# FINDING YOUR SECOND HOME

**Piyush Gupta**

March 3, 2018

**Problem Statement:** You run a company XYZ which operates in the Real State business, your company *(via its app/web application)* provides property recommendations to the customers (for buying/rental) who are settling abroad for long/short term. Based on the customer profile/preferences, you recommend them the best suited neighborhood(s) in their choice (if any) of city.
In simple terms, customer from a **[Location]** $_{A}$ wants to go to

**[Location]** $_{B}$, where B is supposed to be his/her the best possible option(s) out there.

Here by best possible options we mean the target location (B) which closely matches location A in terms of the amenities present around it.

Now there are two scenarios:

1. The customer is *extremely well adjusted* to kind of surrounding she had at her old place **A** and would like to select a neighborhood **B** in the new city which closely matches **A** in terms of the great amenities and the venues in her neighborhood **A**
2. The customer is *not that satisfied* with her current neighborhood **A** and would like to select the new location **B** which matches some or all of her preferences.

   **Note 1.** *Both the above scenarios are essentially the same, as we would have a certain set of preferences in any case which we would like to match to the new location B, the only difference is that in the first case the preference list can be generated using the location data but in the second case the preferences would have to be fed manually into the system. (selection of options from a list of venue items)*
   **Note 2.** *For sake of simplicity in this project example we would take scenario 1 where customer is happy with her present location.*

**SAMPLE CASE:** A person P who lives in Toronto City <Address: > wants to move to New York City, and wants the best matching neighborhood as per his Toronto city address. He also wants the new place to be suitable as per his financial position. *(the affordability factor)*

## DATA ACQUISTION:

we will need the location profile of an address, which will be the mean frequency list of the venues around it.

For example, top 6 most common venues around a location might look like

| location | Clothing Store | Spa | Cosmetic Shop | Restaurant | Brewery | Cafe |
|---|---|---|---|---|---|---|
| Origin_location | 0.074 | 0.053 | 0.051 | 0.042 | 0.022 | 0.018 |

