IBM Assignment Apr 2020

APRIL 24

Alex Holloway

UK Contractor New City Guide

Project Report and Evaluation

Introduction

The business problem I have chosen to tackle in this final assignment is a common concern for contractors who undertake onsite assignments across the country. Contractors who need to stay away from home during the week to complete a project onsite with their client often need to source accommodation in a town or city with which they are not familiar. To facilitate this process, I have decided to create as my submission the UK Contractor New City Guide which will characterise a geographical area of the user's choosing using clustering to inform the contractor which areas might be of interest to him or her.

In reality, these decisions are usually based on a number of factors; typically, proximity to location of work and price and availability of suitable accommodation. However, there are often additional criteria against which a contractor will choose a location to stay and that can include a range of venues to support their lifestyle while working away. It could be proximity to shops, leisure facilities or even sites of tourist interest should they wish to maximise the value they get from staying somewhere new.

To this end, a tool which is of value to a range of users who could be staying anywhere in the UK who themselves have a range of tastes and which is responsive to the user's selection of location AND venues of interest is required.

"The UK Contractor New City Guide which will characterise a geographical area of the user's choosing"

Data Used

The data which will be required to create this tool consists of two main components: some UK-specific geographical data and secondly categorised venue data as provided by the Foursquare API.

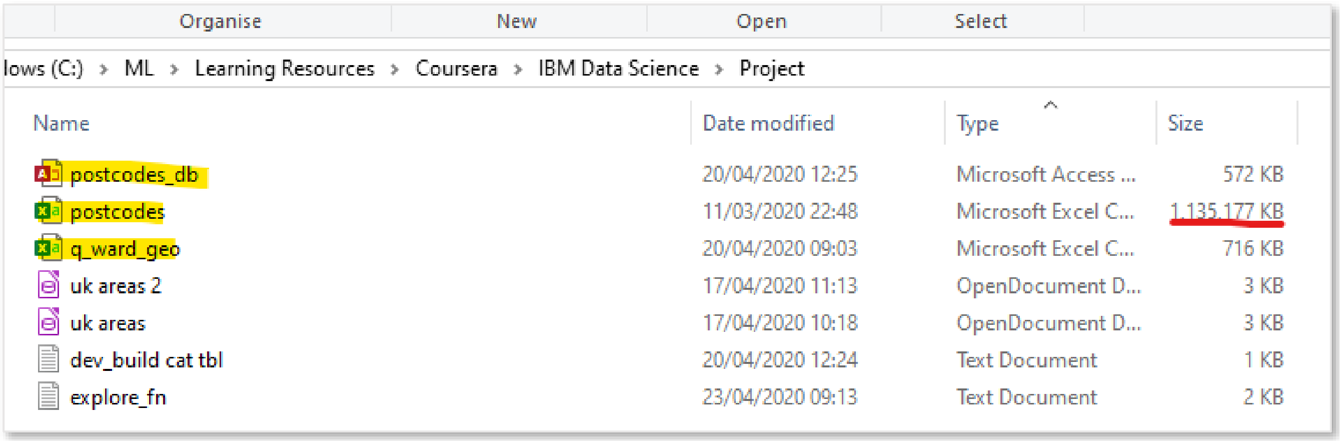
The geographical data must be at a sufficient granularity to pinpoint specific areas of a town or city and for this the political demarcation of 'Ward' will be used. These are UK areas to which members of parliament are elected and a typical sized UK city may have 5 - 10 of these. Rural wards are typically much larger as they are setup to contain a similar number of voters and so population density is a determining factor in the size of the geography. For what is most likely a town/city analysis, this level of geography is deemed appropriate. This data can be found online though some wrangling may be required.

The venue data will be lifted from the Foursquare API as in earlier Labs in this course. Foursquare returns venue information in a specified radius of provided latitude and longitude coordinates. Thus as the dataset will contain ward-level coordinates for the user's chosen city, the API can be leveraged to characterise the venues in each area. The intention is that the user will be able to alter a simple search pattern to return venue information from a subset of the Foursquare database. Thus, if the contractor is primarily interested in proximity to supermarkets or amphitheatres, this can be accommodated.

The application of machine learning to the data described will be to cluster wards in the scope of the search terms provided by the user to characterise areas of the town and city against the subset of venue categories selected. A map will be generated with wards identified by map pointers which are in turn color-coded by cluster so that an at-a-glance characterisation of area is provided. As such the user can answer such questions as 'which area of Birmingham provides best access to Athletics and Leisure Facilities', for example.

Methodology

The first practical step in starting the analysis was to source and wrangle some geographical data for mapping containing latitudes and longitudes for UK Wards. It was suspected that this might not exist in a clean form against a publicly accessible url, either to scrape from a webpage or to glean directly from a csv or similar. This turned out to be the case and the most useful source was a postcode-level (zip) download which required aggregation.

