Sales Territories for Manufacturer in Georgia

Data Science Capstone Project

James Cage
July 2019

Table of Contents

1 Introduction

This is my final project for Coursera's Applied Data Science Capstone class, which is the final step in IBM's Data Science Professional Certificate. The project assignment instructs students to "come up with a problem that you can use the Foursquare location data to solve."

The project I devised is to define sales territories for a B2B supplier. In this project I refer to a "restaurant equipment manufacturer" and give some details about that company, the market it participates in, and its business strategy. That company is not real. I am not revealing (or for that matter, concealing) any proprietary details about it, because it is a work of fiction and not based in any way on any real company.

I created this project to demonstrate my skills in data gathering, data cleaning (very important), machine learning, Python programming, and communicating the results through a report (this), a blog post, and a Jupyter notebook. While the problem described here is a work of fiction, the data, tools, methods, and constraints are entirely real. Nothing has been fudged or ignored in order to make the results more satisfying.

While this scenario is fictional, I am not. My background is in the industrial process control industry, where I worked as an engineer, product manager, and sales consultant. I have some background in inside and outside sales, but I do not pretend to have expert domain knowledge of the restaurant industry. I chose this project in large part because Foursquare data is dominated by information about restaurants.

1.1 Acknowledgements

Data in this analysis appears courtesy of Foursquare and its admirable commitment to providing free access to students and individual users. I would also like to thank RestaurantSupply.com of Tempe, Arizona and their inside sales team, which gave me a valuable overview of the sales process in the restaurant industry. Their help made this a more enjoyable and educational experience for me, but of course any mistakes in the scenario or the project's conclusions are entirely my own.

2 Business Problem

A (fictional) company that manufactures equipment and supplies for Asian restaurants plans to expand into Georgia in the United States. Different types of Asian restaurants use different types of equipment, but the company sells its products to all types. For example, it sells sushi display cases to Japanese restaurants and tandoor ovens to South Asian restaurants. The company wishes to define territories for its salespeople based on the type of restaurants and location in the state. A large concentration of Korean restaurants will hopefully be in a single territory, but that territory might also include a Vietnamese location that is not close to other Vietnamese restaurants.

Sales territories may overlap geographically but given all factors (including the restaurant's cuisine) all territories must be unique. A salesperson may cover Chinese restaurants in one part of the state,

while another salesperson would cover all other ethnicities (Japanese, Vietnamese, Korean, etc.) in the same area (assuming there are enough customers in this area to justify multiple salespeople).

The company plans to hire five "outside sales" people who will visit potential customers (restaurants) and sell them on the benefits of the company's products.

2.1 Audience

This analysis should be useful to the following:

- Business-to-business vendors: This analysis gives an example of how to define a customer base, gather data, and use machine learning to segment the base both geographically and by business needs.
- Companies that need a data-driven method to define sales territories.
- People in the restaurant industry who need information on concentrations of venues by type and location.

Data scientists and students faced with the following issues should also be interested:

- Georgia has over 3,000 Asian restaurants. The state covers over 150,000 square kilometers. With my account type, Foursquare limits requests to 100 returned venues in an area no larger than 10,000 square kilometers. This project demonstrates how to **efficiently gather thousands of Foursquare venues** over a very wide area, while **preventing missed venues and duplications**.
- The problem required clustering by geographic location and venue type (non-location) data. This project shows one way to use different types of factors in machine learning to obtain practical results.
- Like most data sets in the real world, the information gathered for this project contained mistakes. A Mediterranean restaurant may have been placed in the Asian category, or a restaurant that should be labeled "Japanese" may have been included in the general Asian category. This project demonstrates how to **clean data programmatically, using key-word searches** within venue categories.
- Finally, this project gives many examples of displaying data on maps, tables, and charts.

3 Data

I used data from Foursquare to create a database of Asian restaurants. Foursquare does not include South Asian restaurants in its "Asian" category, so those were added to the database using "Indian" and "Pakistani" category codes.

3.1 Obtaining Data

As noted in the Introduction, it was necessary to retrieve thousands of venues over a large area. The point-and-radius method of requesting Foursquare data (demonstrated in the class) was not adequate, as it either introduces gaps between the circles or creates duplicates due to overlaps. I obtained the

data using a bounding box method instead. Foursquare limits this both by the area covered (approximately 10,000 square kilometers) and the number of results returned (no more than 100). I wrote Python code to divide the geographic area into boxes that are small enough for Foursquare queries. The code does something similar when a request returns 100 results. This number indicates that there are probably more restaurants in the defined area and that the request was trimmed to maximum allowed value. My code divides the box into four smaller boxes and submits a new request for each one. This continues (using recursion) until all requests return 99 or fewer results, thereby capturing all the desired restaurants in the area.