*Terminologies used:*

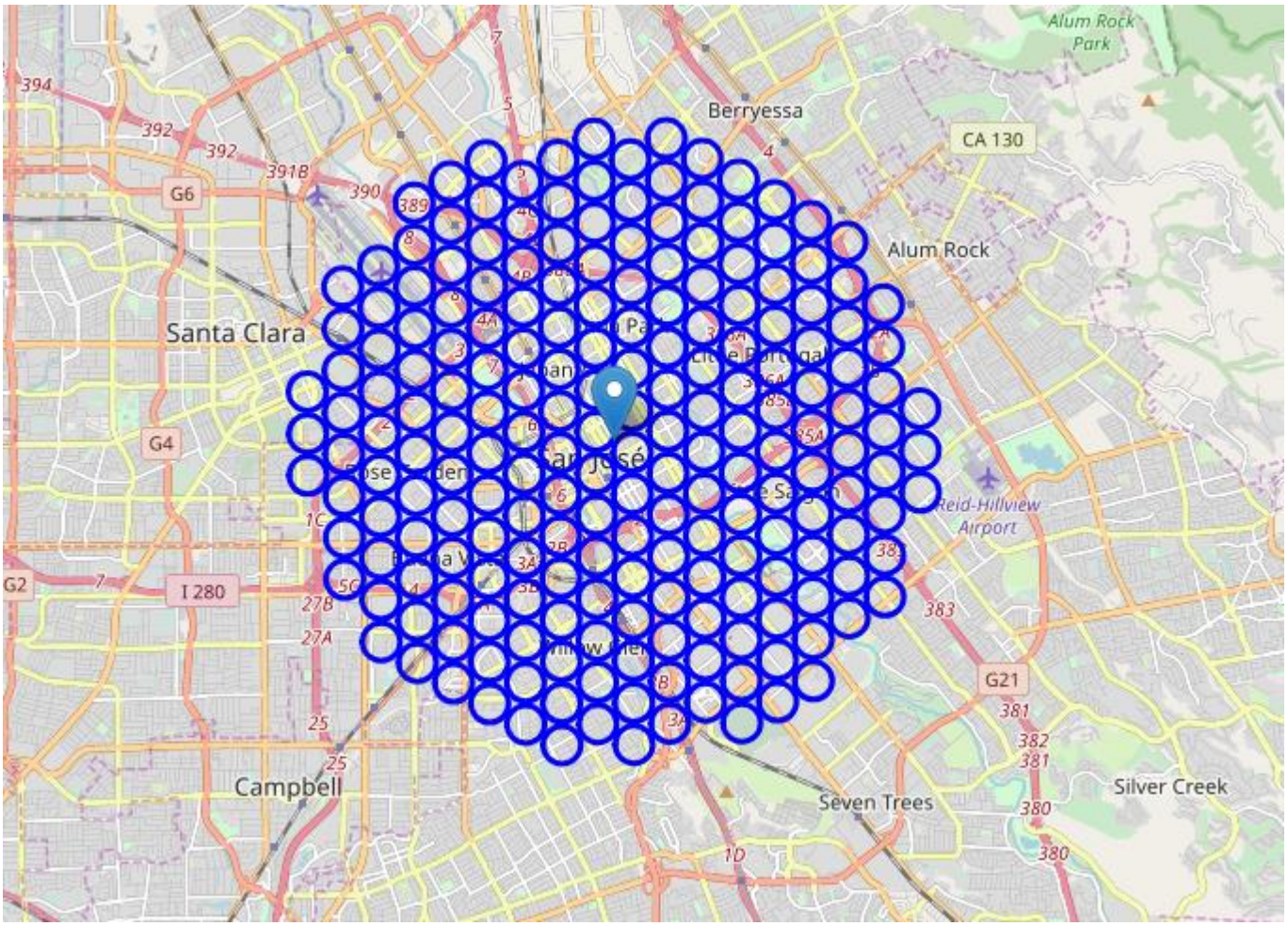
**Target location:** where the user wants to go
**Origin location:** where the user is coming from
**Location profile:** feature vector containing venue categories and corresponding mean_freq count.

**Location Generation:** Candidate coordinates around the target locations are generated algorithmically in a particular radius centered around the target location using **pyproj library**, then we will use **Google maps API** to generate addresses for those coordinates. This will be saved as the location dataframe of the candidate addresses.

**Profile Building:** profile building of a particular location will be done using **Foursquare API**, using *venues/search* and *venues/explore* call.

## Methodology/Analysis:

- First, we will obtain the candidate location data (*address/lat/long/distance from target center)* using the pyproj library. These locations will be equally spaced and will be distributed in a region of radius of **6KM** from the center location.

