EATING MY WAY THROUGH EUROPE

A QUEST TO DISCOVER THE BEST FOOD-CITY TO RIVAL CHICAGO

Bart Onkenhout

Submitted as a Capstone Project for the IBM Data Science Certification

22 April 2019

INTRODUCTION

Background

In the previous capstone assignment in Week 3, I spent a significant amount of time exploring the city of **Toronto**, enriching the neighborhoods with venue data from the Foursquare API. A nice isolated exercise to flex some of the Python muscles that have lain long-dormant. However, Toronto is really more of a proof-of-concept -- barring some out-of-the-blue job offer in Canada I don't think I'll be moving up to the Great White North. Firstly, I'm not really a cold-weather person; and secondly, I'm not sure I have enough points to qualify for an ExpressEntry visa ;). Canada -- you're lovely, but you're also buried in snow 80% of the time, so no thanks!

So perhaps it's time to kick our analysis up another notch. In the near future, I plan to move to Europe for a work assignment. However, my employer has headquarters in several European cities. So which work assignment should I apply for? The one in Frankfurt? Barcelona? Perhaps one in Amsterdam? I'd like to find the most similar cities in Europe and do the following:

1) find the cities in which my employer has headquarters,
2) enrich this set of cities with data from the Foursquare API,
3) see how this compares with my current city of Chicago in terms of venue diversity,
4) make a decision based on this data, and
5) show *you*, my lovely audience (and classmates) my reasoning!

Importance

So who really cares? Well, there's a couple of reasons why this is important.

First, it's always good to make **evidence-driven decisions**. With increasing globalization, it's important to spend one's time wisely and get the most out of every place we travel to. However, there's an increasing problem of **information overload** -- known in psychology as the Paradox of Choice. I'm a bit on the older side and no longer have the luxury of 'finding myself' while backpacking for months through the Continent by rail like a bohemian. No--I've got *bills! Student loans!* A bedtime of *9:30 PM!* This means I'm looking for maximum payoff with my time. And third, I **really, really** like food, so I want to find a place that's like Chicago in its diversity of cuisine offerings. (For my fellow students who are not familiar with the US, Chicago has a reputation as a mecca for food!)

So, because I definitely can't *eat* my way through Europe I'll need to be able to filter out some of the places I definitely *don't* want to go. This will help me to continuously refine my selection until I have a shortlist of, say, three cities that I can choose to focus on before settling on a final decision. And data science will help me to arrive at this shortlist in a systematic, reproducible way -- unlocking the secrets of the Foursquare API in a way that we never could with Excel!.

Problem Statement

Which city/cities in Europe should I pick for my next work rotation so that I still have similar food options to Chicago?

DATA

In a nutshell, the data I'll need is fairly straightforward in concept. I want:

- a **list of European cities** in which my employer has offices as a base comparison set
- **geolocation data** of each city so we can plot everything on a map
- **Foursquare API data** for looking up venues in each city
- **GeoJSON shapes** of each city so we can use a GeoJson layer to outline each city on the map

List of European cities

This list is obviously only available to employees of my company. Luckily, we have a central company directory that has the following data for each office:

- `LOCATION`: Office's official location name in our directory
- `CITY` : City of the office
- `COUNTRY` : Country of the office

Fairly straightforward. I'll be loading each of these as a CSV file into a dict keyed on `LOCATION`, with fields like in the following example:

```
offices['location']: {'city':'Frankfurt', 'country':'Germany'}
```

Features:

- city
- country

Geolocation data

This one is a bit tougher, since Google made their Google Maps API a premium product. However, according to Google there are USD 200 in free credits allowed every month, and the prices start at only USD 5 per 1000 requests. So if I sign up for a free account, I can make up to 40,000 requests before I need to start paying. This is one possible route.

Another possible route would be to use the OpenStreetMaps alternative API, Nominatim. I do recall from previous usage that OSM can be finicky in which inputs it does and does not accept, so geocoding our European cities may prove difficult. Depending on the number of cities available, I may want to do this offline in Excel by hand-querying the Google Maps website, rather than using the API.

