Identifying Sales Territories for an Asian Restaurant Supplier in Georgia

Coursera Data Science Capstone Project

James Cage
June, 2009

Introduction / Business Problem

A (fictional) company that sells equipment and supplies to 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 devices and nonperishable supplies to all types. For example, it sells sushi display cases to Japanes restaurants and tandoor ovens to South Asian restaurants. The company wishes to define territories for its sales people 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 the Savannah area, while another salesperson would cover all other ethnicities (Japanese, Vietnamese, Korean, etc.) in the same area.

Audience

This analysis will be useful to the following:

- Business-to-business vendors: This analysis will give 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 will 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 will demonstrate how to **efficiently gather thousands of Foursquare venues** over a very wide area, while **preventing missed venues and duplications.**
- The problem requires clustering by geographic location and venue type (non-location) data. This project will show how to **scale factors prior to machine learning analysis to give usable results** in the real world.
- Foursquare frequently categorizes restaurants incorrectly. A Mediteranian restaurant may be placed in the Asian category, or a restaurant that should be labeled “Japanese” may be placed in the general Asian category. This project will demonstrate how to **clean data programmatically, using key-word searches** within venue categories.
- Finally, this project will give many examples of displaying data on maps, tables, and charts.

Data

I will use data from Foursquare to create a database of Asian restaurants. Foursquare does not include South Asian restaurants in its “Asian” category, so those will be added to the database using “Indian” and “Pakistani” category codes.

Obtaining Data

As noted in the Introduction, it will be necessary to retrieve 1000s of venues over a large area. The point-and-radius method of requesting Foursquare data will not be adequate, as it either introduces gaps between the circles, or creates duplicates due to overlaps. I will obtain the data using a bounding box method instead. I will write Python code to divide the geographic area into boxes that are small enough for Foursquare queries. Foursquare also limits the number of returned venues to 100 per request. When a bounding box request returns 100 results, my code will divide the box into four smaller boxes and submit a new request for each one. This will continue (using recursion) until all requests return 99 or fewer results.

The figure below shows the requests (represented by boxes of various sizes) and the number of results (represented by the color of the box) for my tests so far. 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.

