Data Science Capstone Project: Analyzing Venues in Johannesburg Suburbs with Machine Learning

Brighton Nkomo

August 2020

1 Introduction

Johannesburg, informally known as Jozi, Joburg, or "The City of Gold", is the largest city in South Africa and one of the 50 largest urban areas in the world [3]. It is the provincial capital and largest city of Gauteng, which is the wealthiest province in South Africa. Johannesburg is the seat of the Constitutional Court, the highest court in South Africa. The city is located in the mineral-rich Witwatersrand range of hills and is the centre of large-scale gold and diamond trade. It was one of the host cities of the official tournament of the 2010 FIFA World Cup.

In this project, I analyzed different kinds of venues using the power of k-means clustering to seek the hidden patterns about the most visited venues in each of the suburbs within the City of Johannesburg municipality.

1.1 Business Problem

Suppose that there is a contractor trying to open a restaurant within the Johannesburg municipality, how can we use the current machine learning techniques to determine the suitable locations? To begin answering the question, it is plausible to ask what makes a good location to situate a restaurant at? These are some of the key features to consider when opening a new restaurant:

- Visibility: Urban areas tend to have high car and foot traffic. Locating a restaurant around towns would hence be a good choice. However, it could be possible to find places that offer high visibility for the restaurant, but also have high crime rates. Such areas are not best suited for family-style restaurant.
- Parking: Places near parking lots would be another good choice. It would be ideal to have a restaurant with it's own parking lot.

Out[10]:

	Province	District	Local_municipality	Suburb	Metro	Latitude	Longitude	Main place
0	Gauteng	Sedibeng	Midvaal	Brenkondown	Johannesburg	-26.343133	28.073783	Alberton
1	Gauteng	Sedibeng	Lesedi	Masetjhaba View	Johannesburg	-26.388533	28.384250	Duduza
2	Gauteng	Sedibeng	Lesedi	Sonstraal AH	Johannesburg	-26.406613	28.361255	Sonstraal
3	Gauteng	West Rand	Mogale City	Ruimsig Noord	Johannesburg	-26.075359	27.865240	Krugersdorp
4	Gauteng	Ekurhuleni	Ekurhuleni	Germiston Ext 3	Johannesburg	-26.214897	28.181906	Germiston

Figure 1: The Relevant Features

- Accessibility: It could be beneficial to have a restaurant built across a road with a relatively low speed limit and high car traffic. Supposedly around freeway/highway exits. As for foot traffic, a location near urbanized areas would be ideal. Inside shopping malls, an ideal place to have a restaurant would be within or near food courts.

There are many other factors to also consider such as average income and the population of the area of interest. However, the goal of this project is to find out how urbanized an area is by finding out the most popular venues within that area (by 500 meters to be exact) and seek out hidden patterns that may reveal some additional information about a location.

1.2 The Data Set

For this project, the location data that I used was from ... After downloading the geojson file from this website and loading the data set in a jupyter notebook, the data set contains information we need such as name of the province, suburb (also known as a neighborhood in Commonwealth countries), main place, local municipality (also known as a borough in some English speaking countries) latitude and longitude coordinates of the locations. There is also other information such as the population of black people, colored people (a term referring to people of mixed race in South Africa) and white people. Although this information may be relevant when it comes to which picking out which locations have the people with the highest average income and locations that may offer high foot traffic, however this can be misleading because a population density doesn't necessarily mean more customers. So I did feature selection and decided to drop the population data. The table in Figure 1 shows the relevant features after feature selection.

With this location data, I then used the Foursquare API which is 'a local search-and-discovery mobile app developed by Foursquare Labs Inc. The app provides personalized recommendations of places to go near a user's current location based on users' previous browsing history and check-in history.' So basically this app can be used for location detection. As explained from the Foursquare Wikipedia page 'When users opt in to always-on location sharing, Pilgrim determines a user's current location by comparing historical check-in data with the user's current GPS signal, cell tower triangulation, cellular signal strength

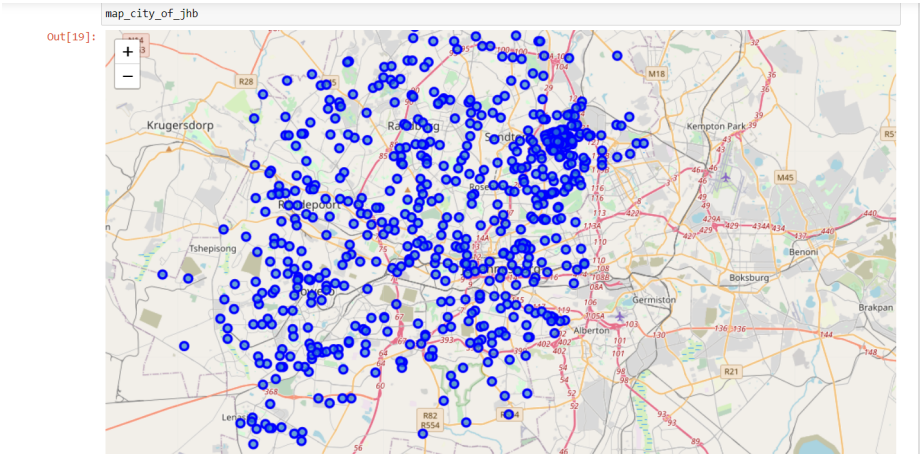


Figure 2: The Locations of the Suburbs within the City of Johannesburg Municipality

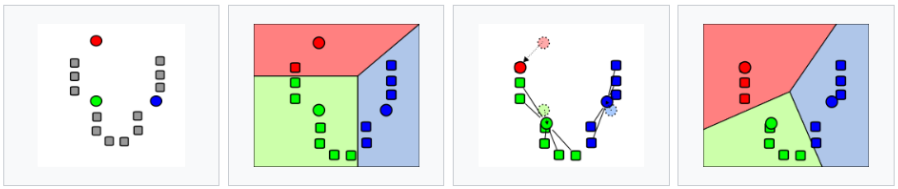
and surrounding WiFi signals.'