In any case, the Nominatim API can be queried using the following syntax:

`https://nominatim.openstreetmap.org/reverse?format=jsonv2&lat=-34.4391708&lon=-58.7064573`

This will return a JSON object containing geographic information on the location(s) at the specified latitude and longitude (in this example, the Aramburu highway in Argentina). The JSON object looks like this:

```
{

  "place_id":"134140761",

  "licence":"Data © OpenStreetMap contributors, ODbL 1.0.
http:\/\/www.openstreetmap.org\/copyright",

  "osm_type":"way",

  "osm_id":"280940520",

"lat":"-34.4391708",

  "lon":"-58.7064573",

  "place_rank":"26",

  "category":"highway",

  "type":"motorway",

  "importance":"0.1",

  "addresstype":"road",
  "display_name":"Autopista Pedro Eugenio Aramburu, El Triángulo, Partido de
Malvinas Argentinas, Buenos Aires, 1.619, Argentina",

  "name":"Autopista Pedro Eugenio Aramburu",

  "address":{

    "road":"Autopista Pedro Eugenio Aramburu",

    "village":"El Triángulo",

    "state_district":"Partido de Malvinas Argentinas",

    "state":"Buenos Aires",

    "postcode":"1.619",

    "country":"Argentina",

    "country_code":"ar"

  },

  "boundingbox":["-34.44159","-34.4370994","-58.7086067","-58.7044712"]

}
```

Features:

- city latitude & longitude
- addresses
- geographic location
- bounding box

Foursquare API data

This one is fairly straightforward. I've already signed up for an API key at http://developer.foursquare.com, which will allow me to make a number of calls (at no charge) to get the information I need from the various endpoints that are exposed by the API. The most interesting information I'll want is the VENUE DATA within city limits.

We will use the `explore` endpoint to get venue recommendations based on a latitude and longitude passed in the `ll` parameter. However, according to the documentation, the non-premium API only returns 50 VENUES at a time. I'll have to use the `offset` parameter to page through the different results.

As an example, here is a list of 10 venues returned near the Willis Tower (formerly Sears Tower) in Chicago, IL (41.8789° N, 87.6359° W):

```
https://api.foursquare.com/v2/venues/explore?&client_id={CLIENT_ID}&client_secret={CL
IENT_SECRET}&v={API_VERSION}&ll=41.8789,-87.6359&radius={radius}
```

Where `CLIENT_ID` and `CLIENT_SECRET` are the API ID and API key, `API_VERSION` is the version date of the API, `ll` is the latitude and longitude of the location I want to search around in a radius of `radius` meters.

Features:

- nearby venues
- venue latitude & longitude
- venue type
- venue rating

GeoJSON shapes

Luckily, the European Commission has an excellent website that may help us with this. In its Urban Audit section, we can find `.shp` boundary files for each city that falls under the EC's watchful eye. I'll use this article to convert the `.shp` files into GeoJSON files, as I did for Part 3 of the Week 3 assignment, using the open-source QGIS software.

Features:

- GeoJSON polygons
- cloropleth zones

METHODOLOGY

Data extraction, transformation, and loading

First, I scraped the necessary data from various sources and cleaned/scrubbed everything into the required format. Please see the accompanying notebook for more information on how this was done.

Initially, I had many directions I wanted to go in, including weighting each venue by its average number of ratings and incorporating this data into the final decision. However, because the non-premium Foursquare API data is so limited in what it can do, I had to settle on performing a similar analysis as the one in Week 3, except with differing parameters.