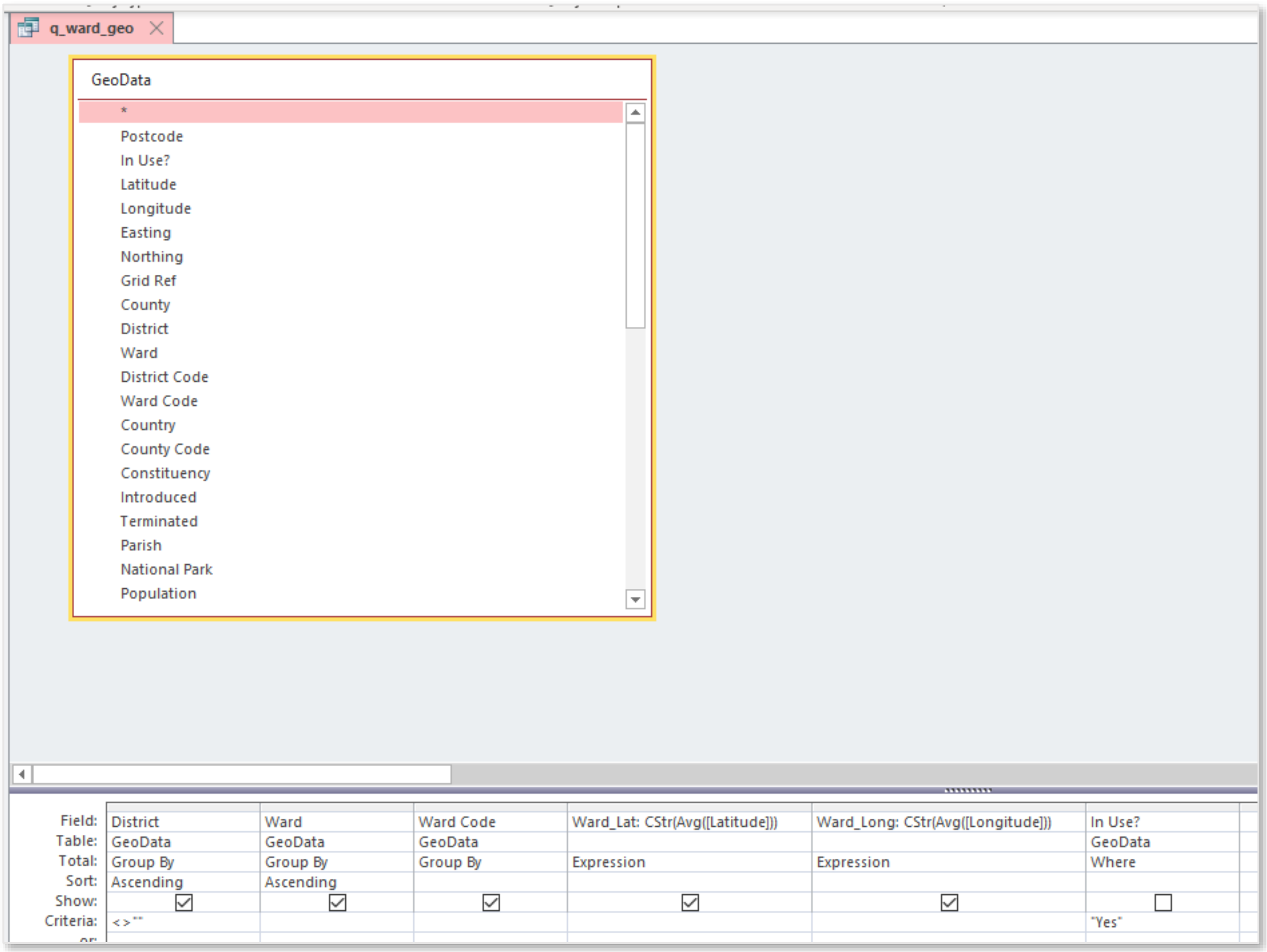


Organise
New
Open
Select
lows (C:) > ML > Learning Resources > Coursera > IBM Data Science > Project

Name	Date modified	Type	Size
postcodes_db	20/04/2020 12:25	Microsoft Access ...	572 KB
postcodes	11/03/2020 22:48	Microsoft Excel C...	1,135,177 KB
q_ward_geo	20/04/2020 09:03	Microsoft Excel C...	716 KB
uk areas 2	17/04/2020 11:13	OpenDocument D...	3 KB
uk areas	17/04/2020 10:18	OpenDocument D...	3 KB
dev_build cat tbl	20/04/2020 12:24	Text Document	1 KB
explore_fn	23/04/2020 09:13	Text Document	2 KB

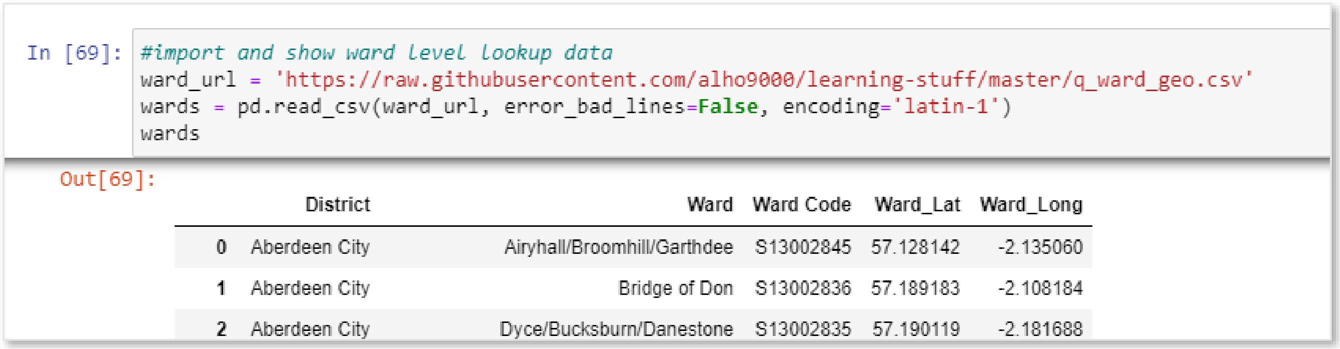
Above: note the filesize of the base csv, 'postcodes.csv'

Rather than attempt to host this file or handle a gigabyte of data in python, the data was summarized in MS Access and aggregated to Ward level. The resulting query was exported as 'q_ward_geo.csv' [716kb!] and the file was uploaded to GitHub to be used in the project's python Notebook as a source.



Field:	District	Ward	Ward Code	Ward_Lat: CStr(Avg([Latitude]))	Ward_Long: CStr(Avg([Longitude]))	In Use?
Table:	GeoData	GeoData	GeoData			GeoData
Total:	Group By	Group By	Group By	Expression	Expression	Where
Sort:	Ascending	Ascending				
Show:	☑	☑	☑	☑	☑	☐
Criteria:	<>""					"Yes"

Above: summarizing data in MS Access.



