Predicting Zip Code Relevancy in Developing Areas in and Around Austin, Texas

By: Eric Lu

Table of Contents

1. Introduction

Austin, Texas has been ranked the number one "Best Places to Live" by U.S. News & World Report and is one of the fastest growing cities in the United States. A lot of the current studies focus on developed urban areas and neighborhoods of Austin, instead, I will be focused on Zip Code areas that are developing/up-and-coming. I will be trying to cluster the zip codes and their venues to see how the areas around Austin are developing.

This kind of prediction would be very useful for investors and potential new Austinites. It can give a bird's eye view of the developing areas around Austin which could be useful to decide in which parts of Austin would be a good real estate investment. This would also be useful for people moving to Austin trying to find the area that is best suited for them to live.

2. Data Acquisition and ETL

2.1 Data Source

- Open Source Zip Data: I used an open source zip code geography to properly map the zip codes to the folium map.
- Foursquare API: I also used Foursquare's location data to get venue data to do the clustering on. After getting the zip code data, I used the Foursquare API to find venues around the zip codes.
- Google: I also used Google's information on specific coordinates of towns (these town's coordinate will be used to filter through some zip codes)

2.2 Zip Code Data Transformation

The Open Source Zip Data was a critical component of this project as it mapped out the geography of each zip code's area. The open source data was in GeoJSON format, an open standard format for storing geographical information within a JSON file. Unfortunately, this data was split by state and each set of data had all the thousands of zip codes within each state. According to www.zip-codes.com, there are 2,598 zip codes within Texas so running the analysis on the vast quantity of "areas"/zip codes would be virtually impossible with Foursquare's API daily request limitations.

The first step in removing the zip codes that were unnecessary was loading it in QGIS, an open-source geographic information system (GIS). While this step could potentially be completed with an online GeoJSON editor like http://geojson.io/, the zip code GeoJSON file for Texas was too large to be loaded in and would likely result in the webpage crashing. Within QGIS, all the excess zip codes what are not around Austin needed to be removed; this step I just eyeballed it as there is no objective way to determine the Austin boundary as well as all the developing areas that should be considered "around" and "part of" Austin. I kept all the zip code until I started to hit San Antonio in the south, Fort Hood and Waco in the north, and College Station in the east.

After this step, I exported the reduced data into a Shapefile, another geospatial vector data format.

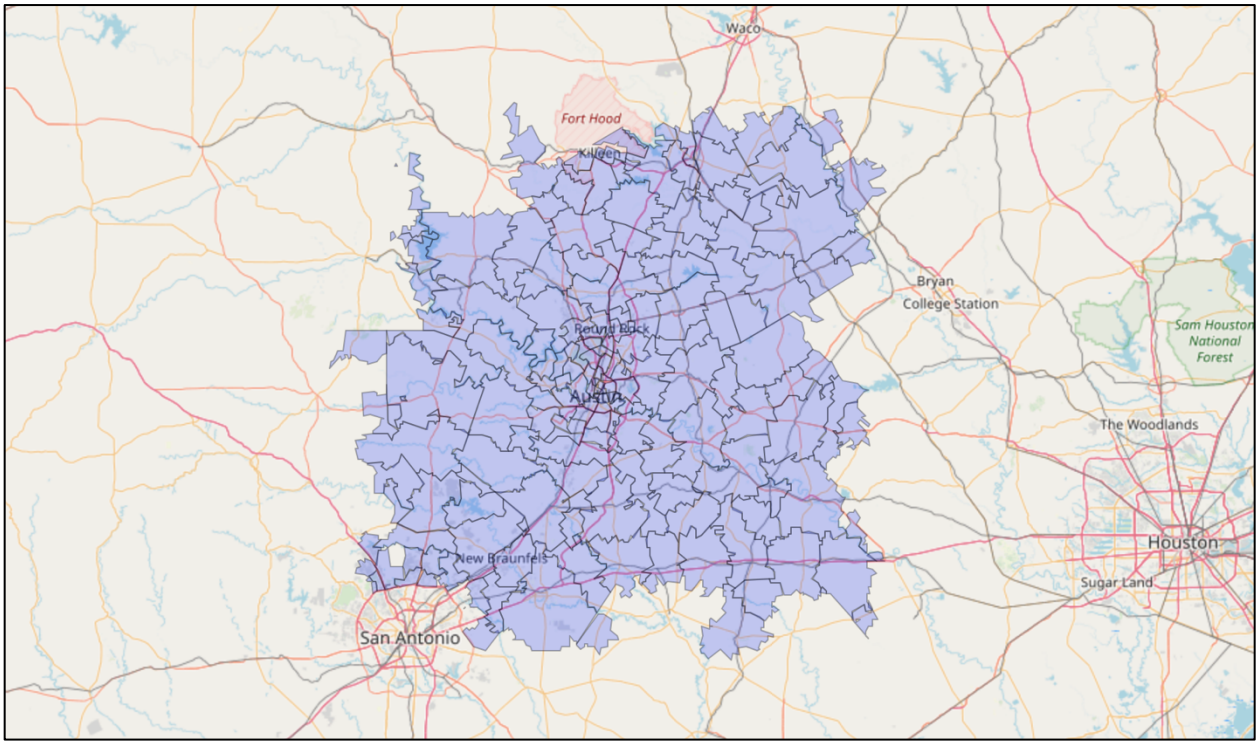

Figure 1. Reduced Zip Code GeoJSON data

The next step was to import the Shapefile into https://mapshaper.org/. This website allows many GIS data file conversions as well as the ability to simplify the geospatial vector data. I simplified the data down to about 5 percent of original data size, removing a lot extraneous vector data that while could look more detailed when mapped out, was ultimately unnecessary for my use case. After the simplification process, I exported the data back into a GeoJSON file and a second CSV (the CSV file is just a file for easy access to the critical information of each zip code and is not necessary).