Exploratory analysis and error-checking

As discussed in the accompanying notebook, I used the Nominatim API to query OpenStreetMaps for each city's geodata. One of the data points returned is the city's *bounding box*. This bounding box represents either one of two things: 1) a GeoJSON polygon containing the necessary information to draw the legal city limits on a map, or 2) an array containing four numbers that represent two (latitude, longitude) tuples. I used the first, if present, to plot the city's complete limits on the Folium map later on (see section RESULTS). I used the latter to determine the geographic boundaries within which to search using the Foursquare API. I then transformed this data into a Polygon object to be used as a substitute if any missing GeoJSON in the OSM database.

Important about the bounding box data is that it defines a search grid with a particular area. Since the OSM database can sometimes be finicky, I needed to first compare the search area of each city to determine whether the Foursquare API would accept it as a search parameter. To do this, I examined the quartiles and z-score of the area. (For details on how I calculated the area, please see the accompanying notebook.)

First, I examined the interquartile ranges of each city's area. I noted that there was one huge outlier:

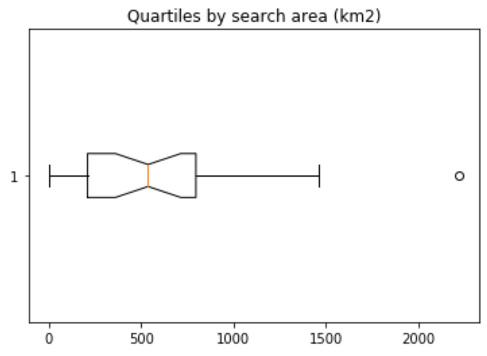
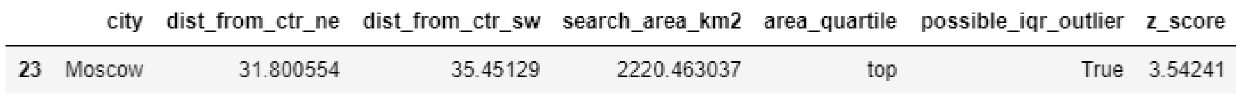


Figure 1: Boxplot of quartiles by search grid area

This outlier corresponded to the city of **Moscow**, which is known to be a very large metropolitan area. Since many European cities tend to be more compact, it made intuitive sense that Moscow would be flagged as an outlier compared to only Western Europe, which is where the majority of my company's European offices are. However, if I were to run the same analysis on all the world's cities, Tokyo and Shanghai would be more likely to be flagged as outliers.



	city	dist_from_ctr_ne	dist_from_ctr_sw	search_area_km2	area_quartile	possible_iqr_outlier	z_score
23	Moscow	31.800554	35.45129	2220.463037	top	True	3.54241

Table 1: Moscow as an outlier, based on search area (km^2)

When looking at a histogram of the distribution of all the search areas and examining datapoints with z-scores more than one standard deviation outside the mean, I made another two important insights. First, Chicago (the base point of comparison) was flagged as an outlier. This was clearly a measurement error, since Chicago is both an American city as well as the base point of comparison.

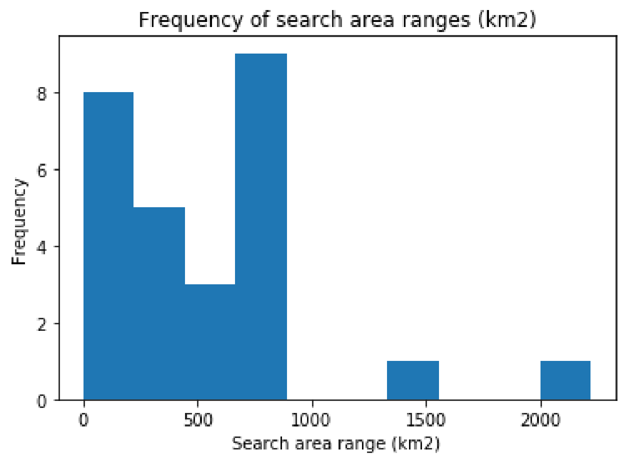
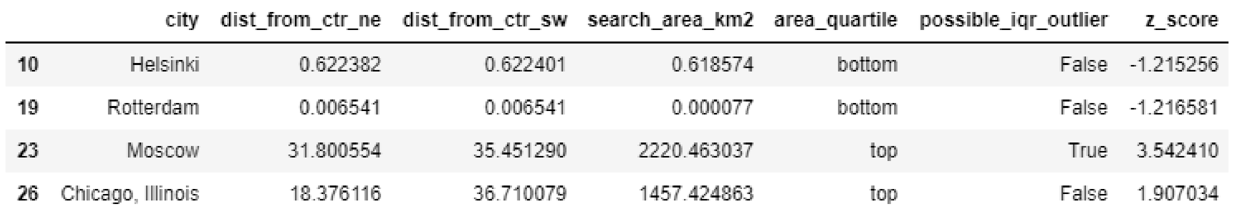