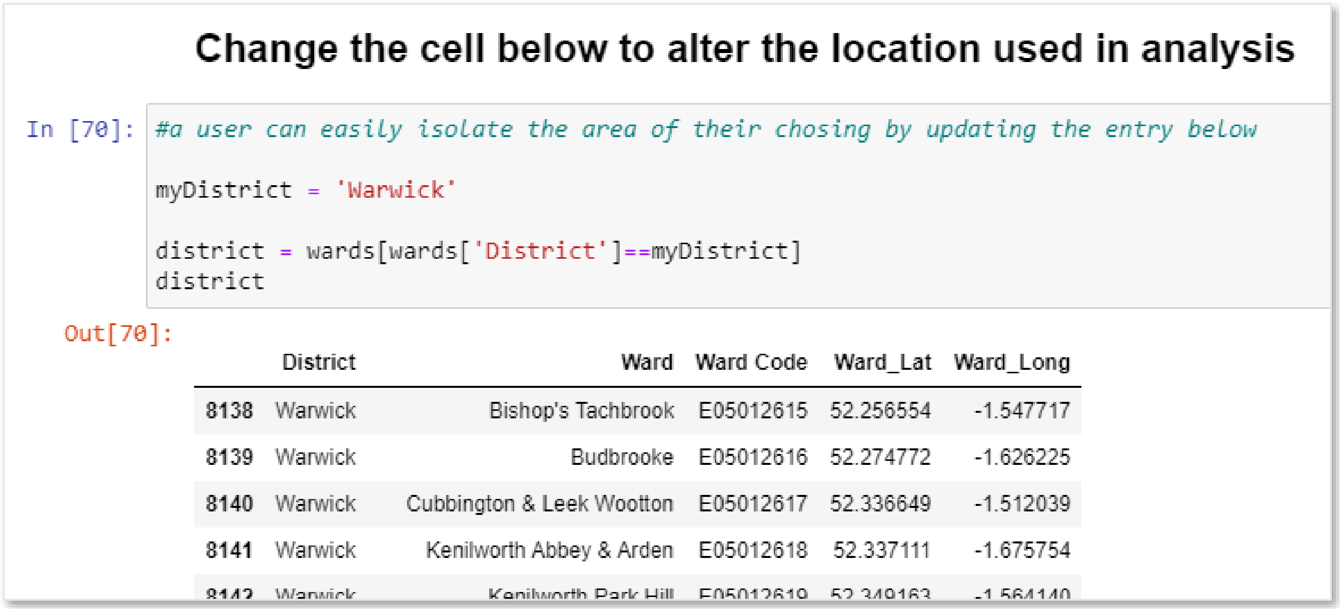
```
In [69]: #import and show ward level lookup data
         ward_url = 'https://raw.githubusercontent.com/alho9000/learning-stuff/master/q_ward_geo.csv'
         wards = pd.read_csv(ward_url, error_bad_lines=False, encoding='latin-1')
         wards
```

Out[69]:

	District	Ward	Ward Code	Ward_Lat	Ward_Long
0	Aberdeen City	Airyhall/Broomhill/Garthdee	S13002845	57.128142	-2.135060
1	Aberdeen City	Bridge of Don	S13002836	57.189183	-2.108184
2	Aberdeen City	Dyce/Bucksburn/Danestone	S13002835	57.190119	-2.181688

Above: the csv data imported to a dataframe using Pandas read_csv() function

The notebook continues by offering the user some dynamic choice in selecting the area they are interested in analyzing. The dataframe shown confirms to the user that their selection for myDistrict is valid. Should the user return 0 results, a search tool follows where they can see valid entries for their intended search pattern.



Change the cell below to alter the location used in analysis

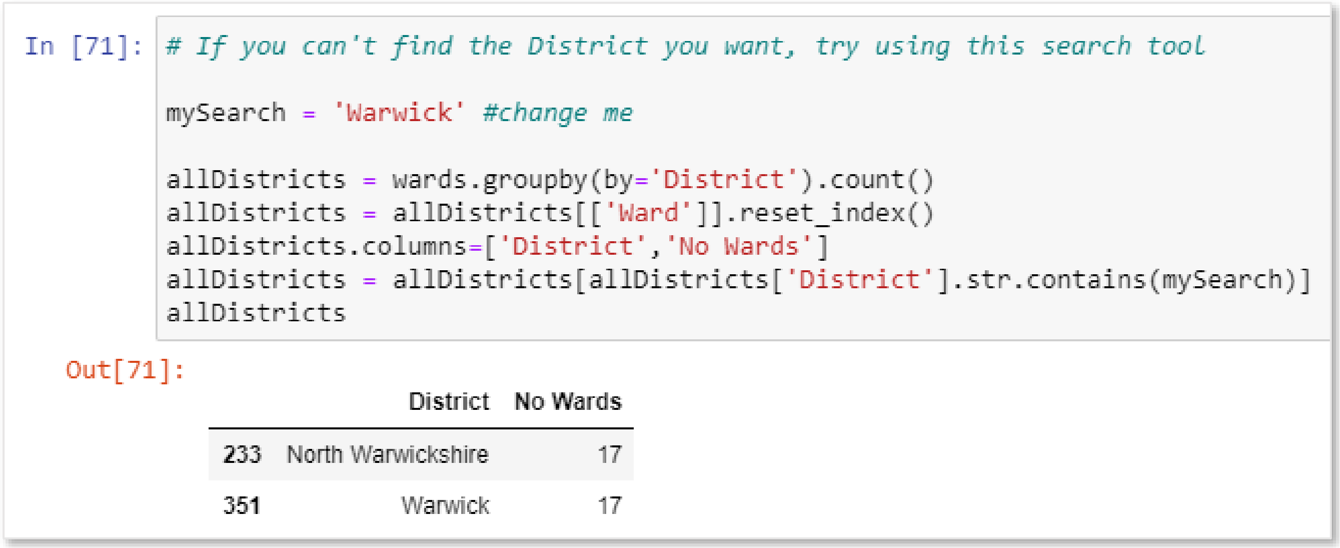
In [70]:

```
#a user can easily isolate the area of their chosing by updating the entry below

myDistrict = 'Warwick'

district = wards[wards['District']==myDistrict]
district
```

Out[70]:

	District	Ward	Ward Code	Ward_Lat	Ward_Long
8138	Warwick	Bishop's Tachbrook	E05012615	52.256554	-1.547717
8139	Warwick	Budbrooke	E05012616	52.274772	-1.626225
8140	Warwick	Cubbington & Leek Wootton	E05012617	52.336649	-1.512039
8141	Warwick	Kenilworth Abbey & Arden	E05012618	52.337111	-1.675754
8142	Warwick	Kenilworth Park Hill	E05012619	52.349163	-1.564140

For example, ‘Warwickshire’ would have returned no results and the user can determine this for themselves.