So Foursquare uses one's location information and visit frequency to "learn" what the user likes, which aims to improve user-facing recommendations and gauge the popularity of a venue. With the location data I had to make calls to the Foursquare API to find the most common venues per suburb in the Johannesburg, by constructing a URL to send a request to the API to search for a specific type of venues and to explore a geographical location. Also, prior to this I used the visualization library, Folium, to visualize the Suburbs in Johannesburg and find out how big the size of the data set was. In the City of Johannesburg municipality, there were 659 Suburbs as shown on the map in Figure 2 (each of the blue points are a suburb within the local municipality).

Since I was analyzing the most common venues within 500 meters per suburb, I decided to limit the number of the most common venues to just 100 venues.

2 Methodology

Since there are no labels in the data set for this particular problem, unsupervised learning is best suited to solve this problem. In particular clustering algorithms such as k-means clustering are good candidates for dealing with location data. The k-means clustering algorithm creates clusters automatically and takes the mean values of the to determine the cluster centers.



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

Figure 3: Demonstration of the Standard Algorithm

2.1 k-Means

'The k-means algorithm is used to partition a given set of observations into a predefined amount of k clusters.' The algorithm begins with randomly generated set of k centroids or cluster center (μ). The algorithm update step assigns all the observation x to their closest cluster center (see eq. 1). 'In the standard algorithm, only one assignment to one center is possible. If multiple centers have the same distance to the observation, a random one would be chosen.'

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - \mu_i^{(t)} \right\|^2 \leq \left\| x_p - \mu_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\} \tag{1}$$

The cluster centers would then be moved to a new position by calculating the mean of the assigned observations with respect to their cluster centers (eq. 2).

$$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{2}$$

This updating process carries on until there are clustering centers that can be shifted/re-positioned and when all observations cannot be assigned to a new cluster center. This is best illustrated in the figure 3 that I took from wikipedia

This means that the k-means algorithm tries to optimize the *objective function* (eq. 3).
Since there's only a finite number of possible assignments for the finite number of cluster centers and observations and also that each iteration is an improved solution, it is guaranteed that the algorithm will stop in a *local minimum*.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{3}$$

$$\text{with } r_{nk} = \begin{cases} 1 & x_n \in S_k \\ 0 & \text{otherwise} \end{cases}$$

The main disadvantage of k-means is its dependency on the initial randomly chosen cluster centers. The cluster centers could end up splitting the the clusters whilst observations that clearly belong to other clusters get grouped together especially when some of the cluster centers are attracted by outliers. In other words, the k-means is not robust to outliers.
The common solution to this problem is to have multiple clusterings with different starting positions. Then after, the most frequently occured clustering is considered as correct. The textbfk-means++ algorithm is also another solution to solving this issue which was proposed by Arthur and Vassilvitskii [2]. The k-means++ tries to distribute the initial chosen cluster centers over the given data so as to minimize the probability of bad clustering outcomes. According to Arthur and Vassilvitskii, the initial cluster centers are set as follows:

1. Take uniformly a random data point from the data X and mark it as centroid c_1
2. Choose another centroid c_i with the probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$ where $D(x)$ denotes the shortest distance from the data point x to its closest, already chosen centroid.
3. Repeat 2. until all k initial centroids are chosen.

After these steps, the standard k-means algorithm as described from the beginning of this section is performed. The authors also showed that with this initialization algorithm, k-means++ approximately can be computed in $O(\log n)$, compared to $O(n^{dk+1} \log n)$ for the standard algorithm. Therefore there k-means++ algorithm is faster that the standard algorithm.

2.2 Data Preparation

- **data cleaning**: In this project it was necessary to clean the data by deleting rows with missing values. If the latitude and longitude coordinates are missing, then the Foursquare API would not be able to retrieve any information about the unknown locations. Perhaps taking the mean of the longitude and latitude coordinates would have also worked because the coordinates do not differ by a large order of magnitude. In fact, the latitude and longitude coordinates differ by an order of magnitude at most -1. No outliers were removed from the data, because this would have been very tricky to do for all 659 suburbs.
- **feature selection**: Attributes such as the population of different races, total population within a suburb and names of district municipalities were dropped. In general, redundant features were eliminated.

- **feature scaling**: Since the range of values of raw data varies widely, in some machine learning algorithms, initially the objective functions did not work properly without normalization. This was because the k-means standard algorithm uses the Euclidean distance to calculate the distance between two points. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features were normalized so that each feature contributes approximately proportionately to the final distance.

3 Results

To be continued...

4 Discussion

To be continued...

5 Conclusion

"I always thought something was fundamentally wrong with the universe" [1]

References

[1] D. Adams. *The Hitchhiker's Guide to the Galaxy.* San Val, 1995.

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[3] Th. Brinkhoff. Principal agglomerations of the world.