The figure below shows an example of the requests (represented by boxes of various sizes) and the number of results (represented by the color of each box). Note that there are no gaps between the boxes and no overlap. The "drop_duplicates" method in Pandas indicate no duplicate venues are included when this method is used.

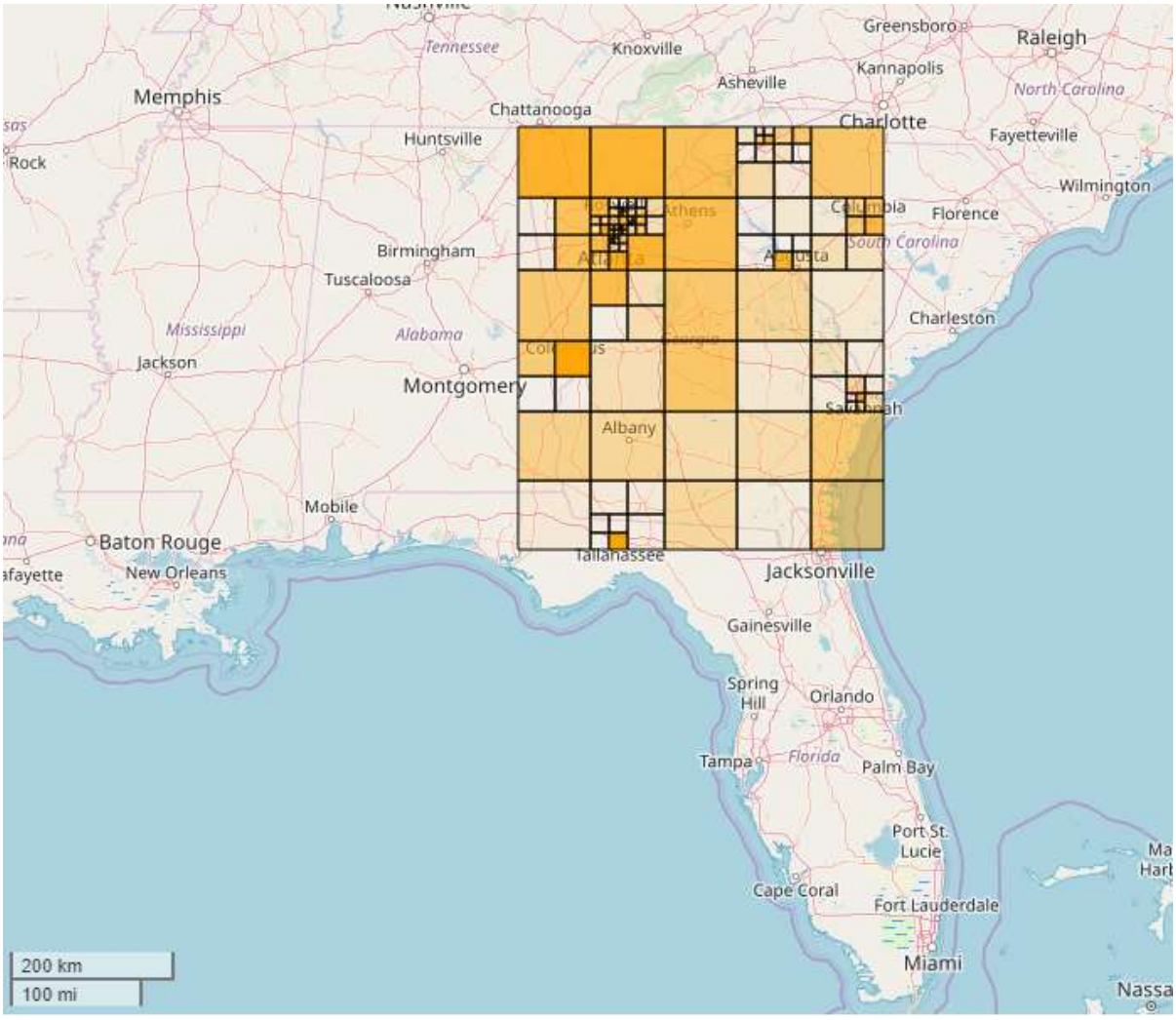

Figure 1: Bounding box Foursquare requests using recursion in Python

The next figure shows some of the smallest boxes in the picture above. These boxes are less than one mile on a side. Obtaining this data using equal-sized boxes (or equal-sized circles using the method

demonstrated in our class) would require **over 50,000 requests.** My method uses **168 requests** and **executes in less than 90 seconds**.