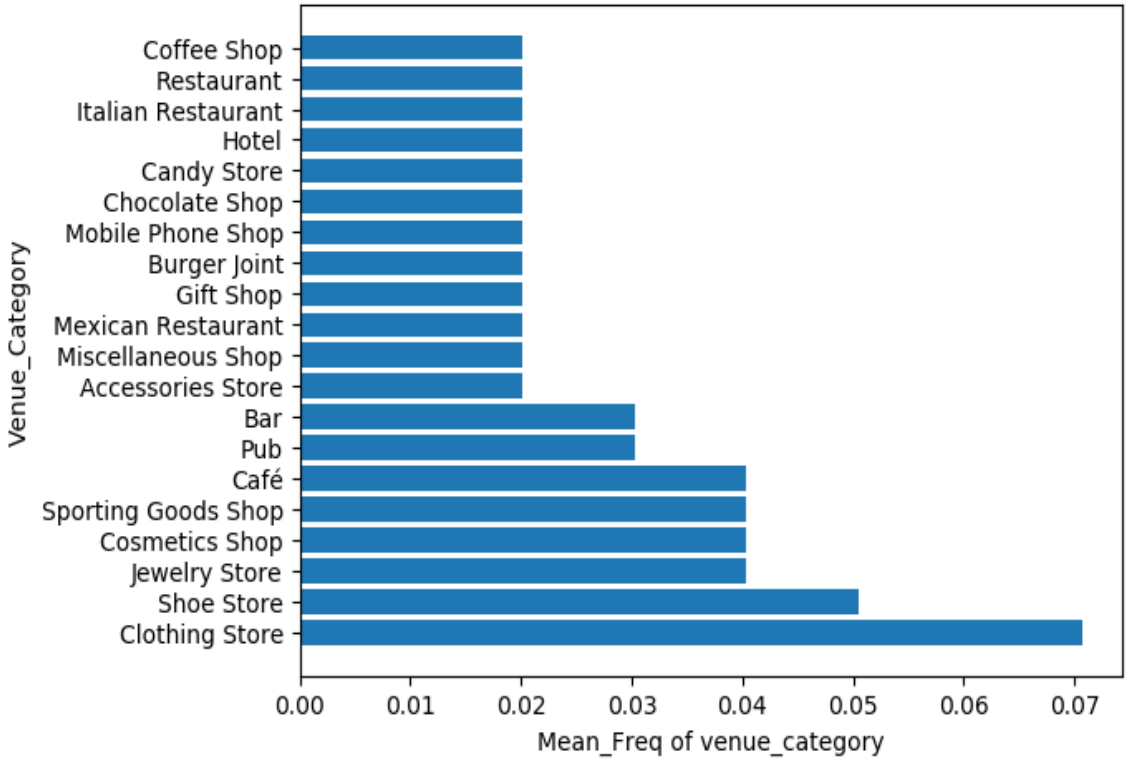



**Here 206 candidate locations were generated centered around the target location of San Jose, CA**

## Feature selection:

- **Getting the origin location profile:** then we will make the location profile of the origin address **(A)** using foursquare API, here we will make the
  GET https://api.foursquare.com/v2/venues/search and
  GET https://api.foursquare.com/v2/venues/explore call to build the initial location profile.

**Raw_Venue_Profile** ⊕ **Rec_Venue_Profile** 〓 **Refined_Venue_Profile**



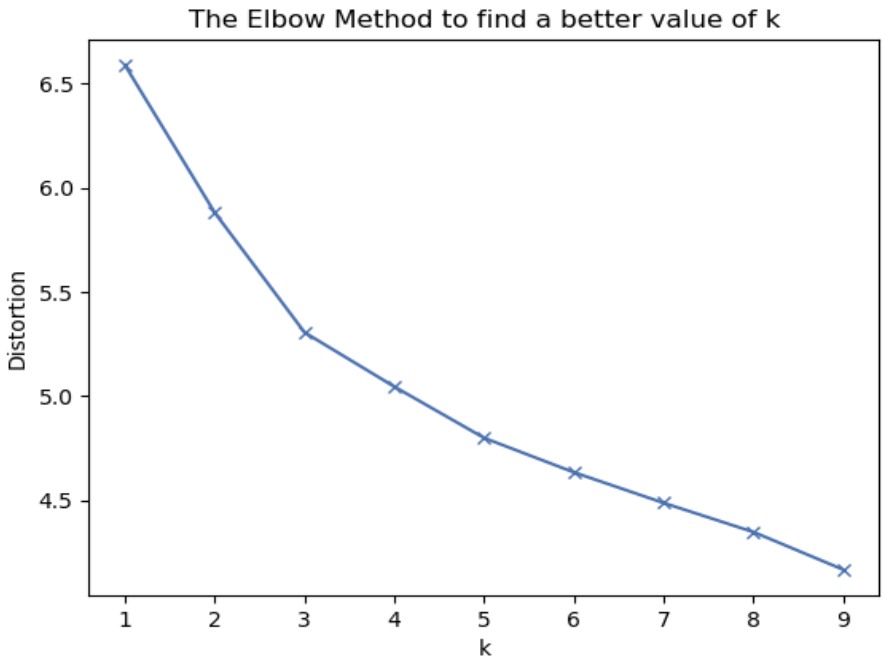

- **Addition of the preference list:** Then we will add the preference list provided by the user to the already built refined venue profile, this will complete our origin location profile building. We will only add those items from the *pref_list* not already present in the *refined_origin_profile* and add median_freq as the value for that category.
  A sample preference list (not in any order) might look like pref_list=
  *['Gym / Fitness Center', 'Hospital', 'Shopping Mall', Italian Restaurant', 'Spa', 'Park', 'Playground']*

- Then we will retrieve the location profile of all the candidate addresses generated previously.

