude. Since it is unlikely that Foursquare will recognise the names of each Bristol neighbourhood in a general search call, the latitude and longitude of each neighbourhood is found using the ArcGIS REST API. Listing 1 shows a snipped of the response from the ArcGIS REST API for the search term 'Bedminster, Bristol'. The actual response returns a number of potential candidates, but for the purposes of this study only the first (or top) candidate is considered. The data is wrangled into a Pandas DataFrame so that each neighbourhood is allocated a set of coordinates.

Listing 1. ArcGIS REST API response for 'Bedminster, Bristol'

```
{'spatialReference': {'wkid': 4326, '
    latestWkid': 4326},
 'candidates': [{'address': 'Bedminster,
     Bristol, Avon, England',
   'location': {'x': -2.6091299999999364,
       'y': 51.44023000000004},
   'score': 100,
   'attributes': {},
   'extent': {'xmin': -2.6191299999999362
     ,
    'ymin': 51.430230000000044,
    'xmax': -2.5991299999999367,
    'ymax': 51.45023000000004}}]}
```

The Foursquare API is then used to collect details of venues for each of the neighbourhoods, by sending the coordinates as arguments to a REST API call. The returned JSON response contains a list of nearby venues for a given call, along with a category for each venue. The category is used for the study going forward, as it allows the grouping of venues together to get a picture of which types of business are present in a given area. Sometimes, there are multiple categories for a given venue, but for our purposes here it is fine to use the first item as this will reflect the most accurate category.

III. Methodology

First, the Beautiful Soup and Requests Python packages are used to scrape the names of Bristol neighbourhoods from the Wikipedia page linked in the previous section. It was found that bulleted (or li in html language) items wrap the names of the neighbourhoods, so Beautiful Soup is used to extract all of the bullets on the page. There are more bullets on the page than just the neighbourhoods, so the beginning and end neighbourhoods of 'Bristol city centre' and 'Withywood' are used to form the start and end points of an index slice. This results in a list of named neighbourhoods in Bristol.

Next, the ArcGIS REST API is used to collect lateral and longitudinal coordinates for each of the neighbourhoods. Functions are created which call the REST API for a given row of a DataFrame to gets either the latitude or longitude of the response, we simply apply these functions to each row of the DataFrame and add the coordinates to the corresponding neighbourhood's row. It was found that a number of neighbourhoods were incorrectly identified as other eponymous neighbourhoods in the UK, for the purposes of this experiment those rows were simply dropped from the DataFrame. The resulting neighbourhoods are then plotted using the Folium package, resulting in the map seen in Figure 1.

Next, the Foursquare API is used to get a list of venues near each neighbourhoods. The coordinates discovered in the last paragraph and fed to a REST API call using the Requests package, making sure to specify a limit of returned venues in a given radius. The response is in JSON format, making it easy to loop through the levels of the response to generate lists

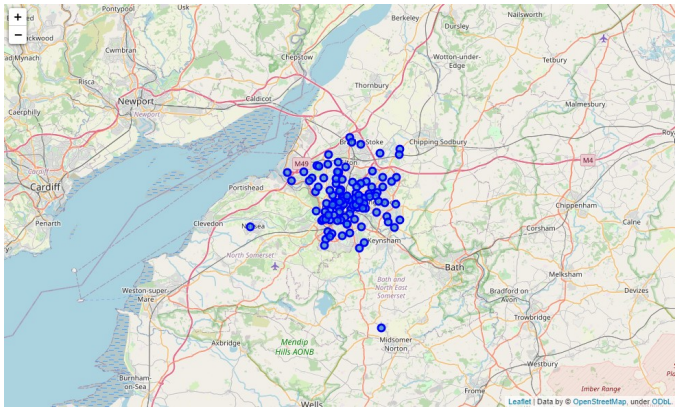

Fig. 1. Bristol neighbourhoods

of venue names, categories and coordinates for each call to a given neighbourhood. This method is written into a function which can then perform the API call for every neighbourhood in the DataFrame when applied row-by-row, with the venue data being appended to the end of the main DataFrame at each step. Duplicates are removed that may have appeared due to overlapping of the search radii for each neighbourhood.

There are a very diverse set of categories, so much so that it becomes infeasible to make any generalised obervations. For example, it isn't useful to know that there is an Indian Restaurant and a Fish & Chips Shop in an area, it is more useful that both of these are identified as Food places. Foursquare provdes a special REST API link which will return a JSON file of the hierarchy of venue categories. This is called and Python dictionary is defined which creates key, value pairs of top-level categories to a list of all the lower-level category names. From here it is simple to loop through the DataFrame once more, replacing each lower-level category with the corresponding top-level category name (Table III)

Top-Level Category
Arts & Entertainment
College & University
Event
Food
Nightlife Spot
Outdoors & Recreation
Professional & Other Places
Shop & Service
Travel & Transport

TABLE I
TOP-LEVEL CATEGORIES FROM FOURSQUARE VENUE HIERARCHY

Density-based spatial clustering of applications with noise (DBSCAN) is used to find geographical clusters of venues. From this point on, the neighbourhoods are not longer used in the analysis. This is because there may be instances of natural clusters of venues being sat on the border of two neighbourhoods - in this case is makes no sense to worry about which neighbourhood the venues officially sit in because the pedestrianisation can occur across neighbourhood lines. Scikit-learn's DBSCAN class is used to model the density-regains of the Bristol venues based entirely on latitudinal and longitudinal coordinates.