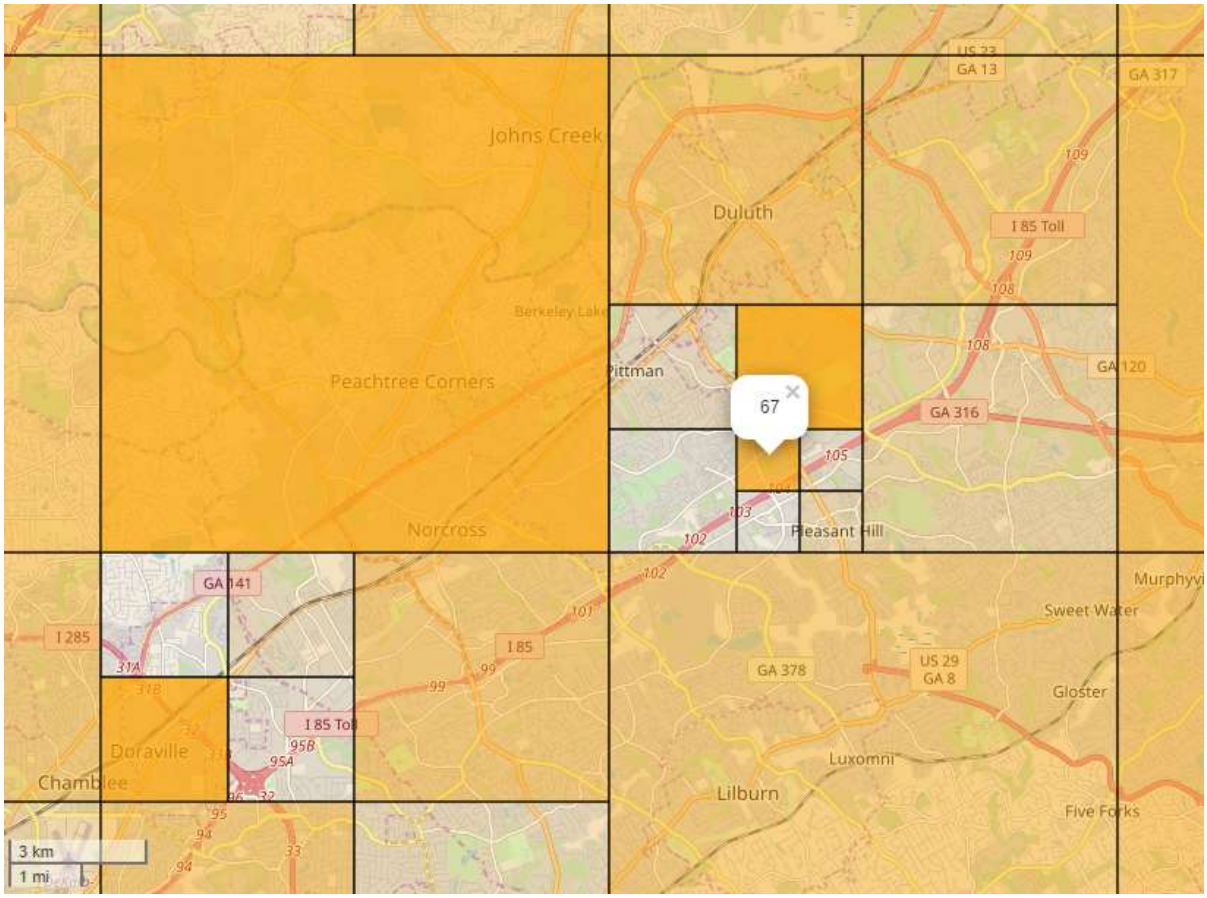

Figure 2: Detail showing size of smallest bounding boxes (scale at lower left)

3.2 Cleaning & Scaling Data

Significant work was required to clean the data. I wrote code to reassign results in incorrect categories and to consolidate categories to a manageable set.

Asian restaurants are assigned to three Foursquare categories: “Asian”, “Indian”, and “Pakistani”. “Japanese” and “Korean” are subcategories of “Asian”, for example. However, many restaurants are lumped into the “Asian” category that clearly belong to a sub-category (based on the name of the restaurant). Some restaurants are assigned to the wrong category. My code reassigns restaurants where possible based on keywords in the restaurant’s name. Other restaurants offer a mix of types and remain in “Asian”.

```
[ ]  # Define terms commonly found in restaurant names for each country of origin.

     china_terms = ['China', 'Chinese', 'Wok', 'Hong Kong', 'Panda', 'Peking',
                    'Beijing', 'Great Wall']
     japan_terms = ['Japanese', 'Tokyo', 'Japan', 'Osaka',
                    'Shogun', 'Fuji', 'Sumo', 'Ichiban', 'Kobe', 'Sakura', 'Ramen',
                    'Teriyaki', 'Ninja', 'Shabu']
     korea_terms = ['Korea', 'Gogi']
     thailand_terms = ['Thai', 'Bangkok']
     vietnam_terms = ['Pho', 'Saigon', 'Viet', 'Banh Mi']
     indopak_terms = ['India', 'Bombay', 'Biryani', 'Naan', 'Masala']

     term_list = [['Chinese Restaurant',china_terms],
                  ['Japanese Restaurant',japan_terms],
                  ['Korean Restaurant', korea_terms],
                  ['Thai Restaurant', thailand_terms],
                  ['Vietnamese Restaurant', vietnam_terms],
                  ['Indo-Pak Restaurant', indopak_terms]]
```

Figure 3: Programmatic approach to data cleaning

Foursquare provides latitude and longitude data, which I scaled to give accurate distance information.