Figure 2: *Histogram of frequency by search grid area (km^2)*

Second, a quick scan of the z-scores for each city's search area exposed Helsinki and Rotterdam as outliers, next to Moscow and Chicago.



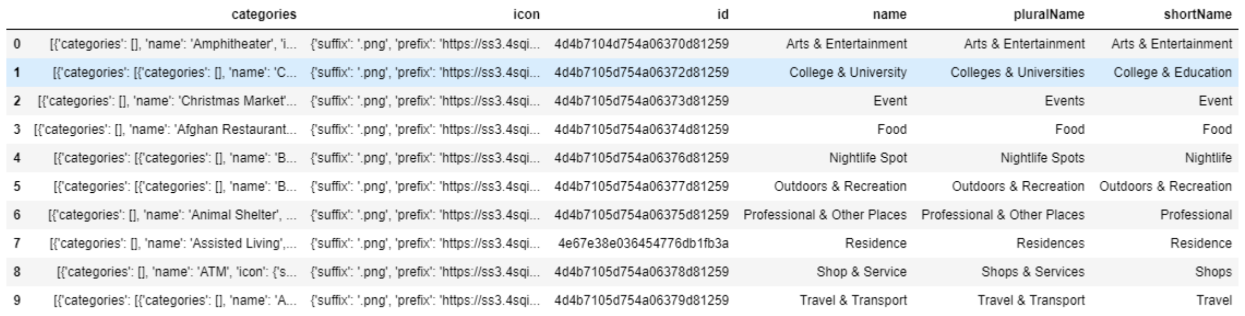
	city	dist_from_ctr_ne	dist_from_ctr_sw	search_area_km2	area_quartile	possible_iqr_outlier	z_score
10	Helsinki	0.622382	0.622401	0.618574	bottom	False	-1.215256
19	Rotterdam	0.006541	0.006541	0.000077	bottom	False	-1.216581
23	Moscow	31.800554	35.451290	2220.463037	top	True	3.542410
26	Chicago, Illinois	18.376116	36.710079	1457.424863	top	False	1.907034

Table 2: Helsinki, Rotterdam, Moscow, and Chicago as outliers, based on z-score

After poking around on Google Maps and looking at the Wikipedia pages for Helsinki and Rotterdam, I came to the conclusion that both Rotterdam and Helsinki have errors in their OpenStreetMaps bounding box data. I therefore decided to drop them both. I also decided to keep Moscow and Chicago, since there is no obvious error with Moscow besides the fact that its mere size makes it an outlier, and since Chicago is the base city I am comparing all other cities in the dataset to. (After all, I want to find the European city with the *most similar food scene* to Chicago where my company has offices.)