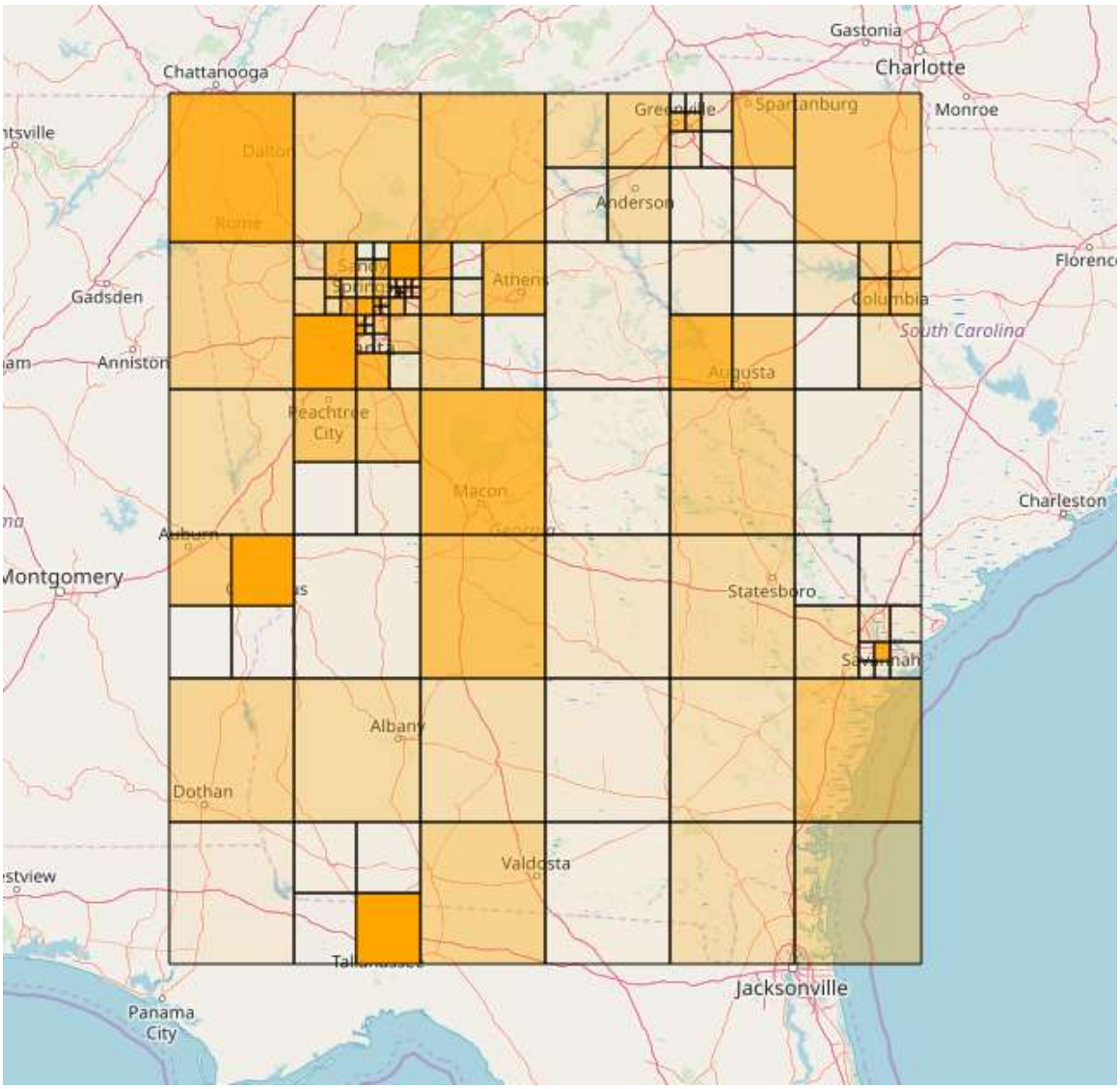

Figure 1: Bounding box Foursquare requests using recursion in Python

The next figure shows some of the smallest boxes in the picture above. These boxes are about 1.4 miles on a side, or roughly 2 square miles. Obtaining this data using equal-sized boxes (or equal-sized circles using the method demonstrated in our class) would require **over 25,000 requests.** My method uses roughly **150 requests** and **executes in less than 90 seconds**. It returns some venues in adjacent states which are dropped during data cleaning.

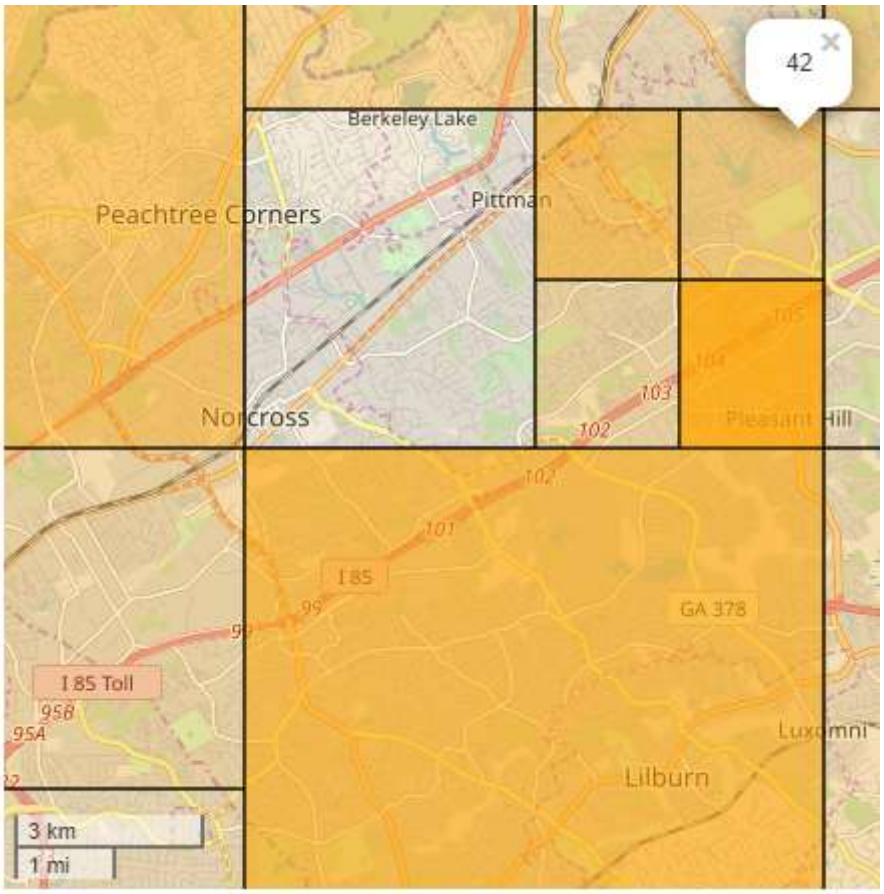

Figure 2: Detail showing size of smallest bounding boxes (scale at lower-left)

Cleaning & Scaling Data

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

Most Asian restaurants (with the exception of South Asian restaurants) in Foursquare are assigned to sub-categories under “Asian”. “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). My code will correctly assign restaurants where possible based on keywords in the restaurant’s name. Other restaurants offer a mix of types and will remain in “Asian”.

Finally, this project will demonstrate how to mix different types of data in a machine learning project. The analysis will use geographic (latitude and longitude) data, along

with the type of the restaurant, to segment the restaurants. One-hot encoding will turn the restaurant category into numeric data. How should the machine learning algorithm compare a unit of longitude with a 1-or-0 measurement of a restaurant's cuisine? I will address this by defining the trade-off using the miles between venues. If sales people would be willing to travel no more than 50 miles to visit a restaurant within their cuisine sets, then my code will scale the latitude and longitude so that a one-unit change will be equivalent to 50 miles.

Methodology

TBD

Results

TBD

Discussion

TBD

Conclusion

TBD