4 Methodology

This problem requires segmenting a market both by geographic data (numeric) and by the restaurant's cuisine (categorical data). Unfortunately, mixing data of different types in a single machine learning is problematic. Categorical data can made numeric (by using one-hot analysis, for example), but how should the machine learning algorithm interpret combining this with X-Y location data? It is easy to imagine the algorithm grouping Indian restaurants that share a common latitude while their longitudes (and hence the physical distance between them) varies widely.

After examining the problem (including the geography of Georgia and the distribution of restaurants in the state) I chose to use geographic grouping to create territories outside the Atlanta area, and grouping by restaurant type within the Atlanta area. I used k means as the grouping algorithm.

I first attempted to create territories using just the cuisine data. I chose the k means algorithm to perform the grouping, and converted the type of restaurant (stored as text strings in a single column) to separate columns for each type containing a "1" or "0" (one-hot encoding). The number of clusters was set to the number of salespeople the customer planned to hire (in this case, five).

Grouping solely on restaurant type produced unwieldy territories. Most sales territories only serve one kind of restaurant, but every territory covers the entire state. Travel time would harm productivity in this case. This analysis also uncovers a charming attribute of k means clustering - its tendency to produce groups of widely different sizes.

Table 1: Territories based on restaurant type alone

Territory:	yellow	purple	green	red	blue	Total
Asian Restaurant	0	0	0	477	0	477
Chinese Restaurant	0	1216	0	0	0	1216
Indo-Pak Restaurant	0	0	0	0	276	276
Japanese Restaurant	0	0	798	0	0	798
Korean Restaurant	215	0	0	0	0	215
Thai Restaurant	202	0	0	0	0	202
Vietnamese Restaurant	121	0	0	0	0	121
Total	538	1216	798	477	276	3305