Finally, the JSON file can be imported to https://ogre.adc4gis.com/ and convert it to the GeoJSON format and then prettify it via https://jsonformatter.org/json-pretty-print (this is also a great site to view JSON files in multiple viewing formats like form and tree view).

2.3 Zip Code Data Reduction: Finding Relevant Zip Codes

After formatting the zip code data into a manageable GeoJSON file, the file can now be load into python. Now I need to determine which zip codes are relevant to the analysis as I did not want zip codes that extended all the way to San Antonio but still wanted data/areas as far as possible around Austin to have a meaningful analysis.

To determine which areas are considered “relevant” to Austin and considered developing areas around Austin, I decided to call the Foursquare API to search for the amount of venues around a particular zip code’s center, this way I can figure out the general gradient of the Austin area (in

terms of venues), see when it starts to encroach on nearby towns, and decide when I can consider it non-Austin. Since Foursquare's API uses a parameter "search radius" in meters, I decided a 2000 meter search radius (about 1.25 miles) around the center coordinates of each zip code. Keep in mind Foursquare sets a return limitation for each request at 100 results so I did not want to make the radius too big where every zip code returned the maximum 100 results. A radius of 2000 meters should be enough to portray the density of an area while allowing densities of urban downtown areas to be differentiated from suburban areas.

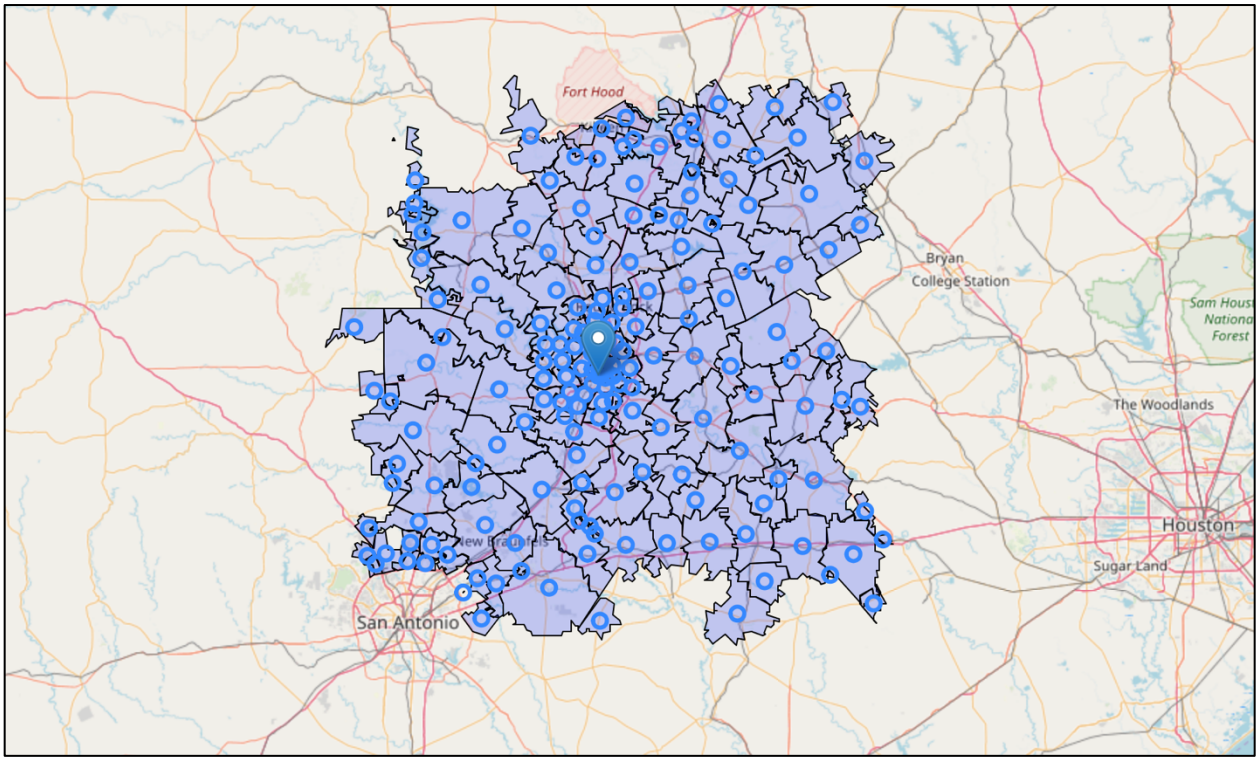

Figure 2. Search radius of 2000 meters

After gathering all the search results for each zip code, I mapped out the count of venues of each zip code in a folium Choropleth map. This map displayed the general gradient of the Austin area and allowed me to select the area I would be analyzing (specifically the area within San Marcos, Lockhart, Bastrop, Taylor, Georgetown, Leander, Lago Vista, and Dripping Springs).