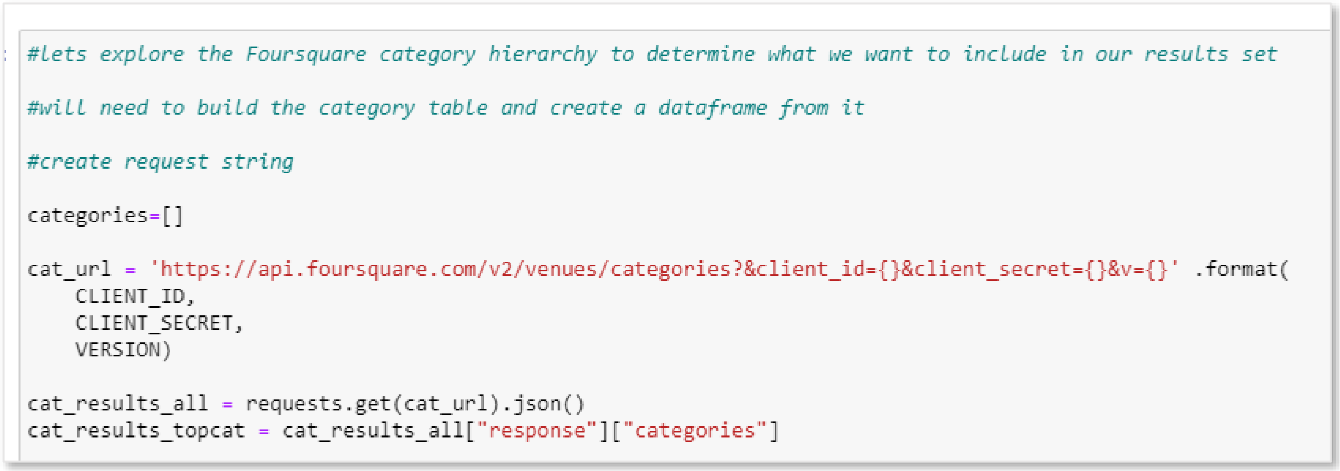
In [71]:

```
# If you can't find the District you want, try using this search tool

mySearch = 'Warwick' #change me

allDistricts = wards.groupby(by='District').count()
allDistricts = allDistricts[['Ward']].reset_index()
allDistricts.columns=['District','No Wards']
allDistricts = allDistricts[allDistricts['District'].str.contains(mySearch)]
allDistricts
```

Out[71]:

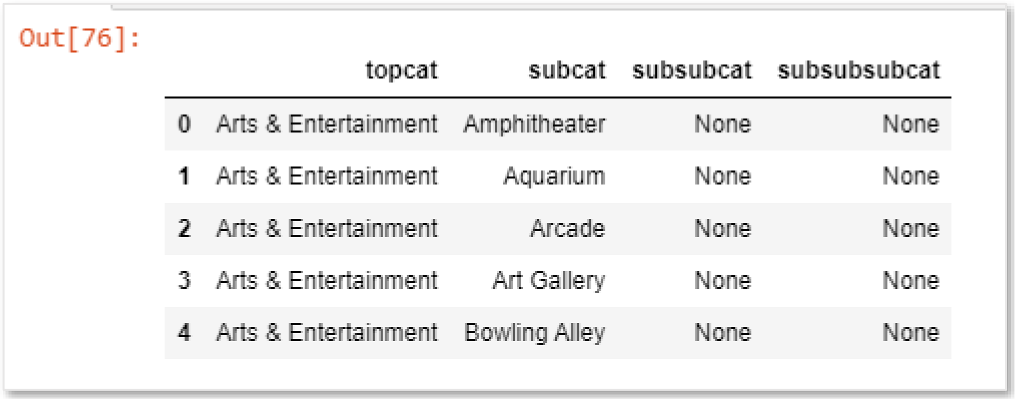
	District	No Wards
233	North Warwickshire	17
351	Warwick	17

The tool requires a degree of flexibility in allowing the user to select either a high- or low level of venue category in Foursquare and so a lookup table is generated from a Foursquare request and is unrolled into a single list of permissible venue categories.



```
#lets explore the Foursquare category hierarchy to determine what we want to include in our results set

#will need to build the category table and create a dataframe from it

#create request string

categories=[]

cat_url = 'https://api.foursquare.com/v2/venues/categories?&client_id={}&client_secret={}&v={}' .format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION)

cat_results_all = requests.get(cat_url).json()
cat_results_topcat = cat_results_all["response"]["categories"]
```

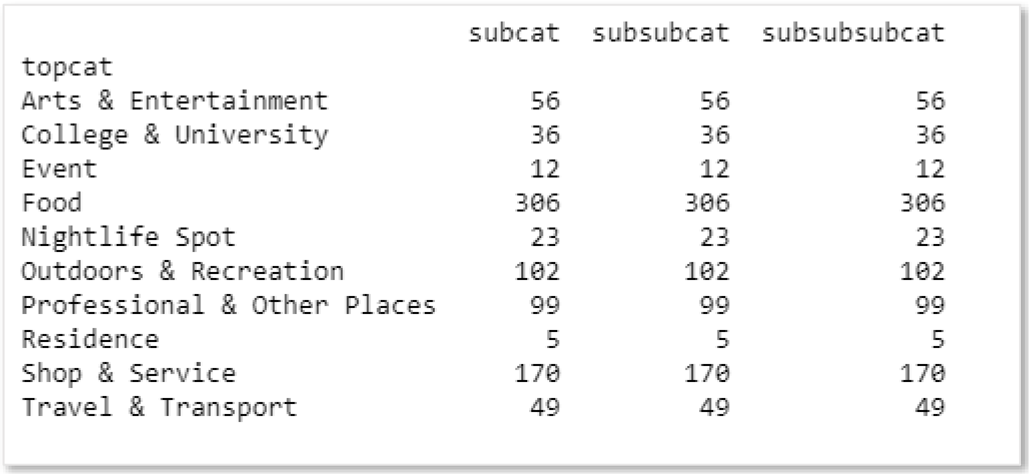
Above: create simple request



Out[76]:

	topcat	subcat	subsubcat	subsubsubcat
0	Arts & Entertainment	Amphitheater	None	None
1	Arts & Entertainment	Aquarium	None	None
2	Arts & Entertainment	Arcade	None	None
3	Arts & Entertainment	Art Gallery	None	None
4	Arts & Entertainment	Bowling Alley	None	None