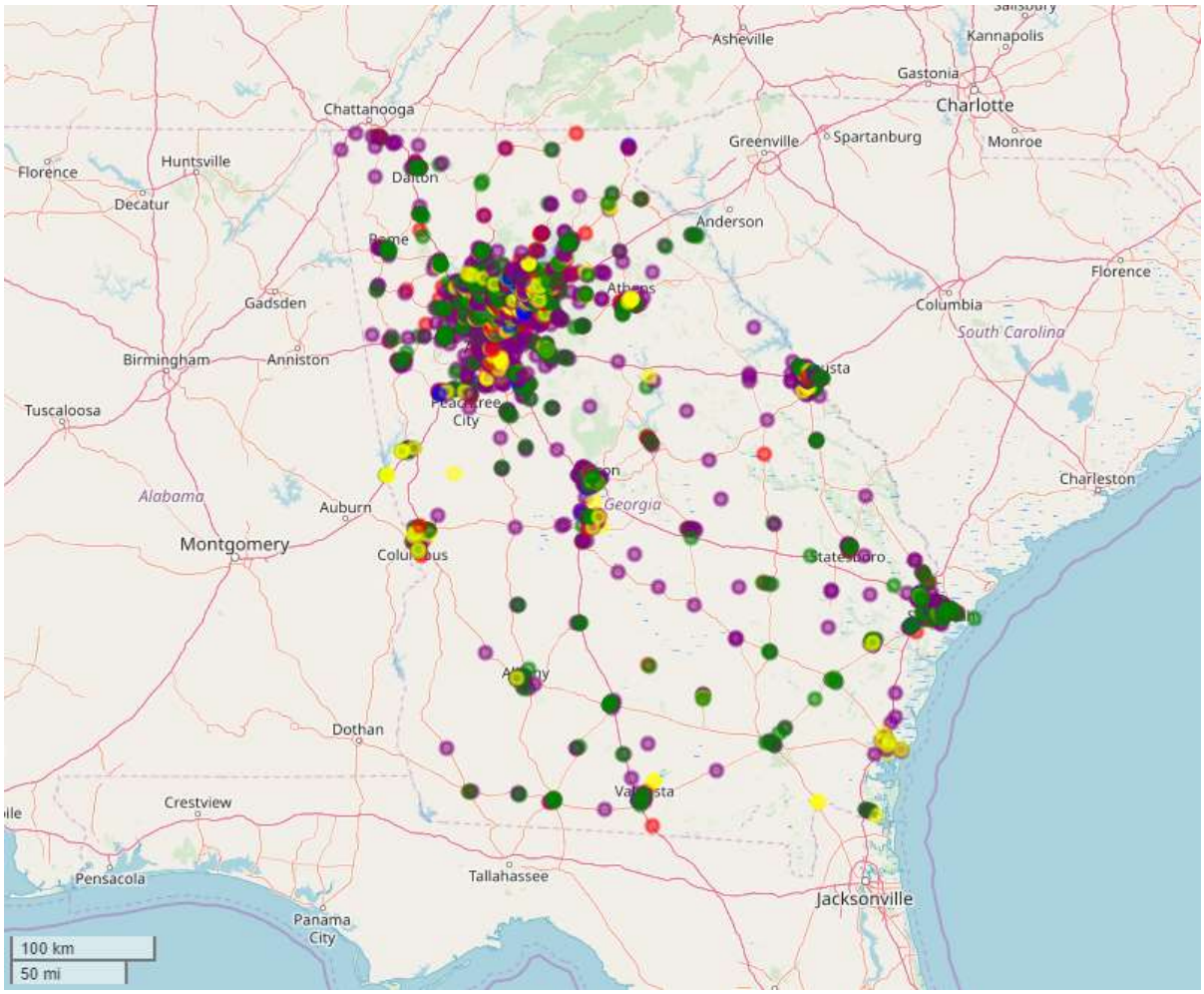

Figure 4: Territories chosen based on restaurant type only

Next I attempted to create the territories using location data (latitude and longitude) alone. A one-degree change in latitude represents a different distance than a one-degree change in longitude. (In Georgia, lines of latitude are roughly 30% shorter than lines of longitude.) I rescaled latitude and longitude using the average distances covered by each before performing the analysis

Grouping the restaurants physically yielded sales territories that include all types of restaurants. This increases the load on the salespeople, as each one must learn to service all types of restaurants. Opportunities to specialize are lost. The high concentration of restaurants in the greater Atlanta area also ensures that the number of restaurants in each territory is uneven.

Table 2: Territories based on location only

Territory:	yellow	purple	green	red	blue	Total
Asian Restaurant	15	39	386	13	24	477
Chinese Restaurant	52	108	859	69	128	1216
Indo-Pak Restaurant	4	9	250	5	8	276
Japanese Restaurant	42	89	559	39	69	798
Korean Restaurant	0	5	188	6	16	215
Thai Restaurant	5	18	160	4	15	202
Vietnamese Restaurant	1	6	107	1	6	121
Total	119	274	2509	137	266	3305