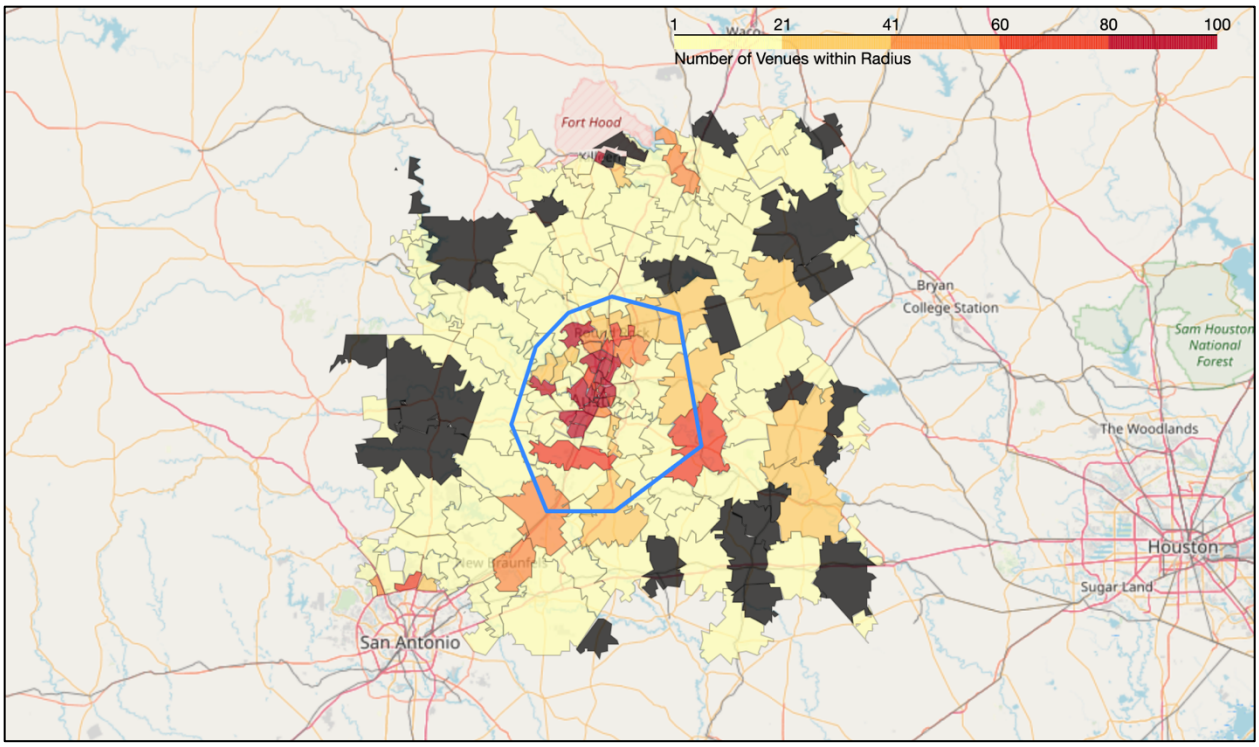

Figure 3. folium Choropleth map of venue counts of each zip code

With this area of zip codes selected, I went through the 2.2 Zip Code Data Transformation again to remove all the irrelevant zip codes.

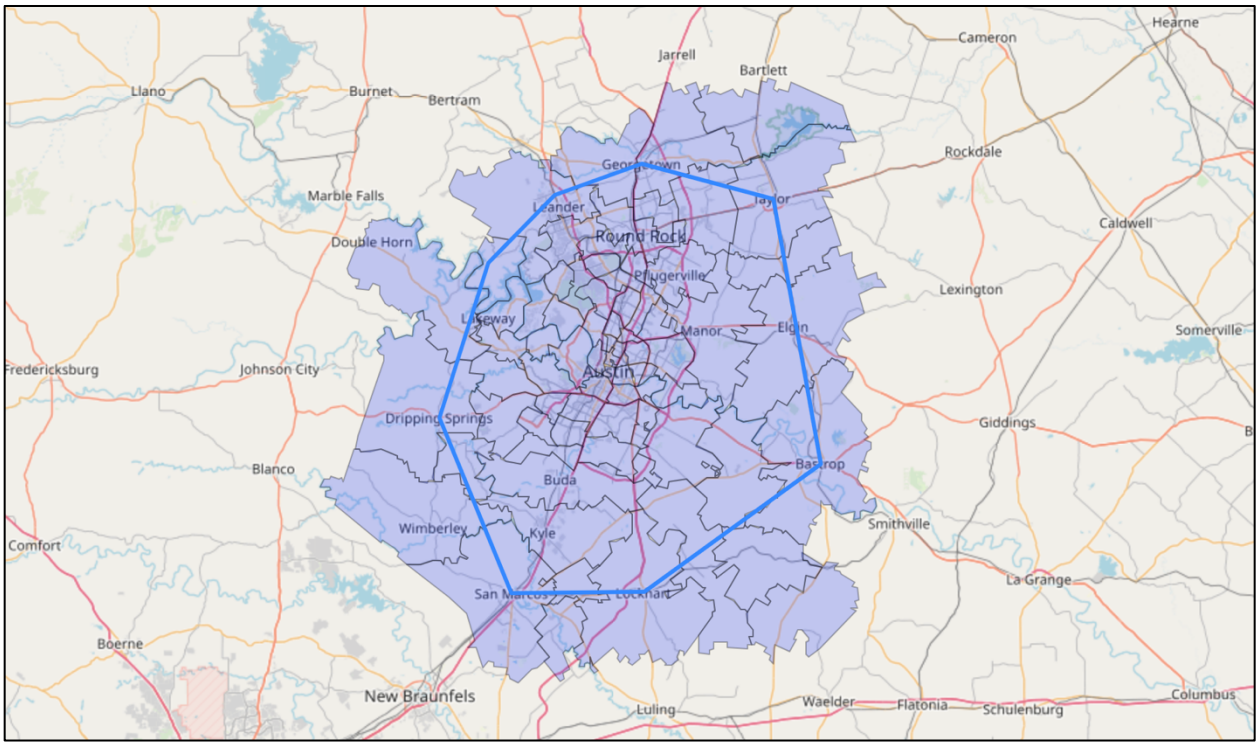

Figure 4. Selected "relevant" zip codes

The next step was to remove all the already developed zip codes (like downtown Austin and developed suburban areas around Austin) as I did not want developed areas to interfere with the rest of the data. To determine which zip codes were considered "developed" I created a boxplot to view the spread of the zip codes' counts.