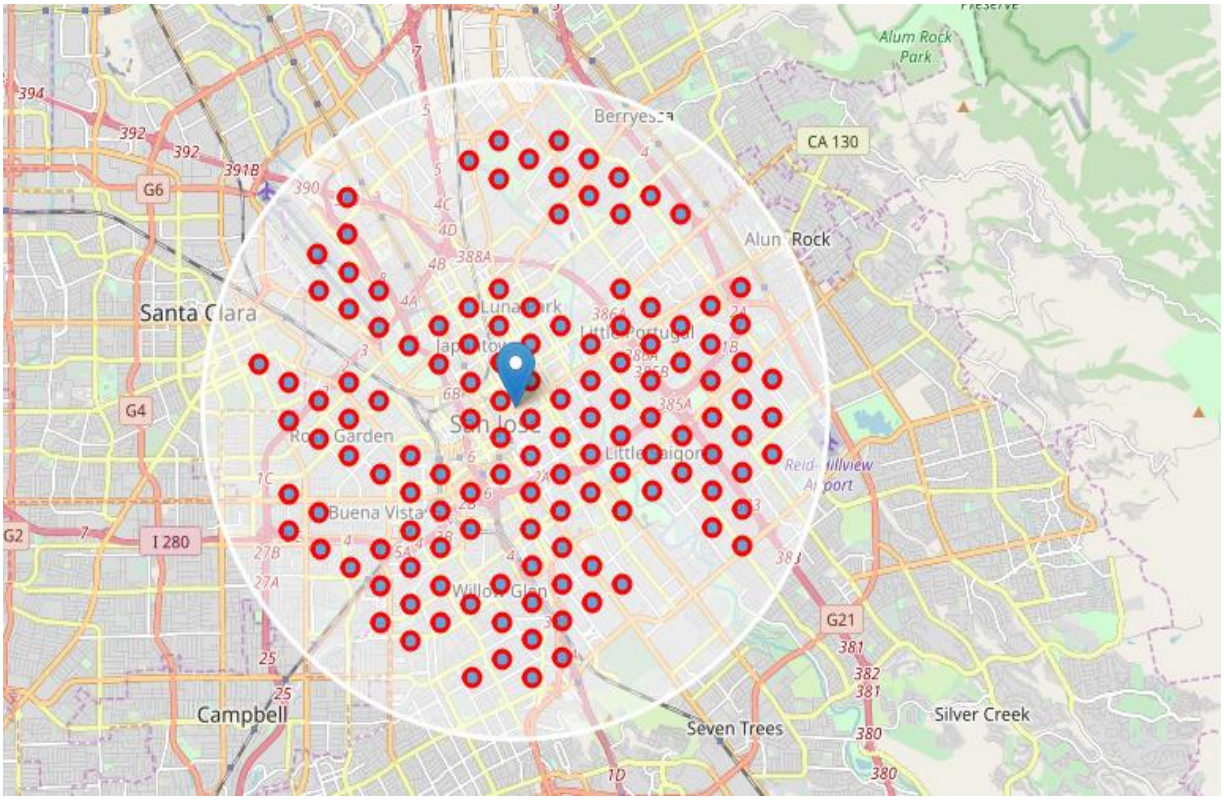
| | Address | ATM | Accessories Store | Acupuncturist | Adult Boutique | Adult Education Center | Afghan Restaurant | African Restaurant | Airport | Airport Food Court | ... | Whisky Bar | Wine Bar |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **201** | 572 MacArthur Ave, San Jose, CA 95128 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.028986 |
| **202** | 1242 Norval Way, San Jose, CA 95125 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 |
| **203** | 1318 Glenwood Ave, San Jose, CA 95125 | 0.0 | 0.0 | 0.011236 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 |
| **204** | 1501 De Anza Way, San Jose, CA 95125 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 |