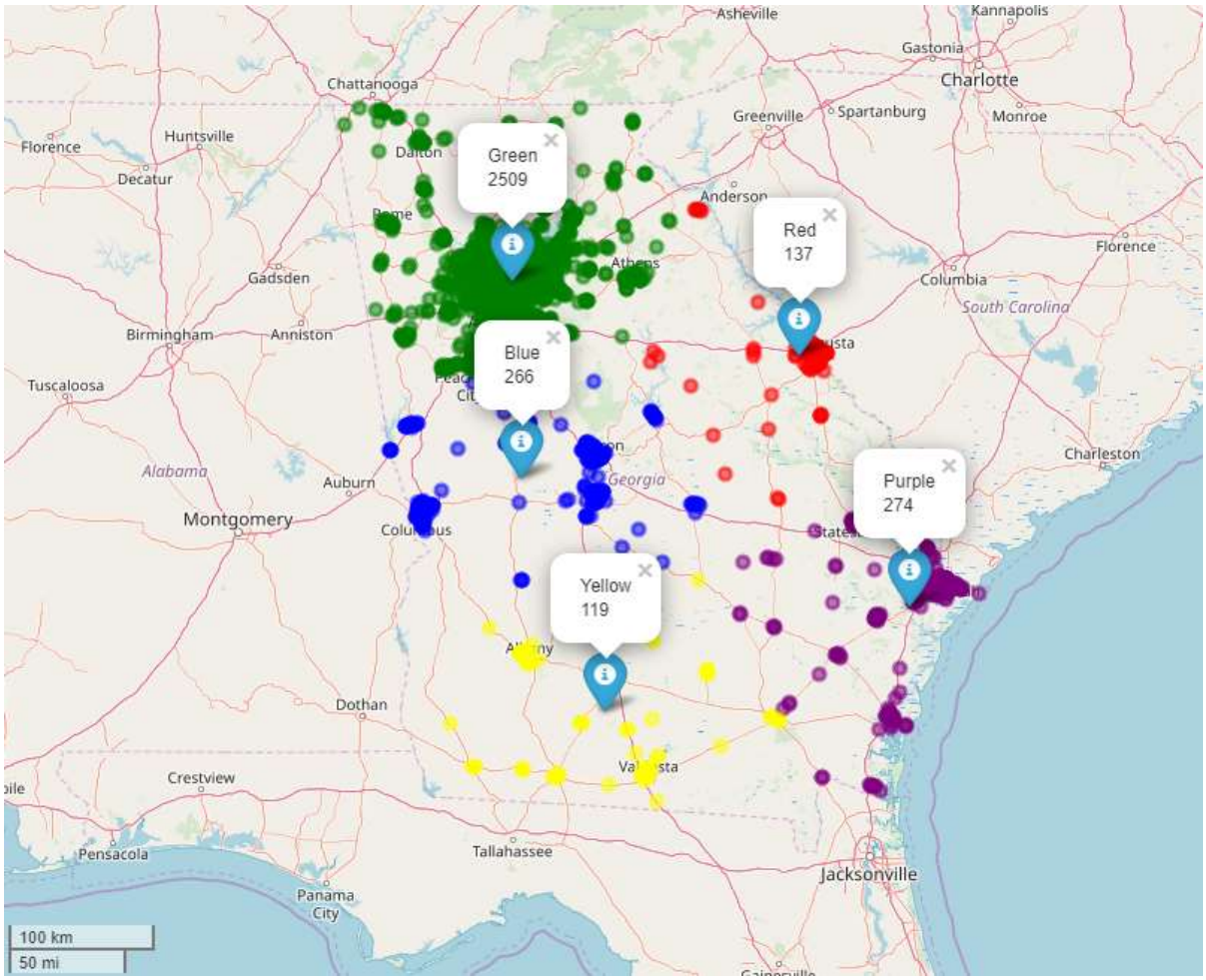

Figure 5: Territories chosen based on location only, displaying number of restaurants

While this grouping is clearly inadequate, does suggest a possible solution. Outside of the Atlanta area, the density of restaurants is low and the area to cover is quite large. The four territories outside Atlanta could be merged together to form two territories, freeing up two salespeople to move to the Atlanta area. The two new territories outside of the Atlanta area would cover a larger area and include all types of restaurants, thereby increasing the time the salespeople would spend travelling and learning about the restaurant types. However, the number of restaurants would still be lower than average helping to balance the load.

The Atlanta area has a high concentration of restaurants and it seems appropriate to create overlapping territories based on restaurant type. These salespeople would have a smaller physical territory to cover and fewer restaurant types to learn about, but more restaurants to cover.

I used this approach to create two territories outside of Atlanta and ran the k means grouping on just the restaurants in the Atlanta area.

Table 3: First attempt at mixed method solution

Territory:	yellow	purple	green	red	blue	Total
Asian Restaurant	386	0	0	52	39	477
Chinese Restaurant	0	0	859	177	180	1216
Indo-Pak Restaurant	250	0	0	14	12	276
Japanese Restaurant	0	559	0	128	111	798
Korean Restaurant	188	0	0	11	16	215
Thai Restaurant	160	0	0	22	20	202
Vietnamese Restaurant	107	0	0	7	7	121
Total	1091	559	859	411	385	3305

The Atlanta territories (Yellow, Purple, and Green) were better, but still uneven. The average for these three territories should be around 840 restaurants. The territory consisting only of Chinese restaurants (Green) is very close in this regard. That territory only contained one type of restaurant, which required the other two territories to have more restaurant types. I judged that the impact of moving another type of restaurant into this territory (and thereby increasing the number of restaurants by over 100 at the least) would cause the territories to be more unbalanced when all factors are considered.

The other two territories are unbalanced in both number of restaurants and number of restaurant types in each. I considered balancing the other two Atlanta territories (Yellow and Purple) by moving about 270 restaurants from the largest Atlanta territory to the smallest. There were two obvious ways to do this:

1. Move Indo-Pak restaurants, or
2. Move Vietnamese and Thai restaurants

To decide on which approach to use, I decided to see if there were significant differences in the physical locations of restaurants in these two territories by mapping the centers of mass for each restaurant type. (In the picture below, note that the “Asian” category’s pop-up is minimized.)