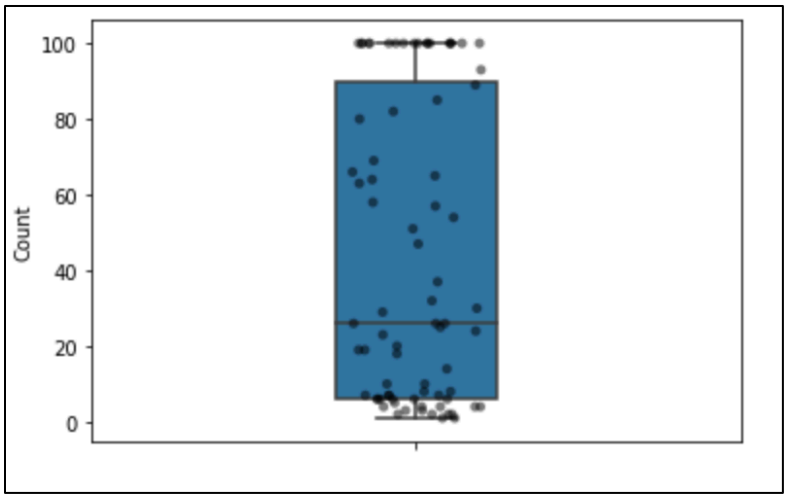

Figure 5. Boxplot of zip codes and their counts

Instead of removing the top quartile, I decided on removing any zip code with a venue count above 80 as it removed about the top 30% of zip codes.

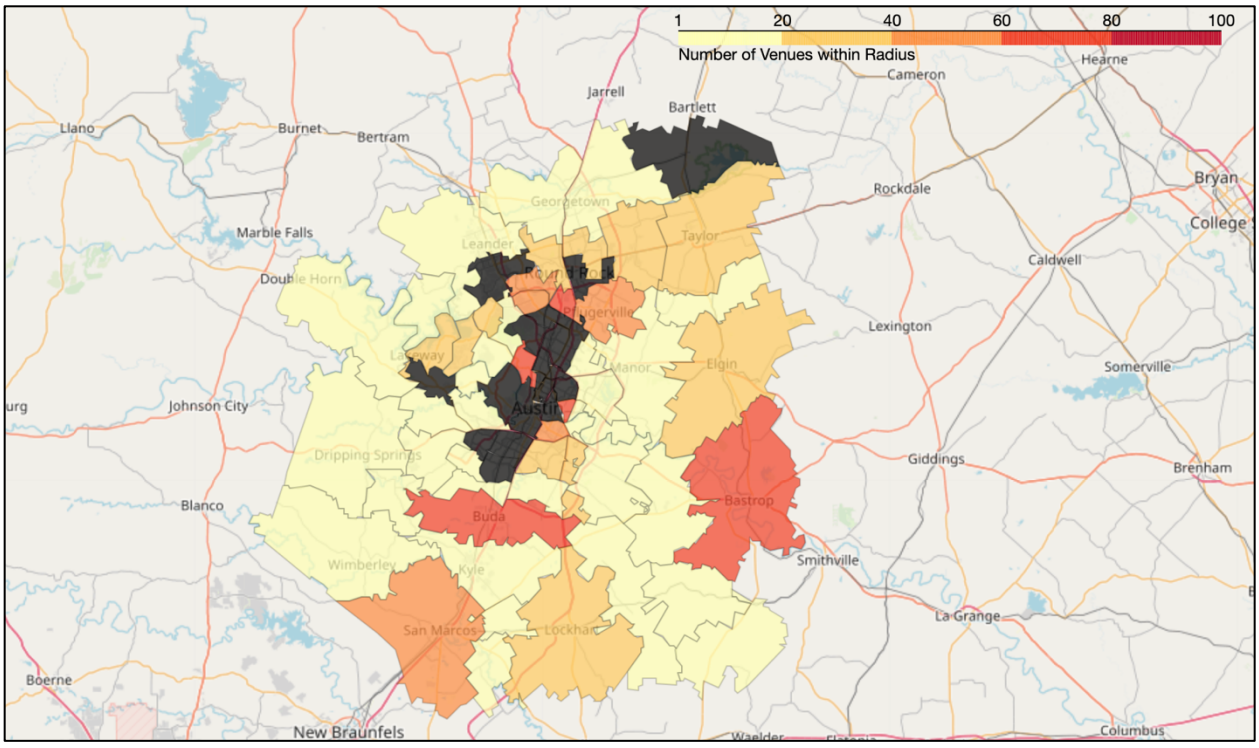

Figure 6. Developed zip codes removed (the topmost blacked-out zip code returned no results so it was not removed)

2.4 Gathering Venue Data from Foursquare

After all the zip codes have been selected, I had to gather more data than what the 2000 meter radius requests had returned. I wanted more data but could not just increase the search radius because I was bottlenecked by the Foursquare return limitation of 100 results. Instead, I decided to "bypass" this limitation by doing multiple searches around each zip code.

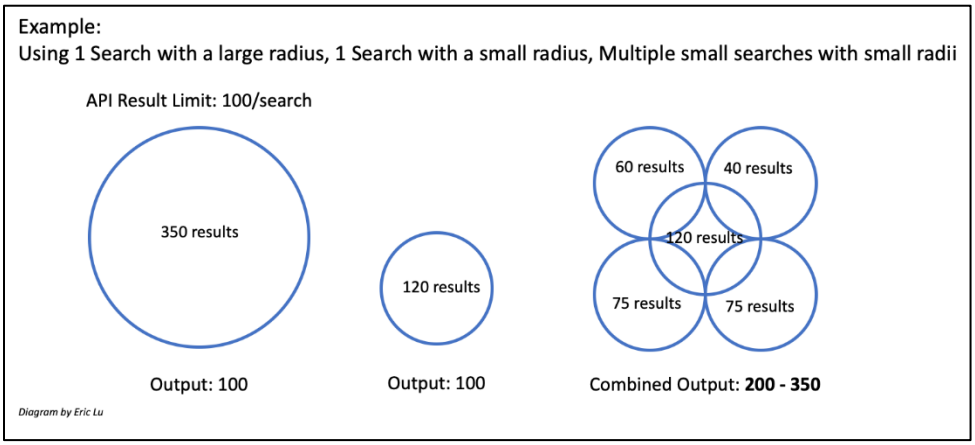

Figure 7. Bypassing search limitation via multiple searches

Given the center coordinate of each zip code, I now needed to determine how to find each of the search coordinates for each zip code. Since Earth's coordinates are not on a flat surface, I had to do the coordinate transformations on the surface area of a sphere.

