**IBM APPLIED CAPSTONE PROJECT**

**Opening a New Shopping Mall in Amsterdam**

**By: Dinesh Kandimalla**

**FEB 2020**

****

1.**Introduction**

**1.1 Background**

Amsterdam is located in the western Netherlands, in the province of North Holland, although it is not is capital which is Haarlem, Amsterdam is the nominal capital. It is also the country’s largest city. The Dutch being the nation of merchants and travelers, until very recently were shy when shopping was concerned. It is only during the last 15 years, that the consumer society ideas, took in the Netherlands over the Calvinist mentality of a simple, modest life and self-limitation. Today Amsterdam is a great place for shopping because it has imports from all over the world. Along with the world’s top brands, you will find here products from many small and underdeveloped countries.

Shopping malls have a prominent place in the configuration of modern cities, affecting the daily activities, social relationships and mobility of their inhabitants.

**1.2 Problem**

Opening a new shopping mall in a city is not an easy task, Location of the shopping mall is one of the important decisions whether mall will be a success or failure. This project aims to find the right location for the investors to open a new shopping mall in the city of Amsterdam.

**2. Data Acquisition and Cleaning**

**2.1 Data sources**

Neighborhood data in the city of Amsterdam can found in Kaggle dataset [here](https://www.kaggle.com/dushyantsapre/amsterdam-neighbourhood-and-venues). Dataset contains Neighborhood names along with their latitudes and longitudes. Venue data in each Neighborhood is collected using FOURSQUARE API and get different venues and category of each venue along with their latitude and longitude.

After collecting both the datasets, merged both the datasets based on Neighborhood column. And checked if any rows are empty, if rows are empty with no venue data then that row is removed from the dataset.

After data cleaning, there were total 4344 samples and 7 features.

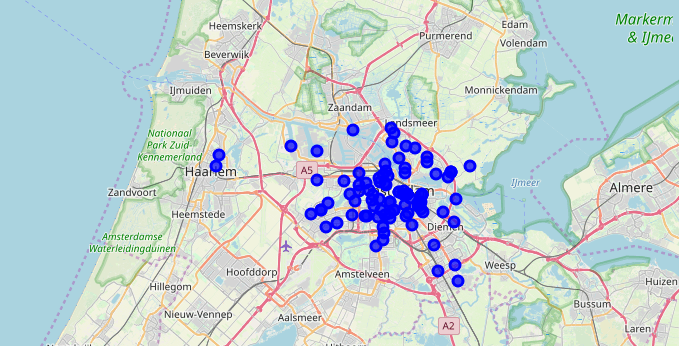


Figure 1: Map of Amsterdam city with Neighborhoods plotted as blue.

**3. Methodology**

Initial data frame of Amsterdam Neighborhoods is shown in the below figure.



Next, we use the Foursquare API to get the top 100 venues that are within a radius of 500 meters from the neighborhood. We need Foursquare ID and Foursquare Secret Key to call api , for that we should register for Foursquare developer account. We then make Foursquare calls by passing in the geographic coordinates of neighborhoods in loop. We get the venue data in JSON format, then we will get venue coordinates and category from that. From this data we can see how many venues were obtained for each neighborhood and get how many unique categories were returned for all venues. From our data we see that 293 unique venues were returned. Now we will analyze each neighborhood by grouping rows of each neighborhood and taking mean frequency of each category in neighborhood. Since we need to analyze about “ Shopping Mall” data , we filter it in each neighborhood and put it in new dataframe.



Figure 2. Sample Data Frame of venues in neighborhood

Last, we perform clustering on the data by using k-means. K-means clustering identifies k centroid points and assign each neighborhood to one of k centroids. The initial k centroids are also from the data points. K-means is one of the popular and simplest clustering unsupervised algorithms. Typically unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labeled, outcomes. We assign each neighborhood to one of clusters centroids based on frequency of presence of “Shopping Mall”. Now we can plot the clustered data points of neighborhoods with different color for each cluster. In the map we can observe which neighborhoods have high number of shopping malls and which neighborhoods have fewer shopping malls. By that observation we can be able to answer the question which neighborhood is most suitable to open a shopping mall.

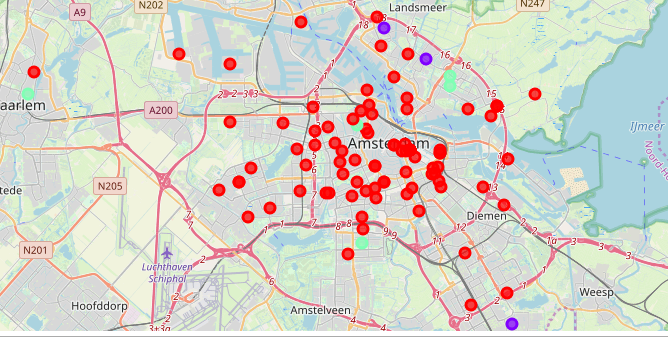
**4. Results**

The results from k-means clustering show that we can categorize neighborhoods into 3 clusters based on frequency of occurrence of “shopping mall”.

Cluster 0: Neighborhoods with least and fewer number of shopping malls.

Cluster 1: Neighborhoods with moderate number of shopping malls.

Cluster 3 : Neighborhoods with most number of shopping malls.



**5. DISCUSSION**

From the observations in the map in the results section we note that most of the neighborhoods in Amsterdam city are in cluster 0, which means there are very few number of shopping malls. Whereas in cluster 1 has moderate number of shopping malls which means there is still chance to open new shopping malls and last there are more shopping malls in cluster 2 which means there is heavy competition and there is no chance to open a new shopping mall and get succeed. Therefore from the results, this project recommends the property developers to open a new shopping malls in the neighborhoods in cluster 0 as there is room for more shopping malls with little to no competition. Property developers to stand out from the competition can also open new shopping malls in neighborhoods in cluster 1. Lastly property developers are advised to avoid opening new shopping malls in neighborhoods of cluster 2 as currently there is oversupply of shopping malls and suffering from intense competition.

**6. Future work**

In this project we considered only frequency mean of shopping malls in each neighborhood, we can extend this by considering population and average income of people living that area. However to the best of my knowledge there is no such data present for neighborhoods. Future work can be to get that data or devise a methodology to get that data and cluster based on it.

In addition to this, this project made use of free Foursquare developer account to get the top venues in neighborhood, which is limited to number of calls and results returned. Future work includes calling using Paid account so that we can have more calls to API and more results are returned.