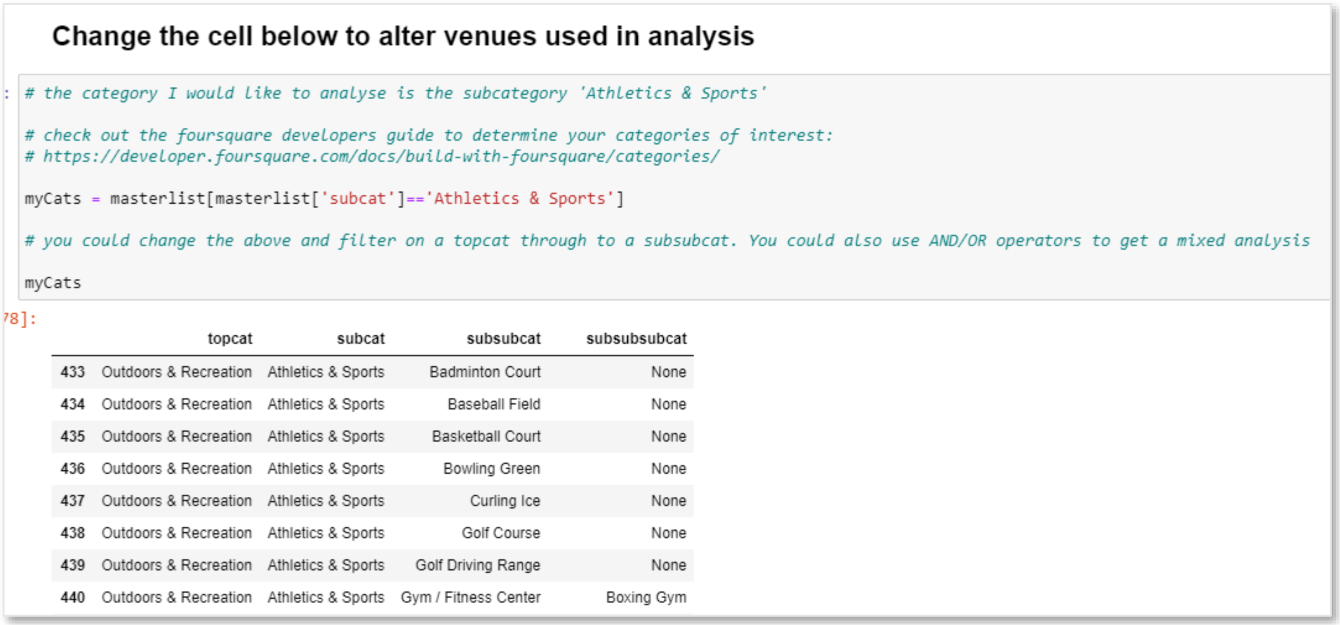
Above: loop through json response and build a hierarchical table



```
                                   subcat  subsubcat  subsubsubcat
topcat
Arts & Entertainment                   56         56            56
College & University                   36         36            36
Event                                  12         12            12
Food                                  306        306           306
Nightlife Spot                         23         23            23
Outdoors & Recreation                 102        102           102
Professional & Other Places            99         99            99
Residence                               5          5             5
Shop & Service                        170        170           170
Travel & Transport                     49         49            49
```

Above: hierarchy summarized to validate response

The user is then presented with the opportunity to select a venue filter of their choosing which will inform all the clustering which follows:



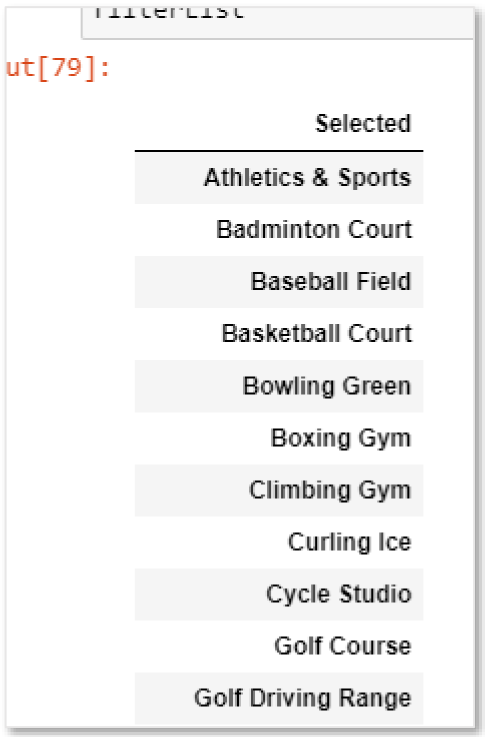
Change the cell below to alter venues used in analysis

```
# the category I would like to analyse is the subcategory 'Athletics & Sports'

# check out the foursquare developers guide to determine your categories of interest:
# https://developer.foursquare.com/docs/build-with-foursquare/categories/

myCats = masterlist[masterlist['subcat']=='Athletics & Sports']

# you could change the above and filter on a topcat through to a subsubcat. You could also use AND/OR operators to get a mixed analysis

myCats
```

78]:

	topcat	subcat	subsubcat	subsubsubcat
433	Outdoors & Recreation	Athletics & Sports	Badminton Court	None
434	Outdoors & Recreation	Athletics & Sports	Baseball Field	None
435	Outdoors & Recreation	Athletics & Sports	Basketball Court	None
436	Outdoors & Recreation	Athletics & Sports	Bowling Green	None
437	Outdoors & Recreation	Athletics & Sports	Curling Ice	None
438	Outdoors & Recreation	Athletics & Sports	Golf Course	None
439	Outdoors & Recreation	Athletics & Sports	Golf Driving Range	None
440	Outdoors & Recreation	Athletics & Sports	Gym / Fitness Center	Boxing Gym

This filter selection is unrolled into a single grouped list:

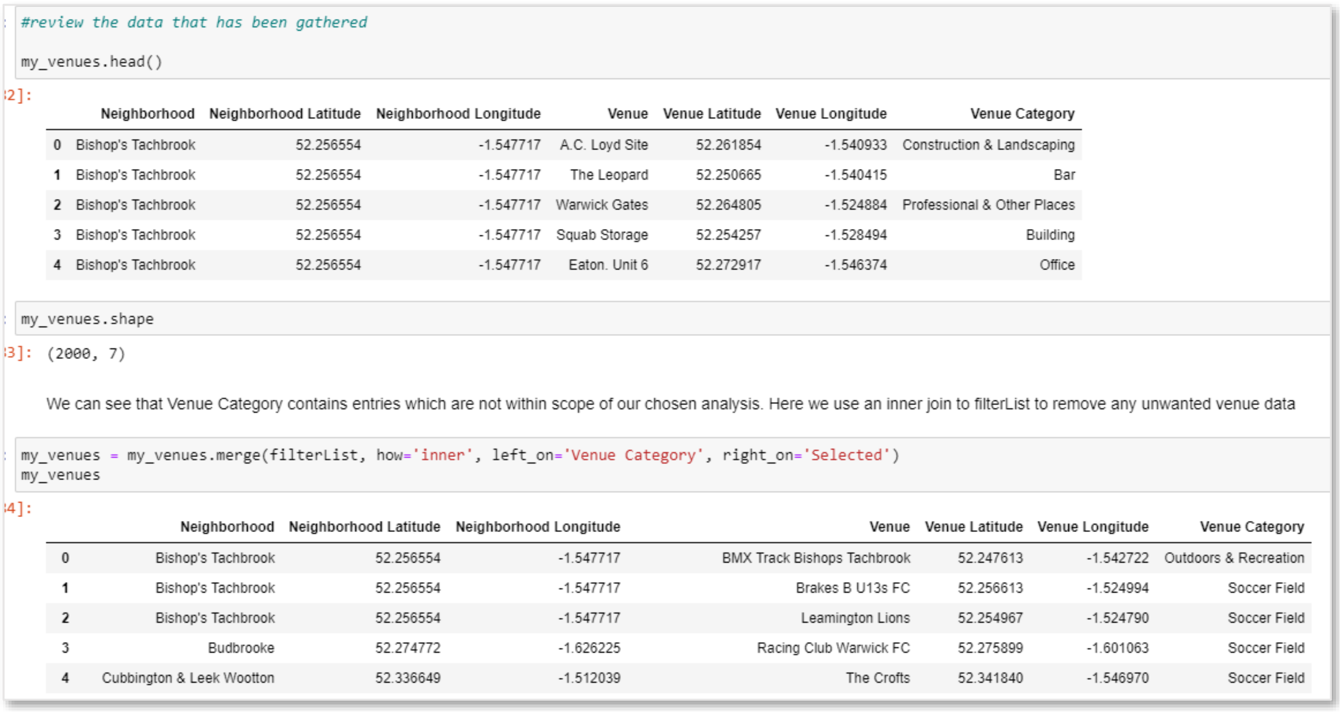


```
ut[79]:
```

Selected
Athletics & Sports
Badminton Court
Baseball Field
Basketball Court
Bowling Green
Boxing Gym
Climbing Gym
Curling Ice
Cycle Studio
Golf Course
Golf Driving Range

Having gathered and cleaned all user requirements, the notebook continues by getting all the Foursquare data for the user's chosen Wards. This was initially attempted by using the function from an earlier Lab which leveraged the 'explore' endpoint. However, when the data returned from a familiar city didn't match expectations, some research indicated that the 'explore' endpoint only returns *recommended* venues that are open at runtime which was deemed too limited. As such, the function was modified to use the 'search' endpoint which returns all relevant results.

The data returned by the function is not yet filtered to match the user's selected categories, so this was the final stage before one-hot encoding of this dataframe for use in clustering.



```
#review the data that has been gathered

my_venues.head()
```

2]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Bishop's Tachbrook	52.256554	-1.547717	A.C. Loyd Site	52.261854	-1.540933	Construction & Landscaping
1	Bishop's Tachbrook	52.256554	-1.547717	The Leopard	52.250665	-1.540415	Bar
2	Bishop's Tachbrook	52.256554	-1.547717	Warwick Gates	52.264805	-1.524884	Professional & Other Places
3	Bishop's Tachbrook	52.256554	-1.547717	Squab Storage	52.254257	-1.528494	Building
4	Bishop's Tachbrook	52.256554	-1.547717	Eaton. Unit 6	52.272917	-1.546374	Office

```
my_venues.shape
```

3]:

```
(2000, 7)
```

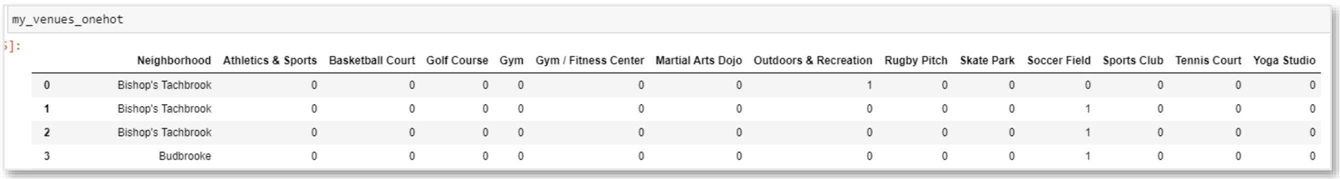
We can see that Venue Category contains entries which are not within scope of our chosen analysis. Here we use an inner join to filterList to remove any unwanted venue data

```
my_venues = my_venues.merge(filterList, how='inner', left_on='Venue Category', right_on='Selected')
my_venues
```

4]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Bishop's Tachbrook	52.256554	-1.547717	BMX Track Bishops Tachbrook	52.247613	-1.542722	Outdoors & Recreation
1	Bishop's Tachbrook	52.256554	-1.547717	Brakes B U13s FC	52.256613	-1.524994	Soccer Field
2	Bishop's Tachbrook	52.256554	-1.547717	Leamington Lions	52.254967	-1.524790	Soccer Field
3	Budbrooke	52.274772	-1.626225	Racing Club Warwick FC	52.275899	-1.601063	Soccer Field
4	Cubbington & Leek Wootton	52.336649	-1.512039	The Crofts	52.341840	-1.546970	Soccer Field