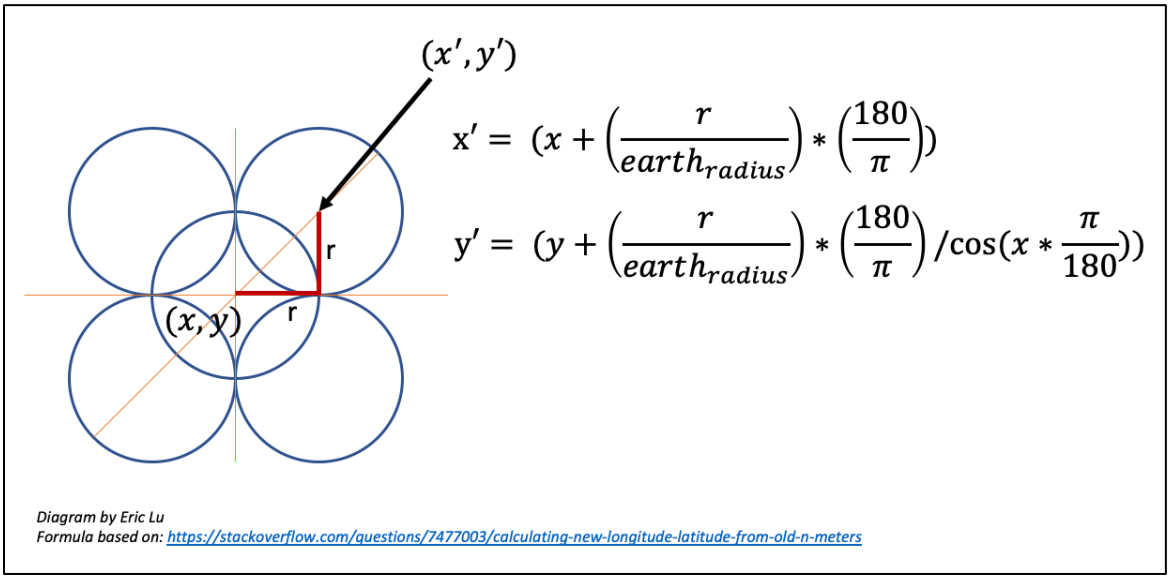

Figure 8. Calculation used to find Earth's coordinate transformations

I also wanted to ensure I was covering enough ground and search a big enough area, so I also had to calculate the total/effective area that was being searched if I was going to search with 9 overlapping searches.

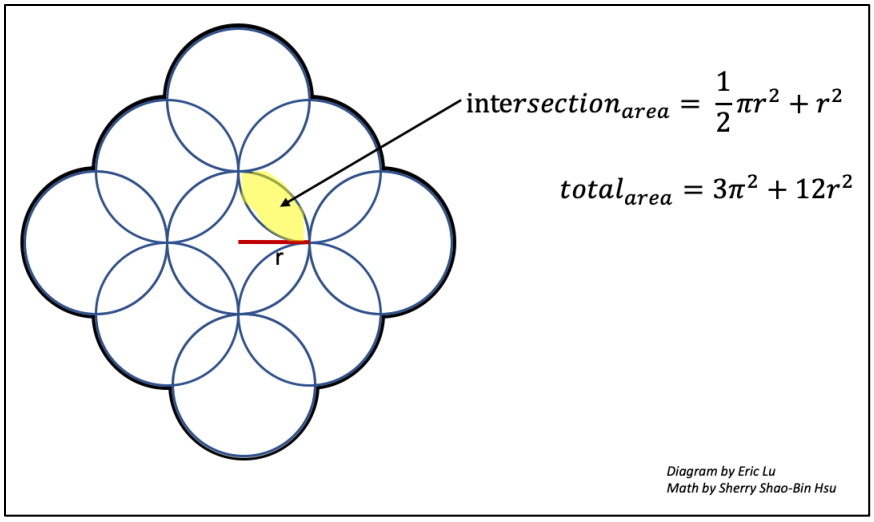

Figure 9. Calculation used to find the effective search area

With my selected search radius, I was now searching an effective area of 41 square kilometers or 16 square miles. Now each zip code would send 9 search requests to Foursquare at a radius of 1000 meters each (with some overlapping results that were easily removed).

3. Methodology

3.1 One Hot Encoding Data

After acquiring the new venue data, I wanted to one hot encode the data so that each category could be processed numerically. With this data, I could now group the data by zip code, find the top ten venue categories for each zip code, and then run K-Means on this new dataset of most common venues for each zip code.

3.2 Simplifying Categories via Foursquare Category Hierarchy

Another issue to consider was that if I just used the top ten venue categories for each zip code, there would be a lot of data that was going to be wasted and not analyzed. With 291 unique categories, there was insufficient data to train with such a high dimensionality. Instead, I went back to Foursquare to find a category hierarchy to try to simplify the categories. Now instead of detailed category names, there were 10 overarching major categories that I could simplify all the categories with.

Recursive Search Through Nested JSON Hierarchy

One issue was that the category hierarchy was a nested JSON with some categories having sub-categories (and some of those sub-categories having more sub-category layers). To find all the categories within each overarching major categories, I had to use a recursive search to extract all the minor categories within each major category.

3.3 K-Means Clustering

I chose to do K-Means Clustering as I thought it would provide good insight on the magnitude of each zip code area and would show the varying degrees in which zip codes were being developed around Austin.

3.4 Means Shift Clustering

Another issue I faced to clustering at a dimensionality of 10. While this is not considered "high", it was high enough to take into account the problems that came with clustering high-dimensional data. I decided to do Means Shift clustering to see if there would be improvements to the results while using 10 features.

4. Results

I trained the dataset 4 times, twice with K-Means and twice with Mean Shift. Each time I made adjustments to the data set itself. In the beginning, I used One Hot Encoded data with the top ten venues of each category as the dataset. Later, I used percentage counts of each category for each zip code. Finally, I ran it over the counts of each category for each zip code.