Narrowing down the Foursquare API data

Since Foursquare returns results on so many venues, I thought it was important to narrow down the search criteria. Most of the API documentation indicates that 100 venues is the maximum number of hits returned in a non-premium call. I first queried the top-level categories that were available to get the 'Food' category, since that is the only criterion I'm interested in for my analysis. I then grouped all the results by city and concatenated this with the office locations DataFrame. (For more technical details on how I did this, please see the accompanying notebook.)



	categories	icon	id	name	pluralName	shortName
0	[{'categories': [], 'name': 'Amphitheater', 'i...	{'suffix': '.png', 'prefix': 'https://ss3.4sqi...	4d4b7104d754a06370d81259	Arts & Entertainment	Arts & Entertainment	Arts & Entertainment
1	[{'categories': [{'categories': [], 'name': 'C...	{'suffix': '.png', 'prefix': 'https://ss3.4sqi...	4d4b7105d754a06372d81259	College & University	Colleges & Universities	College & Education
2	[{'categories': [], 'name': 'Christmas Market'...	{'suffix': '.png', 'prefix': 'https://ss3.4sqi...	4d4b7105d754a06373d81259	Event	Events	Event
3	[{'categories': [], 'name': 'Afghan Restaurant...	{'suffix': '.png', 'prefix': 'https://ss3.4sqi...	4d4b7105d754a06374d81259	Food	Food	Food
4	[{'categories': [{'categories': [], 'name': 'B...	{'suffix': '.png', 'prefix': 'https://ss3.4sqi...	4d4b7105d754a06376d81259	Nightlife Spot	Nightlife Spots	Nightlife
5	[{'categories': [{'categories': [], 'name': 'B...	{'suffix': '.png', 'prefix': 'https://ss3.4sqi...	4d4b7105d754a06377d81259	Outdoors & Recreation	Outdoors & Recreation	Outdoors & Recreation
6	[{'categories': [], 'name': 'Animal Shelter', ...	{'suffix': '.png', 'prefix': 'https://ss3.4sqi...	4d4b7105d754a06375d81259	Professional & Other Places	Professional & Other Places	Professional
7	[{'categories': [], 'name': 'Assisted Living',...	{'suffix': '.png', 'prefix': 'https://ss3.4sqi...	4e67e38e036454776db1fb3a	Residence	Residences	Residence
8	[{'categories': [], 'name': 'ATM', 'icon': {'s...	{'suffix': '.png', 'prefix': 'https://ss3.4sqi...	4d4b7105d754a06378d81259	Shop & Service	Shops & Services	Shops
9	[{'categories': [{'categories': [], 'name': 'A...	{'suffix': '.png', 'prefix': 'https://ss3.4sqi...	4d4b7105d754a06379d81259	Travel & Transport	Travel & Transport	Travel

Table 3: Category ID by Foursquare category (see columns 'id' and 'name')

Lastly, I also took a look at a heatmap of the top 20 most common venues in each city to see how each city compared in terms of venue variety. Results are as one might expect – Italian cities have lots of Italian restaurants, French cities have lots of French restaurants, lots of cafés in each city, etc. So there's nothing really out of the ordinary about the Foursquare data, which is great because it indicates a high quality of the base data to work with. Which brings me one step closer to a solid analysis.