One-hot encoding prepares this data for clustering:



```
my_venues_onehot
```

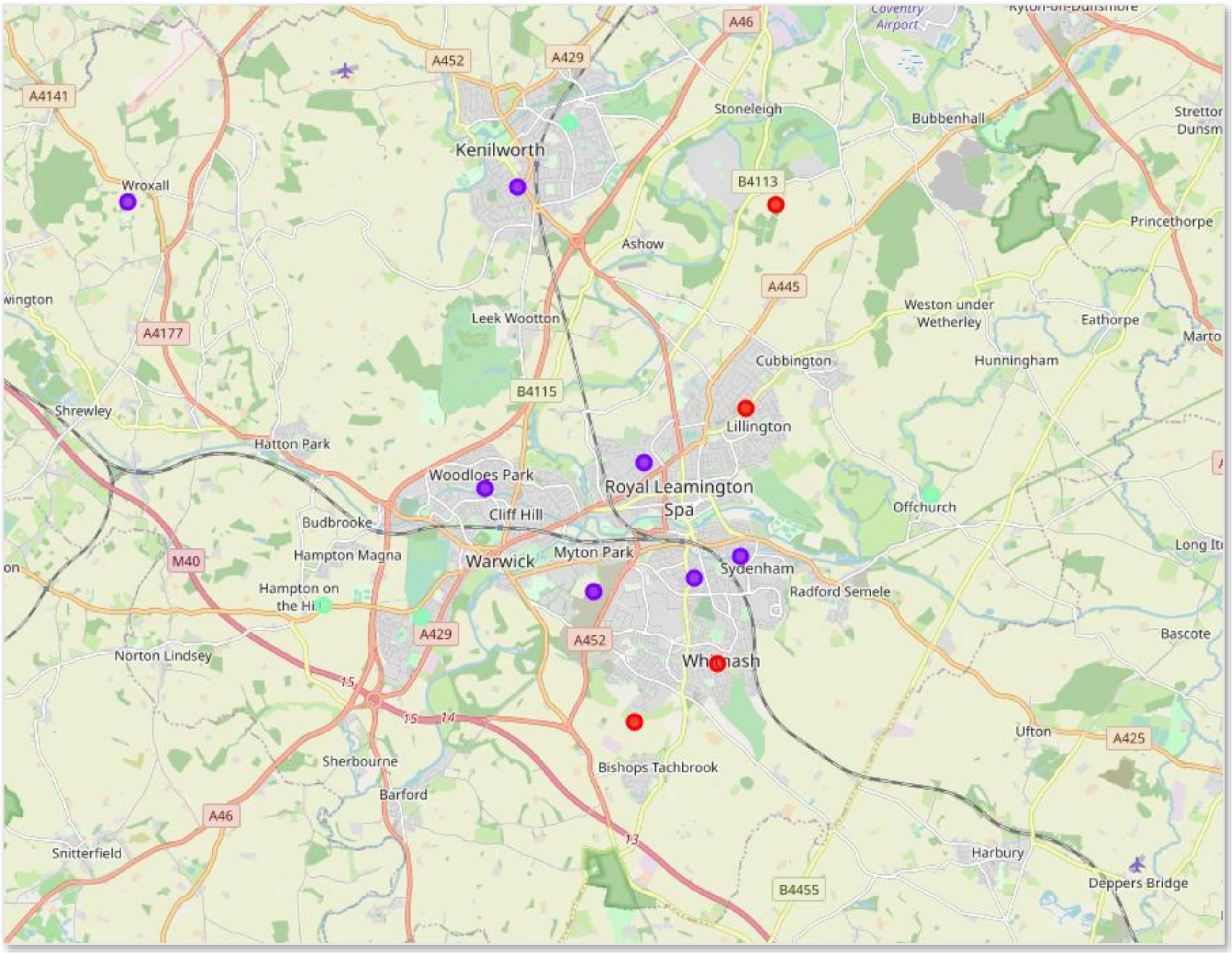
5]:

	Neighborhood	Athletics & Sports	Basketball Court	Golf Course	Gym	Gym / Fitness Center	Martial Arts Dojo	Outdoors & Recreation	Rugby Pitch	Skate Park	Soccer Field	Sports Club	Tennis Court	Yoga Studio
0	Bishop's Tachbrook	0	0	0	0	0	0	1	0	0	0	0	0	0
1	Bishop's Tachbrook	0	0	0	0	0	0	0	0	0	1	0	0	0
2	Bishop's Tachbrook	0	0	0	0	0	0	0	0	0	1	0	0	0
3	Budbrooke	0	0	0	0	0	0	0	0	0	1	0	0	0

In accordance with the use of Foursquare data in earlier Labs, the decision to use KMeans clustering to characterize the geographical areas in the analysis was made. No elbow testing was undertaken to determine the optimal number of clusters as this would surely have varied from one set of user parameters to another. For simplicity, '3' was chosen in the hope that a generic High/Medium/Low ward-suitability of groupings would emerge.

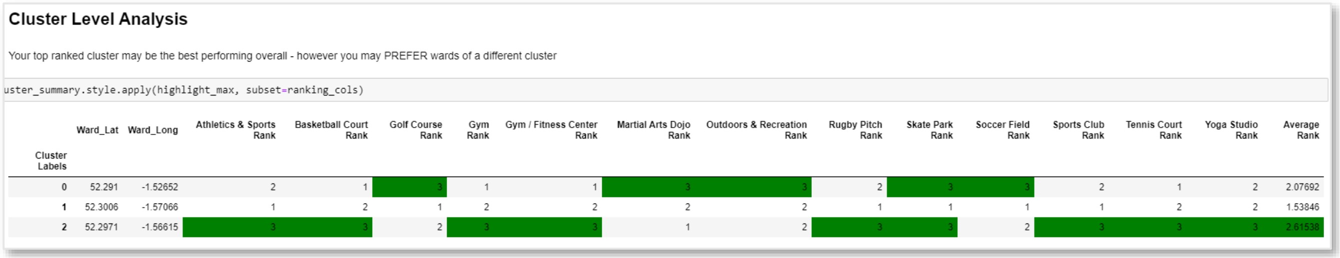
Results

For a given set of parameters (area: “Warwick”, UK ; category: “Athletics & Sports”) the output map was generated.



Above: output map of clustering. Note one cluster group is coloured in light green and is difficult to spot.

In order to assist in the interpretation of cluster meaning, the map was supplemented by some summary data. Rather than provide mean numbers of venues by cluster, the ranking of each cluster by feature was provided.