4.1 K-Means Cluster 1

The first cluster was with the One Hot Encoded data with the top ten venues of each category as the dataset.

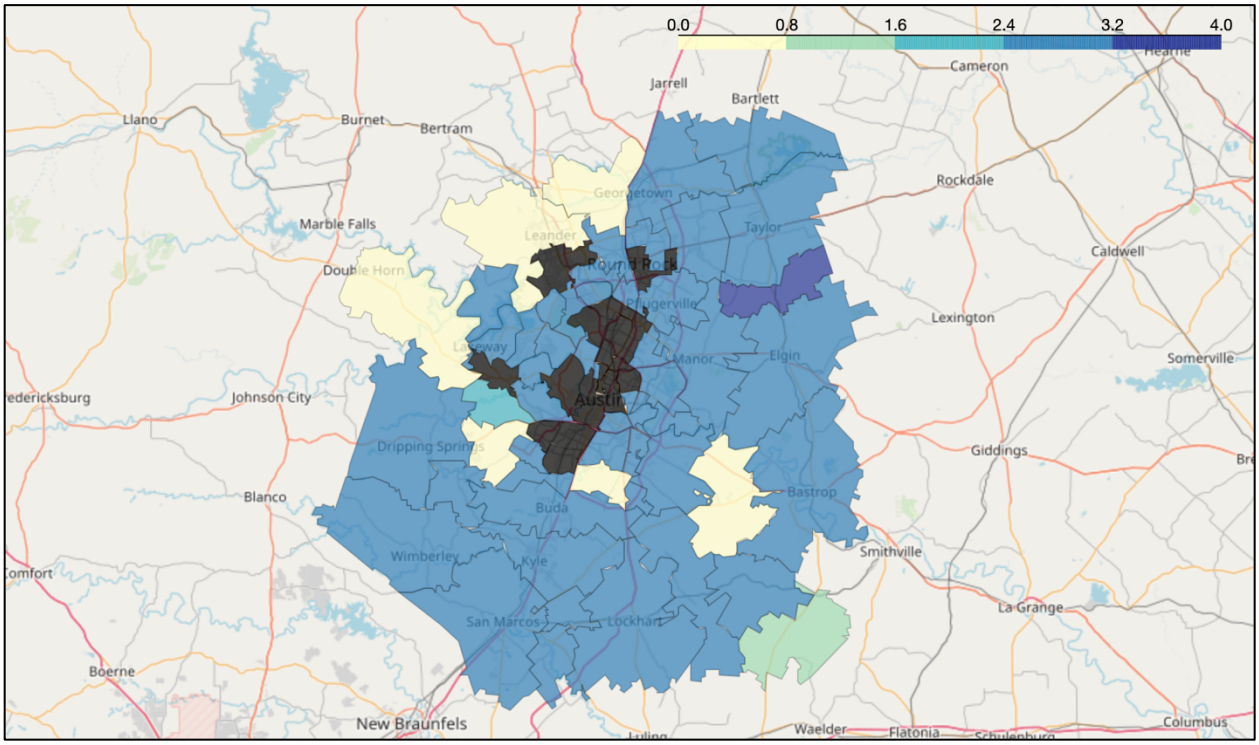

Figure 10. K-Means Cluster 1 Map

This clustering did not really provide good enough insight as a there was basically one cluster that held the majority of zip codes.

4.2 Mean Shift Cluster 1

This cluster (and every cluster after this) took into account the vast amount of categories and simplified it down to 10 major categories. I thought there were uneven clusters because of the high dimensionality so I decided on another clustering algorithm, Mean Shift. I used the percentage counts of each category. This meant that if a zip code had 1 out of the 5 colleges within these zip codes, the percentage count would be 0.2 while a zip code that did not have a college would have a percentage count of 0.