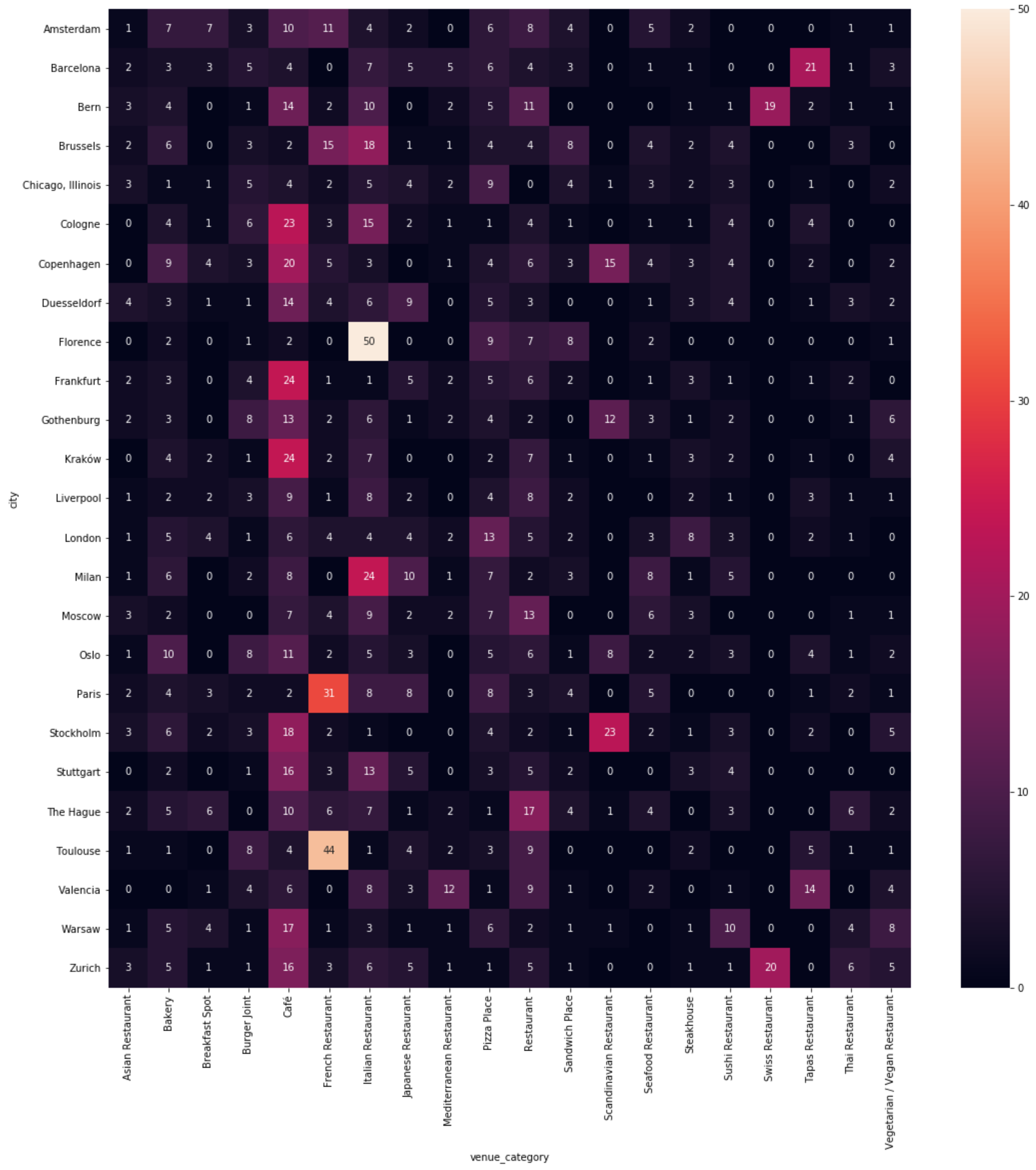


Figure 3: *Heatmap of top 20 venue categories by city*

Classification

The objective is to find a shortlist of European cities that are most similar to Chicago in terms of food scene, which makes this a *classification* problem, so all the models we consider will need to create a discrete group assignment for each data point.

First, I one-hot encoded the Foursquare-plus-office DataFrame. Then, I took the arithmetic mean for each venue type to get a one-hot encoded average in each city relative to the number of venues in each category. Note that calculating purely on the number of venues underrepresents smaller cities—*but only when searching the same surface area size for each city*. Since the Foursquare API limits me to only 100 hits per query, and since I'm searching a bounding box that has an area relative to each city's size, it must therefore follow that I will get a representative sample for each city because the *number of venues returned per square kilometer* increases as the city size gets smaller. Therefore, all I need to do is compensate for any underrepresented cities by multiplying its one-hot encoded average by the underrepresentation factor.

In this case, by looking at the histogram of cities that had less than 100 venues, I found that Bern, Switzerland was underrepresented by a factor of 9%. I therefore multiplied the one-hot encoded average of Bern by 109%.