***A snapshot of the location profile of our target addresses.***

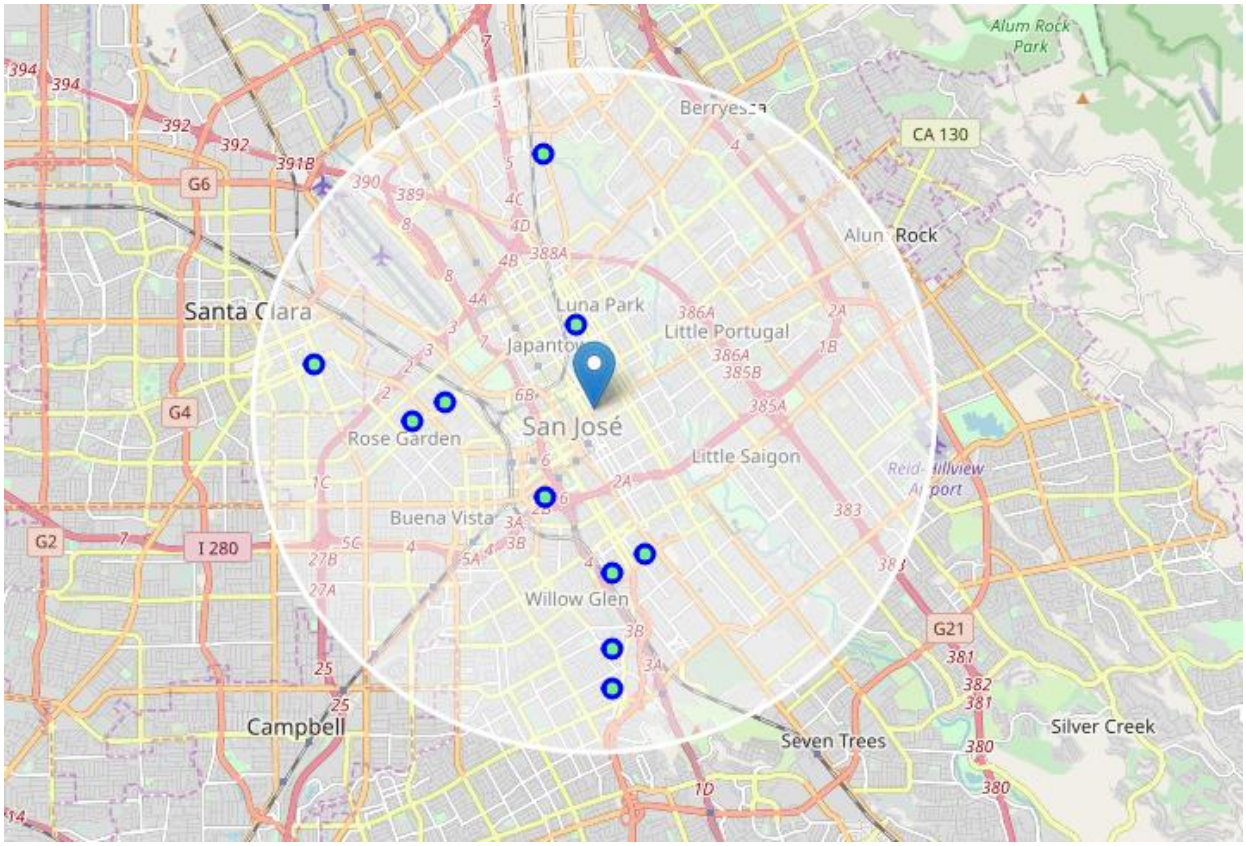
- **Segmentation/Clustering of Candidate locations:** Here we will use **KMeans** to cluster all the candidate location where the feature set will be the union of all categories present in all candidate location profiles + origin profile. after the clustering we will predict the cluster label of our origin location which will give us our cluster of interest. Now in this cluster we will find the similarity score of each location with the origin profile and retrieve the top 10 closest locations, which will constitute our final recommended location.
**Using the elbow method to find a better value of K, here the distortion value is the** Kmean.interia_ **(K=3)**



Our origin data point has been assigned cluster label 1, we will plot the **cluster 1** addresses on a folium map, this is our cluster of interest.