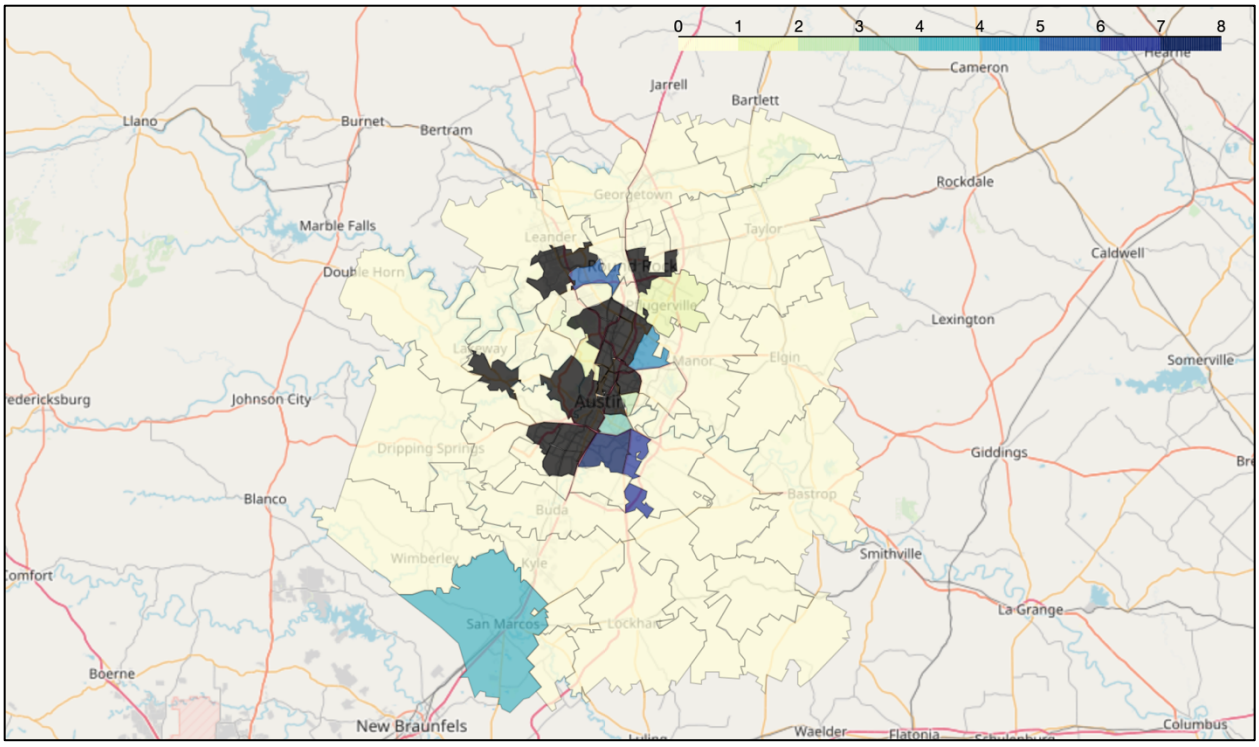

Figure 11. Mean Shift Cluster 1 Map

This produced 9 clusters. While this gave a slightly better result as the data correlated more with the venue counts of each zip code, there was still a high even cluster where most of the zip codes clustered into.

4.3 Mean Shift Cluster 2

While the percentage count I used for Mean Shift Cluster 1 took into account the different ranges of venue counts. I did not think it took into account the magnitude of venue counts enough as I did not want a zip code with 5 venues to be grouped with a zip code with 200 venues solely based on the types of venues that were within each zip code. Instead, this second cluster I used the counts for each category. This meant that if a zip code had 1 out of the 5 colleges within

these zip codes, the count would just be 1 while a zip code that did not have a college would have a count of 0.

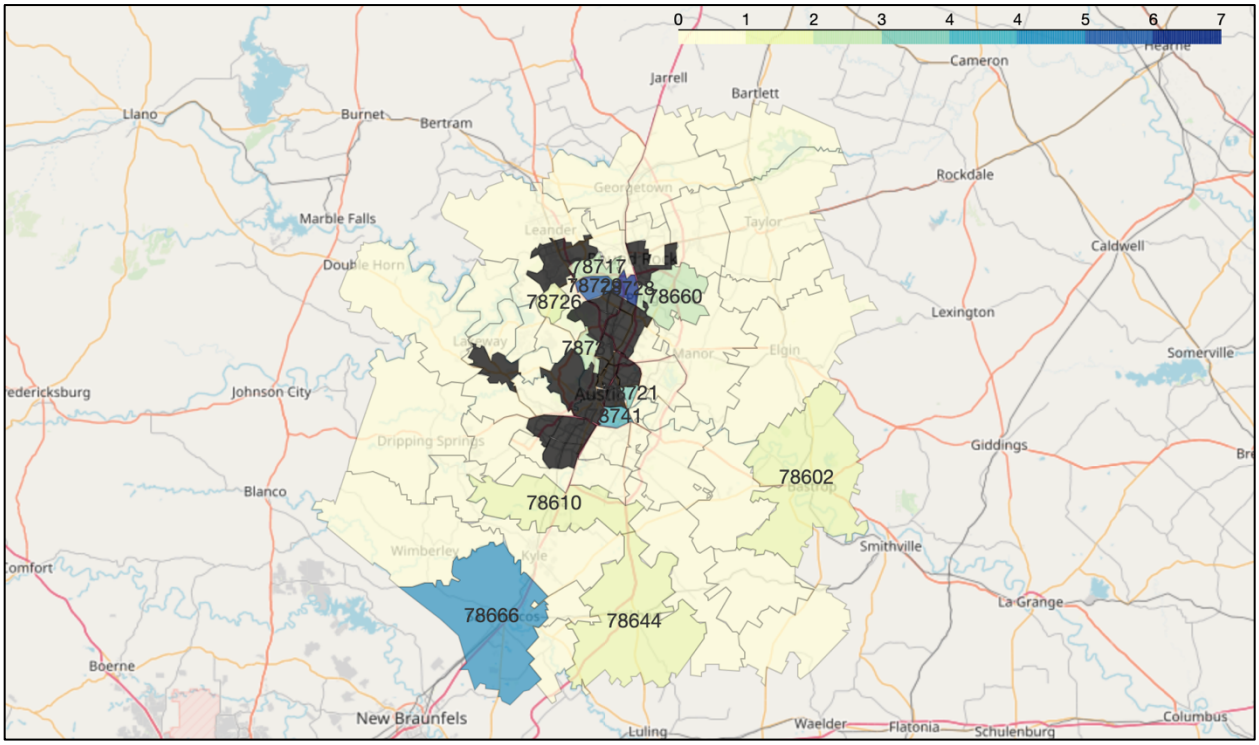

Figure 12. Mean Shift Cluster 2 Map

This produced 8 clusters. While 8 clusters is a lot for this data, this clustering explains each zip code area and its venues relatively well.

4.4 K-Means Cluster 2

I went back to try K-Means again with the counts of the major categories as perhaps a dimensionality of 10 was not large enough for Mean Shift to produce a big enough difference in results. K-Means also allowed me to define the number of clusters which I set to 5.