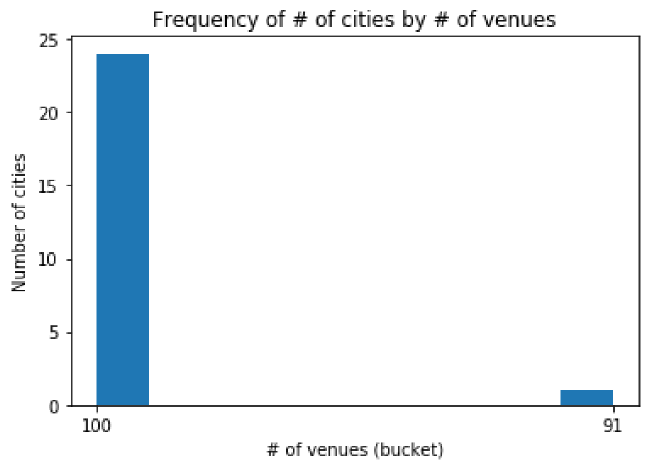
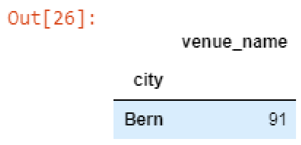


Figure 4: Frequency of cities with less than 100 venues returned from Foursquare

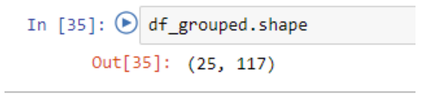


Out[26]:

	venue_name
city	
Bern	91

Table 4: List of cities with less than 100 returned venues

In the end, my adjusted, one-hot encoded DataFrame is of shape 25 rows by 117 columns, with the columns representing the features.



```
In [35]: df_grouped.shape
Out[35]: (25, 117)
```

Table 5: Compensated one-hot encoded DataFrame with 25 datapoints and 116 features

I considered using a number of different models to classify all the cities, including SVM, binary trees, and DB SCAN. However, in the end I settled on k-means for a number of reasons:

- Because the data is one-hot encoded, I have 116 features. This means that my dataset has many dimensions, so any dimensionality reduction will be computationally intensive.
- I expect my dataset not to be so topographically complex that SVM or DB SCAN will need to be used.
- Binary trees might be too discrete and not allow for fuzziness in the classification. It may also be subject to over-fitting with so many features.
- K-means is the fastest, most computationally economical model for general clustering classification.

After running a k-means model on the one-hot encoded data, I extracted the cluster labels and substituted these back into their representative city. I then supplemented this DataFrame with GeoJSON data scraped from OpenStreetMaps. Where the GeoJSON boundaries were not available, I transformed the bounding box into a GeoJSON rectangle instead. Finally, I encoded the classification data and cluster labels into a GeoJSON string, which I displayed as tooltips over each city on a Folium map. (For more details on how I did this, please see the accompanying notebook).