We will now find the distance of each location from our origin location profile using **`scipy.spatial.distance method`** `cdist` and retrieve the top 10 closest location in terms of feature similarity.



## Results & Discussion:

The corresponding **address – feature distance** dataframe:

as you can see the best match location doesn't even lie in proper San Jose but in Santa Clara, since the locations were generated algorithmically on a distance basis, we can expect our final location recommendations to be in a different location but near the target location *(the maximal radial distance is set to be 6KM)*

| | Address | Feature_dist |
|---|---|---|
| 0 | 1111 Bellomy St, Santa Clara, CA 95050 | 0.174060 |
| 1 | 1062 Pear Orchard Dr, San Jose, CA 95131 | 0.176217 |
| 2 | 2147 Coastland Ave, San Jose, CA 95125 | 0.185353 |
| 3 | 1298 Lick Ave, San Jose, CA 95110 | 0.186998 |
| 4 | 1877 Arbor Dr, San Jose, CA 95125 | 0.187818 |
| 5 | 423 Willis Ave, San Jose, CA 95126 | 0.188283 |
| 6 | 752 Emerson Ct, San Jose, CA 95126 | 0.188994 |
| 7 | 1366 Mastic St, San Jose, CA 95110 | 0.189209 |
| 8 | 718 Elm St, San Jose, CA 95126 | 0.190266 |
| 9 | 615 N 10th St, San Jose, CA 95112 | 0.190827 |

Now we will get the top 10 most common venues from our final recommendation locations. ***(first 5 locations)***

| | Address | Similarity_Score | 1st Most Common Venue | 2nd Most Common Venue | 3rd Most Common Venue | 4th Most Common Venue | 5th Most Common Venue | 6th Most Common Venue | 7th Most Common Venue | 8th Most Common Venue | 9th Most Common Venue | 10th Most Common Venue |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1111 Bellomy St, Santa Clara, CA 95050 | 0.174060 | Dentist's Office | Fast Food Restaurant | Sandwich Place | Video Game Store | American Restaurant | Mexican Restaurant | Shoe Store | Bank | Mobile Phone Shop | Gas Station |
| 1 | 1062 Pear Orchard Dr, San Jose, CA 95131 | 0.176217 | Restaurant | Event Space | Building | Bakery | Asian Restaurant | General College & University | Mexican Restaurant | School | Sandwich Place | Health & Beauty Service |
| 2 | 2147 Coastland Ave, San Jose, CA 95125 | 0.185353 | Salon / Barbershop | Spa | Church | Massage Studio | Miscellaneous Shop | Bar | Antique Shop | Event Space | Gym | Gym / Fitness Center |
| 3 | 1298 Lick Ave, San Jose, CA 95110 | 0.186998 | Church | Park | Light Rail Station | Gas Station | Clothing Store | Bridge | Playground | Airport Gate | Art Gallery | Movie Theater |
| 4 | 1877 Arbor Dr, San Jose, CA 95125 | 0.187818 | Salon / Barbershop | Dessert Shop | Spa | Bank | Mexican Restaurant | Arts & Crafts Store | Building | Clothing Store | Coffee Shop | Nail Salon |

Here we can see that recommended locations are similar to our origin location profile. Although we have only shown the top 10 most common venues, remember that on an average a location will have 40-50 venue categories and we will find even more matches if we observe the full category list.

**Conclusion:** In this report I have analyzed the location selected by the user where he/she wishes to go and clustered them basis on their venue profile, then I predicted the cluster of the origin location. I used elbow method to employ an educated guess for the value of K (number of clusters), Then I used the **similarity score (feature distance)** metric to find the top 10 closest location in the cluster of interest. Though this method doesn't guarantee that we will get 100% match of our preference list, using the feature distance surely does improve our final recommendation. We can do further analysis computing the corresponding venue category match count for each location-origin pair and refine our top 10 listing.