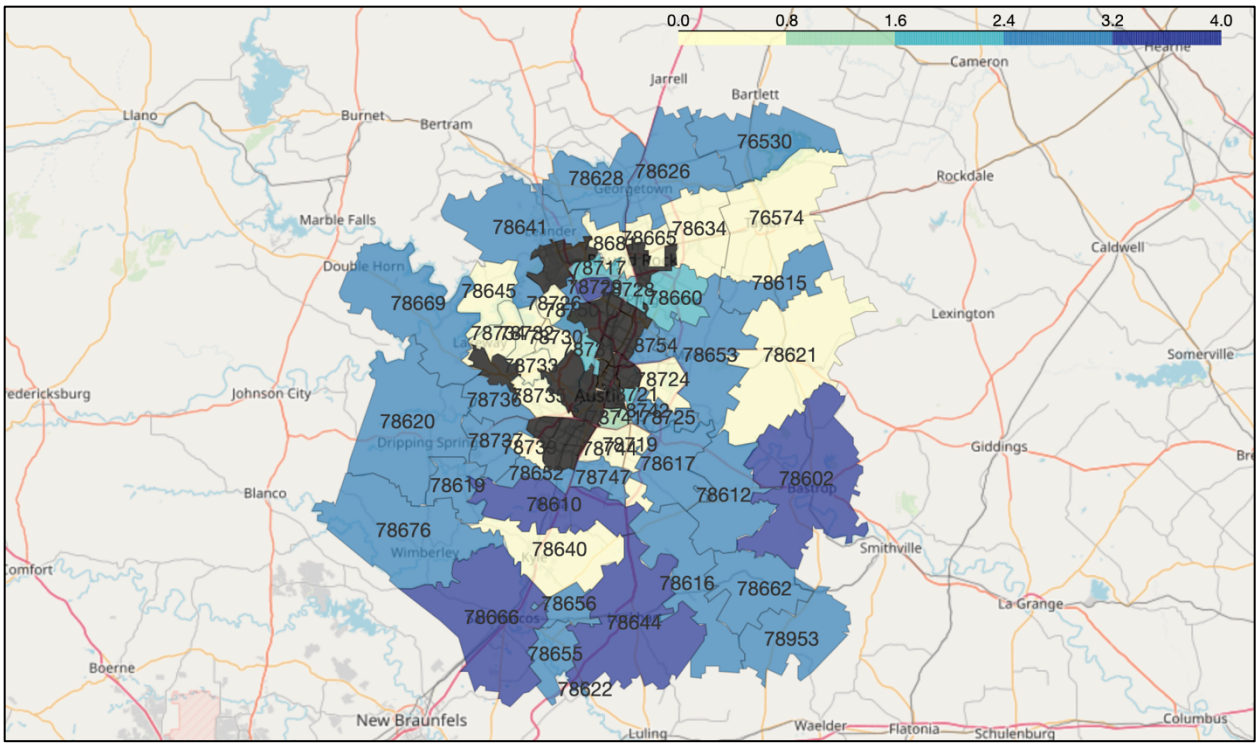

Figure 13. K-Means Cluster 2 Map

This produced the evenest clustering map that showed clusters with little development up to clusters with a lot of development.

5. Discussion

Based on the results of K-Means Cluster 2, there are 4 main types of development:

Type 1: Cluster 3 seems to have most of the zip codes that are the least developed.
Type 2: Cluster 0 seems to have some of the zip codes that are developing.
Type 3: Cluster 2 has zip codes around "developed" North and Northwest Austin.
Type 4: Cluster 4 has the 4 small towns (San Marcos, Bastrop, Lockhart, Buda) and a Northwest Austin zip code

The 5th cluster is interesting because it is just 1 zip code, 78741 (South East of Austin's downtown). Even as I change the number of clusters down (all the way to 3), zip code 78741 is always in its own cluster. This might be because 78741 has disproportionately high amounts of food and shopping areas as well as a good amount of entertainment and transportation venues as well.

Types and their Zip Codes	
Type	**Zip Codes**
Type 1	78669, 78662, 78676, 78747, 78653, 78615, 78617, 78953, 78620, 78612, 78656, 78655, 76530, 78725, 78641, 78754, 78622, 78626, 78616, 78652, 78750, 78736, 78737, 78742, 78619, 78628
Type 2	78733, 78640, 78732, 78645, 78744, 78734, 78730, 76574, 78665, 78726, 78634, 78724, 78621, 78681, 78739, 78719, 78735
Type 3	78717, 78731, 78728, 78660, 78721
Type 4	78602, 78610, 78666, 78644, 78729

Figure 14. Types and their Zip Codes Table

Zip codes in Type 1 are the least developed areas around Austin with very low venue counts across the board, these areas are the least likely to develop a lot in the next few years.

Zip codes in Type 2 are the areas that have little development and sparse venues. These are potential areas that are good to purchase more developed rural areas.

Zip codes in Type 3 are the areas close to Austin that are areas of development. There are probably more suburban residential growths throughout these areas.

Zip codes in Type 4 are mostly the small towns around Austin with condensed development around its area.

6. Conclusion

6.1 Retrospective

A lot of time was spent learning about the GeoJSON datatype and learning how to extract the data from the open source zip code data set. This really shows the 80/20 dilemma of data science where 80 percent of the time is spent on data preparation.

This analysis only uses limited venue data from Foursquare. With the limited data, it was hard to train a lot of models. I only had time to run the data with 2 algorithms. If I could have created more models with the data, there may have had different outcomes. All the algorithms also run on a venue count based dataset. If there were more sources of data and more features to analyze the results may have differed.

6.2 What I learned

This project was interesting as it pushed me to go through the entire project lifecycle in a reasonable time frame. Selecting the problem took a lot of planning as the problem itself had to be doable in within the time frame. This required a lot of research and prototyping before I could even come up with a problem.

For example, I started off thinking I could do the analysis on neighborhoods, a smaller unit than a zip code. Through extensive research, I learned that there was not a lot of data around neighborhoods publicly available. Even the Zillow public dataset had a lot of areas that were not

classified to distinctive neighborhoods. Since neighborhoods were quite subjective, most of the neighborhood boundary datasets were sold by various companies for commercial purposes.

After contending to using zip codes instead of neighborhoods, I had to understand the zip code data set I was using. While the original data set was a GeoJSON, the file was way too big to import into any online editor. I had to download an open source GIS editor to remove unneeded data. I had to simplify the geo-vectors down even more as the file was still too big for most online tools. I had to learn about the GeoJSON and Shapefiles format so that I could do proper modifications as well as read and understand the data I was using. I also had to learn about the folium, the python leaflet.js mapping library I was using to create the graphs as this was critical to actually visualizing the data. While I have learned about the theory behind K-Means, I wanted to choose various algorithms to create models with. I ended up learning about and trying the Mean Shift algorithm.