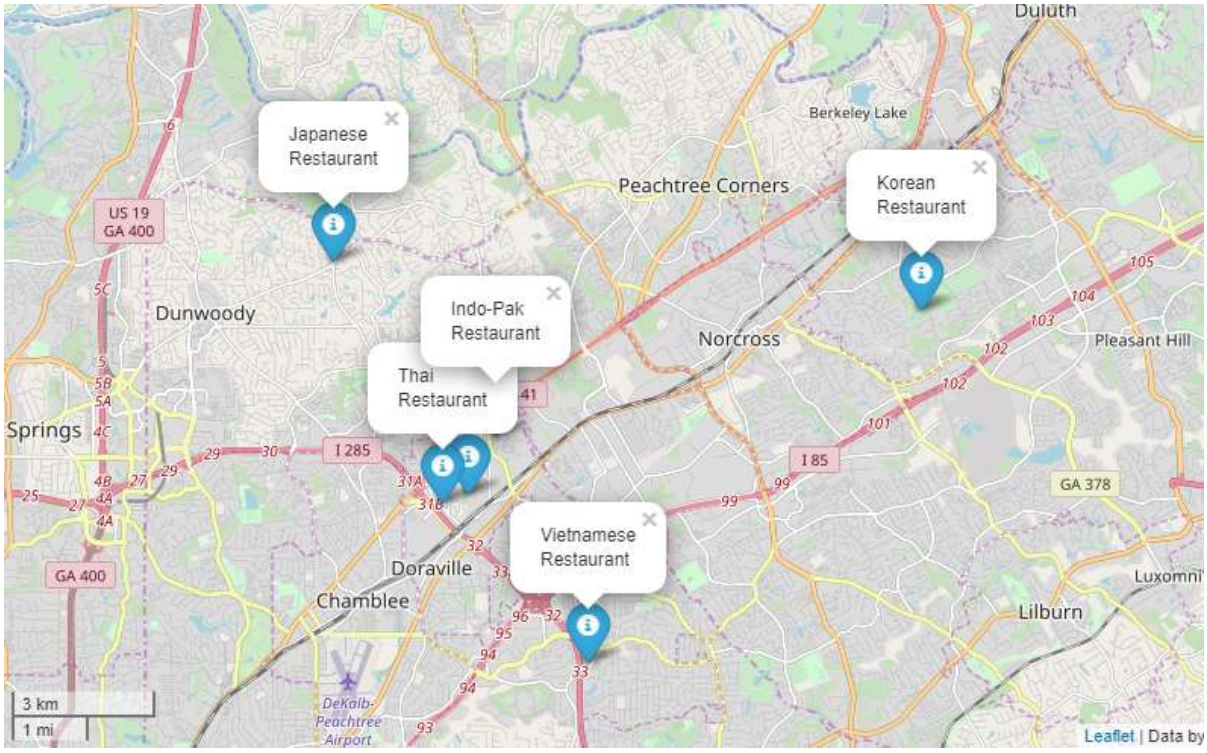

Figure 6: Mean locations of restaurant types in Atlanta area

The mean location of the Indo-Pak restaurants is closer to the Japanese center than are either the Thai or Vietnamese. By this measure, it would have made sense to move Indo-Pak into territory that only contained Japanese Restaurants. However, the difference was only 3-4 miles in a territory that is about 150 miles across. Balancing the number of restaurants in this way would leave the number of cuisines in each one uneven.

I decided instead to move the Vietnamese and Thai restaurants. This produced more even distributions of both the number of restaurants and the number of cuisines in each territory.

5 Results

Using both geographic and categorical data where appropriate in different areas of the state produced territories that are well balanced considering the overall characteristics of each (for example, fewer restaurants in the larger physical territories). As the company's circumstances change (for example, hiring new salespeople in the future or expanding operations to other states) this analysis can be quickly repeated with new data and constraints.

Table 4: Final sales territories

Territory:	yellow	purple	green	red	blue	Total
Asian Restaurant	386	0	0	52	39	477
Chinese Restaurant	0	0	859	177	180	1216
Indo-Pak Restaurant	250	0	0	14	12	276
Japanese Restaurant	0	559	0	128	111	798
Korean Restaurant	188	0	0	11	16	215
Thai Restaurant	0	160	0	22	20	202
Vietnamese Restaurant	0	107	0	7	7	121
Total	824	826	859	411	385	3305