DBSCAN takes two important parameters: eps and min_samples. The former being the minimum distance between two samples for it to be considered as being on the neighbourhood of another. The later is the number of samples in a neighbourhood for a point to be considered as a core point - in other words, this parameter defines the minimum number of points required for a set of points to be considered a cluster. There is a degree of experimentation required to balance the two parameters, but the values of eps = 0.002 and min_samples = 5 were seen to give the best output for the Bristol dataset. The model is fit and each venue is placed into a neighbourhood, resulting in the clusters seen in Figure 2

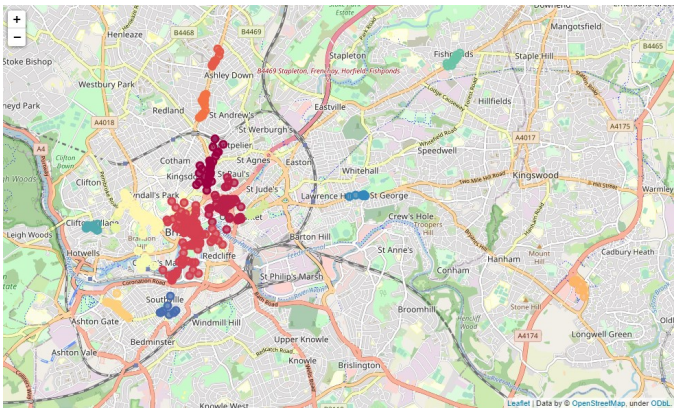

Fig. 2. Bristol venue clusters

Finally, each of the clusters is named based on the model top-level category for each cluster. For example, the cluster of points sitting between Lawrence Hill and St George were were mostly Food venues, so a Folium marker is placed at the centroid of these points and given a 'Food' popup accordingly. This is repeated for each cluster of points on the map, as shown in Figure 3

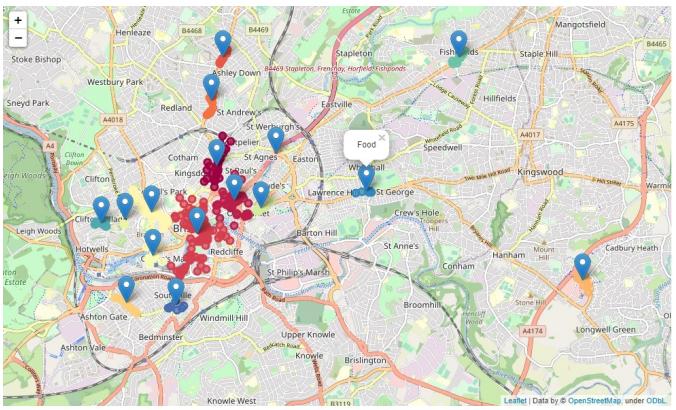

Fig. 3. Lawrence Hill venue cluster with named Folium marker

IV. Results

As can be seen in Figure 2, the DBSCAN algorithm as able to successfully group the Bristol venues into 19 clusters. The largest of these is in or around Bristol city centre, this is expected as it is a densely built-up area with lots of businesses. There are a couple of small clusters towards the outskirts of the city, such as the two seen near Bradley Stoke and the larger one near Longwell Green.

Inspection of the Folium object shown in Figure 3 shows that the overwhelming majority of clusters are full of businesses falling into the following top-level categories: Food, Shop & Service and Nightlife Spot.

V. Discussion

Most of Bristol city centre is already partially pedestrianised, so this area is omitted from future considerations. As is the cluster near Longwell Green, since this seems to have picked up venues belonging to a retail park. It would also be difficult to pedestrianise the two clusters found to the North of the centre on Gloucester Road, as this road is a main road connecting the Centre to the north of the city.

This reduction leaves a number of areas which may be ripe for pedestrianisation. Of particular note are the areas near Ashton Gate and Clifton Village where both aren't on major connecting roads and contain a healthy mixture of Food venues.

VI. Conclusion

A geographical clustering of business venues in the City of Bristol was performed using DBSCAN. Data was scraped from Wikipedia using the Beautiful Soup and Requests Python packages before it is cleaned into a Pandas DataFrame of Bristol neighbourhoods. A call to the REST API of Foursquare was then performed for the coordinates of each of these neighbourhoods, with the discovered venues appended to the DataFrame. Each of these venues were then categorised according to Foursquare's own venue hierarchy. Finally, DBSCAN grouped each of the venues based on their geographical location and the top venue category was determined for each cluster.

Outside of a few notable, and important omissions, a number of promising clusters of business are identified for potential pedestrianisation. The most promising of these are the areas near Ashton Gate and Clifton Village.