RESULTS

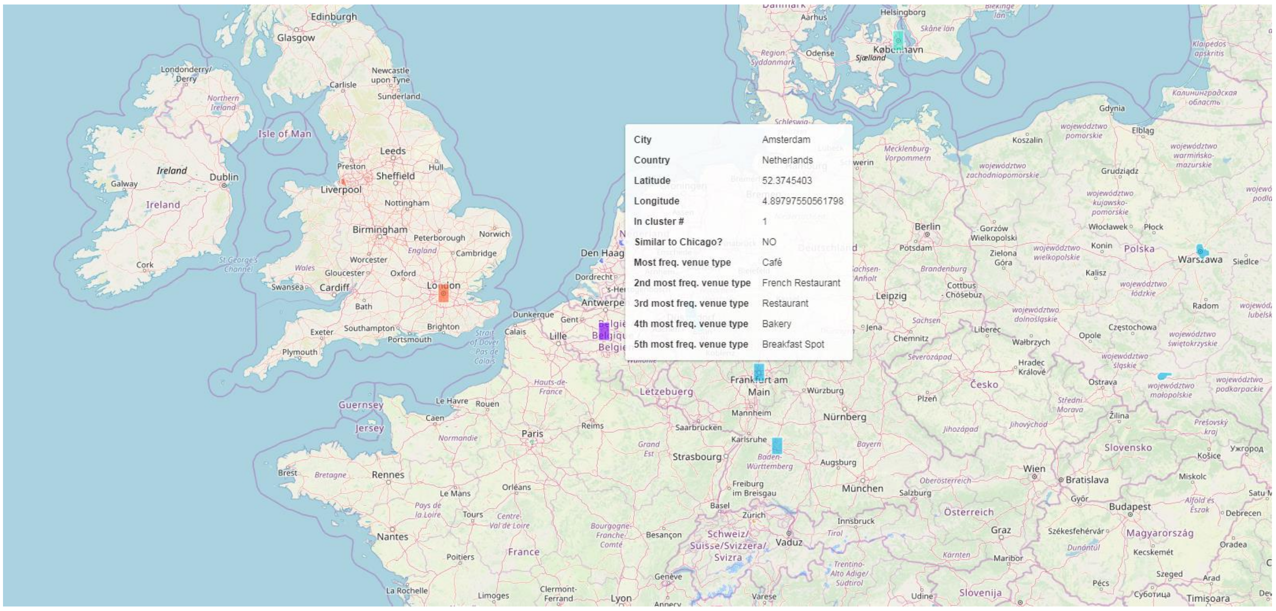
Below is the table of cities that are most similar to Chicago in terms of food venues, according to my model.

```
for e in list(df_map[df_map['cluster'].isin([chicago_cluster])]['city']): print(e)

    Liverpool
    London
    Chicago, Illinois
```

Table 6: List of cities that are most similar to Chicago, food-wise

Here a screenshot of the Folium map used to visually graph my results:



City	Amsterdam
Country	Netherlands
Latitude	52.3745403
Longitude	4.89797550561798
In cluster #	1
Similar to Chicago?	NO
Most freq. venue type	Café
2nd most freq. venue type	French Restaurant
3rd most freq. venue type	Restaurant
4th most freq. venue type	Bakery
5th most freq. venue type	Breakfast Spot

The cities are all colored according to their associated clusters. Hovering over each city will reveal the top 5 most frequent venue types for that city, as well as whether it is similar to Chicago according to the k-means model. This allows a handy way to visually explore the data. (Again, please see the notebook for more details.)

DISCUSSION

According to the model, I should focus on **London** or **Liverpool** for my next assignment. However, instead of blindly following the algorithm and packing our bags, it's important to check if this makes realistic sense. An investigation of the data shows that this may indeed be the case. In various runs of the model, I have seen different results pop up. This is because the k-means model initializes a centroid coordinate randomly, then moves the centroid while trying to find the least mean distance for all points in the centroid. However, this type of clustering is still subject to a **vanishing gradient**, so a local optimum might not necessarily mean that we are getting a global optimum. Therefore, the k-means clustering algorithm is highly sensitive to the initial starting coordinates of each centroid. To refine this model in future, it may be necessary to try a different algorithm, such as DB SCAN or SVM.

Still, after running the model several times and re-initializing it with new data fetched from the Foursquare API, I have come to the conclusion that London is the single-most common city named in each run. That is—it is often and consistently named in the list of cities that the k-means algorithm finds are most similar to Chicago in terms of their food scene. This either means that there is a systemic bias in my modeling, or the two cities are indeed similar. However, on the surface, it appears that Chicago is actually quite similar to London. Both are major metropolitan areas, both are quite spread-out, and both cities have a very diverse population. On an intuitive level, it would make sense that the food scene might be the same.

However, it is important not to jump to conclusions, and to investigate further to see if there is a way to eliminate systemic bias, if any, from the model in order to arrive at true food matches for Chicago!

CONCLUSION

While additional research is needed, and while I must investigate further how to eliminate potential sources of systemic bias, an initial survey of the results finds compelling evidence that **London** is the next place I should look for a work assignment should I want to live in a European city with a food scene that is similar to Chicago.