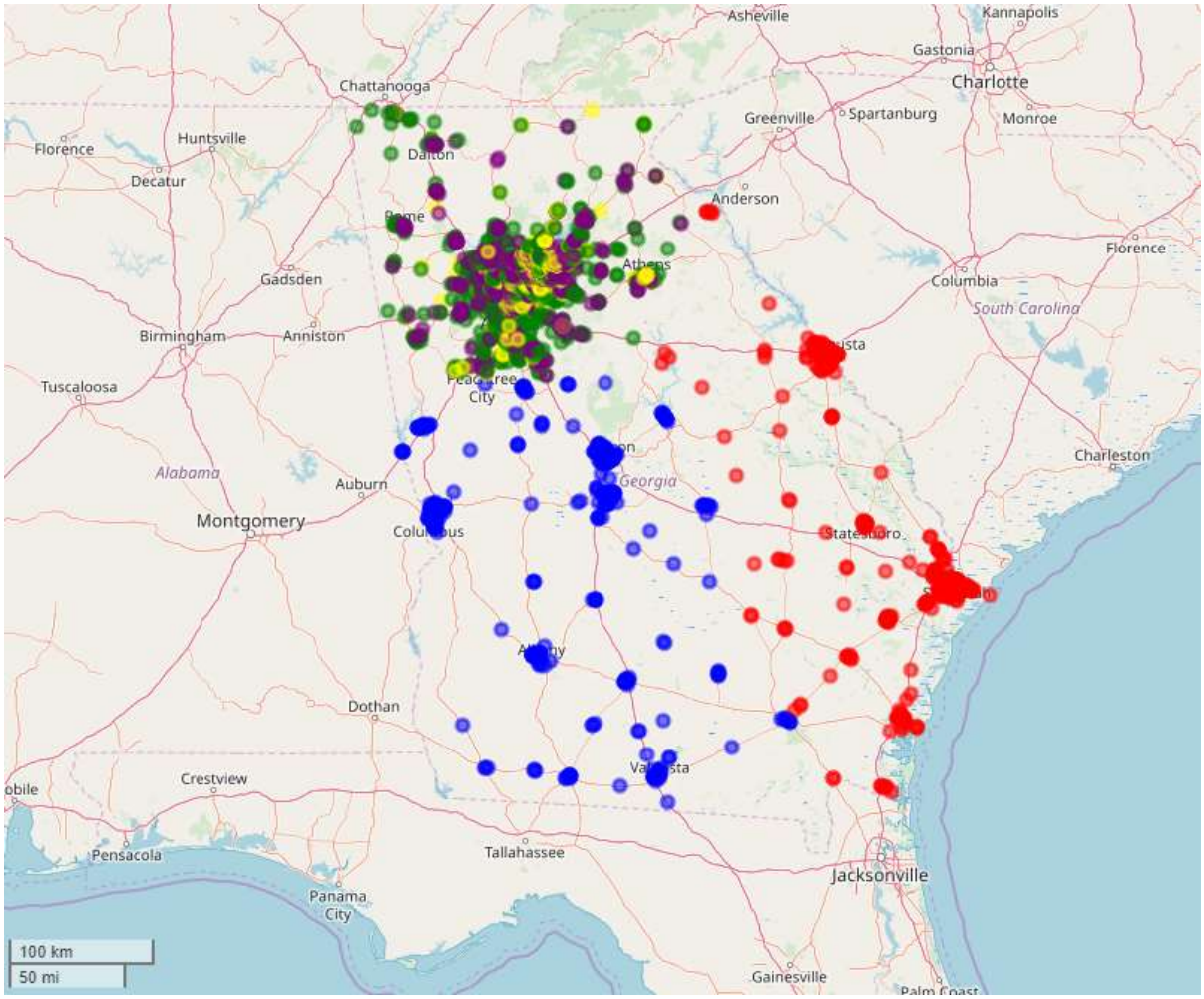

Figure 7: Final sales territories

6 Conclusions and Discussion

6.1 Methods

Extracting and cleaning data for this project was a challenge, because Georgia has both large areas with few Asian restaurants and areas of very high restaurant concentrations. The recursive method of requesting data (using bounding box requests with recursion) is efficient, quick, and provides a complete data set with no duplication.

The data set itself had a high percentage of mistakes, requiring extensive data cleaning. I used programmatic methods wherever possible to reduce labor and to make it possible to replicate the project as circumstances change. While data cleaning is not glamourous, it takes a large part of the time for most data science projects and I gained valuable experience in this area.

This project used machine learning techniques across two different types of data. This required balancing domain knowledge of sales methods and the restaurant industry with the capabilities and restraints of the machine learning technology used (in this case, k means clustering).

6.2 Critique and Future Directions

The final sales territories are clearly defined, which would make it easier for our (fictional) salespeople to identify and pursue opportunities. However, in the densely populated Atlanta metro area location data was used outside of the machine learning algorithm. This was appropriate given the number of clusters (salespeople), but enhancements to this project could include methods to mix location and cuisine data into the same learning algorithm, perhaps using a custom algorithm.

The algorithm chosen for this project (k means) tends to produce uneven groupings (particularly in cases where there are uneven concentrations of samples), and this algorithm could be modified or replaced to automatically produce clusters more appropriate for salespeople. Finally, domain knowledge was used to balance the number of cuisines in a territory with the number of restaurants. Future versions of this project could capture the relevant domain knowledge and perform this step automatically.

James Cage
July 2, 2019