Cluster Level Analysis

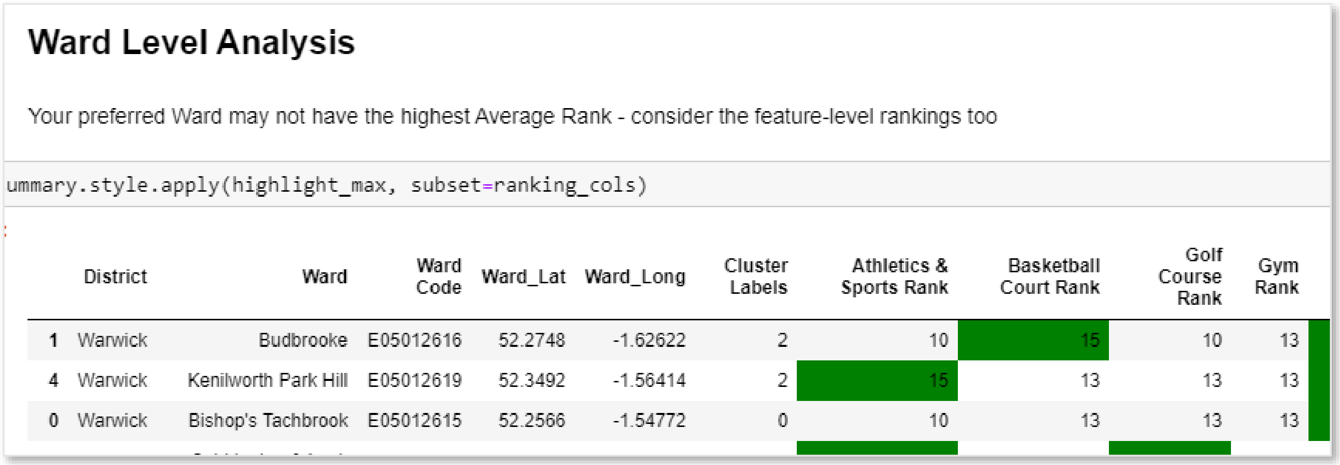
Your top ranked cluster may be the best performing overall - however you may PREFER wards of a different cluster

```
uster_summary.style.apply(highlight_max, subset=ranking_cols)
```

Cluster Labels	Ward_Lat	Ward_Long	Athletics & Sports Rank	Basketball Court Rank	Golf Course Rank	Gym Rank	Gym / Fitness Center Rank	Martial Arts Dojo Rank	Outdoors & Recreation Rank	Rugby Pitch Rank	Skate Park Rank	Soccer Field Rank	Sports Club Rank	Tennis Court Rank	Yoga Studio Rank	Average Rank
0	52.291	-1.52652	2	1	3	1	1	3	3	2	3	3	2	1	2	2.07692
1	52.3006	-1.57066	1	2	1	2	2	2	2	1	1	1	1	2	2	1.53846
2	52.2971	-1.56615	3	3	2	3	3	1	2	3	3	2	3	3	3	2.61538

Each feature received a rank and the highest performing cluster was highlighted green, by feature. In addition to this, the average rank across features was calculated and highlighted in the far right column. This might indicat the best performing cluster for the user – however, by including feature rankings, the user might determine that a different cluster fits their specific requirements. Using the example above, Cluster 2 performs best (bigger ranks are better). However, if the user is particularly interested in playing golf, soccer or taking up a martial art, then Cluster 0 locations may be best. Cluster 1 performs worst overall and is rank leader in no feature categories, so may be considered a poor fit for the user.

The user is also supplemented with Ward summary data, however this does not leverage any machine learning application. In the same way, features are ranked and top performers are highlighted.



Ward Level Analysis

Your preferred Ward may not have the highest Average Rank - consider the feature-level rankings too

```
ummary.style.apply(highlight_max, subset=ranking_cols)
```

	District	Ward	Ward Code	Ward_Lat	Ward_Long	Cluster Labels	Athletics & Sports Rank	Basketball Court Rank	Golf Course Rank	Gym Rank
1	Warwick	Budbrooke	E05012616	52.2748	-1.62622	2	10	15	10	13
4	Warwick	Kenilworth Park Hill	E05012619	52.3492	-1.56414	2	15	13	13	13
0	Warwick	Bishop's Tachbrook	E05012615	52.2566	-1.54772	0	10	13	13	13

Discussion

During the course of testing and fine tuning the notebook, I realized that the use of ward-level geography was of mixed value. It may be of most use in densely populated urban areas but does not scale well to rural areas where a single data point with its cluster label might have a radius of 10km and a user might have been looking for a lower level analysis of that entire area itself. A further development, therefore, would be to generate a lower level geographical analysis. For example, all UK postcodes are in the following format: XXXX XXX or XXX XXX. The first one or two characters indicate a larger postal area. For example, Warwick postcodes all begin CV as they belong to the larger Coventry delivery area (CV). The number which follows the first two letters further splits this area into subgroups. In fact, some parts of the Warwick area have a postcode beginning CV31 which indicates how many subdivisions there can possibly be to a postal area. Following this first geographical declaration is a space and another number and then two characters. That first number is yet another division of postal area, with the last two characters merely identifying the street within the area. It could strongly be argued that a better geography for this analysis would be 'local postal area' and this could be wrangled by extracting the substring 'XXXX X' (XX) or 'XXX X' (XX) from any given postcode. This would be more granular and would scale better than Ward as it is not population-derived.

A second development might be to set a radius value in the Foursquare API call which represents 'venues in walking distance from search area' rather than 'venues IN search area'. In reality, the best place for a contractor to live in might be an neighborhood which straddles two competing areas as he or she can then walk reasonably into both areas rather than having to specifically pick and remain in one of those areas. As such, the 'RADIUS' variable can be configured so as to represent 'reasonable walking distance' rather than trying to choose a value which means areas do not overlap to avoid double counting. In combination with the development proposed above, venues would ultimately serve several 'local postal areas' and the best area would be the one inside the catchment of as many interesting venues as possible.

There is definitely scope to add additional datasets to the clustering analysis undertaken here, with, say, ‘average rental value’ by area being added as a feature.

Conclusion

It has been satisfying to complete a project in its entirely from start to finish, although I was not particularly interested in using the Foursquare API again. It may be that I do not yet have the experience to have seen other uses for it but undertaking a second neighborhood clustering exercise was not particularly interesting after completing the Toronto lab the week before.

It was useful, however, to challenge myself to build something dynamic that would respond to two separate user selections: search area and venue categories. I also had to overcome several unexpected challenges from the outset: building a loop that used the json response to create a hierarchy table and in adapting the lab function that used the explore endpoint to use the search endpoint with its differing response format.

The value and application of KMeans clustering is now proven to me however and I have greater confidence in using and applying this technique elsewhere.

The tool as it stands has value in assessing different areas of a town or city with respect to the venues within.