BATTLE OF INDIAN CITIES

INTRODUCTION: BUSINESS PROBLEM

In this project we will try to find the optimal location among Indian major cities for the initial launch of any specific CONSUMER GOODS products. This will also be helpful for already established companies who wish to expand their business to other cities of India, and do not have enough data to figure out which newer cities are more likely to welcome their PRODUCT A and which will be the best site for their PRODUCT B and so on.

This project categorizes the major cities with similar characteristics into clusters and helps the companies based on **CPG (Consumer Packed Goods)** and **FMCG (Fast-Moving Consumer Goods)**, to plan for the deployment and distribution of their products and efficiently target different cities corresponding to their different products.

DATA ACQUISITION AND CLEANING

Based on the description of our problem, we will be needing a list of major cities (according to the population and various other characteristics) for this project. The population data, literacy rate, density per sq. km and sex-ratio are some of the main data that we need to group the cities into clusters of similar cities. I got these from the link below:

- **https://www.nriol.com/india-statistics/biggest-cities-india.asp**

The coordinates (latitudes and longitudes) for each of these major cities were required in order to get the nearby venues so that we could group these cities into clusters based on the type of venues in the vicinity of the cities.

- **geopy** is a Python client for several popular **geocoding** web services. **geopy** helped me locate the coordinates of the cities using **geocoders**.

Further data regarding nearby venues was gathered using the ***foursquare API.*** Different cities vary greatly in their size and lifestyle. So, I set the RADIUS of 10 km and limit of 1000 so that I won't miss out anything.

- **https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, VERSION, LATITUDE, LONGITUDE, RADIUS, LIMIT)**

FEATURE SELECTION

- **https://www.nriol.com/india-statistics/biggest-cities-india.asp**

After acquiring and cleaning the data, we were left with a dataframe consisting of 100 rows and 8 columns. On further analysis of each column's relevance to our project, it was clear that some features were redundant and needed to be removed from the dataframe. For example, the 'Indian States' and the 'Main Language' columns are not really significant to our purpose.

- **https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, VERSION, LATITUDE, LONGITUDE, RADIUS, LIMIT)**

Foursquare API's response is saved in the form of a json file which contains a lot of unwanted data. We need to be very careful in filtering out what we need. For example, first we need to convert (APIRESPONSE['response']['groups'][0]['items']) into a dataframe. Further, we need to drop all other columns except the following columns: -
'venue.name','venue.categories','venue.location.lat','venue.location.lng'.

METHODOLOGY

Our first table (df_major_cities) contains 100 rows and 8 columns. On further analysis of each column’s relevance to our project, it is clear that some features are redundant and need to be removed from this dataframe. For example, the ‘Indian States’ and the ‘Main Language’ columns are not really significant to our purpose.

Let's add some other important data for each of these cities using another dataframe(cities_r2.csv) that we found on Kaggle.

https://www.kaggle.com/zed9941/top-500-indian-cities

Let's select each city from the df_major_cities and get all the appropriate data that we can, from the second dataframe i.e., df_cities_r2 dataframe.

Moving on, we find that we also need to strip the names of the cities otherwise the city names from both of the dataframe will not match.

Now, let's find out how many of the cities from the 1st dataframe(df_major_cities) are present in the 2nd(df_cities_r2) dataframe. On further manual checking, we find that some cities are spelled differently ans some contain an alternate version of their many known names.

This force us to manually alter the corresponding csv file(cities_r2.csv) of the second dataframe(df_cities_r2) using Microsoft Excel and then read it again for further analysis.

Two of the cities were completely missing from the csv file(cities_r2.csv). So, I extracted the required data for them from the following links:

Kalyan & Dombivali, Maharashtra: -

https://www.census2011.co.in/census/city/369-kalyan-and-dombivali.html

Ghaziabad, Uttar Pradesh: -

https://www.census2011.co.in/census/district/511-ghaziabad.html

Now let's merge both dataframes into one.

Let's remove the redundant data from our table.

Now let's use geopy for finding the coordinates of each of the cities.

Let us define a function which will do all the heavy lifting for us. We just need to call this function and it will return the latitude and longitude of the cities.

Let us visualize the data that we have so far.

Let’s find if any column contains a NaN

We can see that the 95th and the 99th element of the column 'Density(/km2)' has NaN values.

This implies that only one column contains NaN values. Now let us replace the NaN values by the mean of the values present in that column.

Now they have been replaced by the mean of the values in this column.

Premature Clustering of Cities

Let us prematurely cluster our cities just for fun :)

WE WILL BE USING K-MEANS MACHINE LEARNING TECHNIQUE.

The K-means clustering algorithm is used to find groups which have not been explicitly labeled in the data. This can be used to confirm business assumptions about what types of groups exist or to identify unknown groups in complex data sets. You can use the k-means algorithm to maximize the similarity of data points within clusters and minimize the similarity of points in different clusters. Hence, it is the most suitable algorithm to use here as we are trying to group a number of cities according to their similarities.

Now let us drop the 'Cluster Labels' column since we do not need it anymore. Moreover, it will only be a nuisance for any of the further analyses.

EXPLORE THE INDIAN CITIES

FOURSQUARE API

It is time for foursquare API to come to the rescue and provide us with the data of the venues surrounding the city.

For this we will be needing only the coordinates of the cities.
Let us define a function which can be applied to a dataframe in order to extract the category of each venue from its corresponding element of 'venue.categories' column present in that row.

We will need the following credentials in order to use the foursquare API.

Let us set a RADIUS & LIMIT according to the sizes of the cities.

Using the Python requests module we can send HTTP requests. We have already imported it and we will be using it here.

We will be using the json_normalize module from pandas in order to normalize the semi-structured JSON data into flat tables.

The data for all the venues obtained from the requests using Foursquare API is stored in df_venues for further analyses.

The above cell's output shows that we get None value for two cities, namely, Amravati (row index 66) & Jalgaon (row index 97). So, let us drop these two rows from our df_merged dataframe.

Let us merge the dataframes df_merged & df_venues into a new dataframe df_final.

Let us remove the unwanted columns from the df_final dataframe, so that we can create a dataframe containing the onehot vector form of all the appropriate columns for our clustering.

NORMALIZING THE DATA

We will be using the preprocessing module from the sklearn library that we have already imported.
Let us store the column names in a separate list which will be used later to rename the columns of the normalized dataframe.

We first create a minimum & maximum preprocessor object.

Run the normalizer on the dataframe using fit_transform.

Convert the returned numpy array into a Dataframe again.

Now with the help of stored column names in the list_columns variable, we can reassign the column names to the normalized dataframe.

Let us group the data according to the cities to find out how many venues we have got w.r.t each city.

Let us again store the names of the columns which will be used to generate the onehot vector form.

Let us create a dataframe containing the onehot vector form(based on the columns stored in the variable list_columns) in each row.

Now let us group the cities and take the mean of the onehot values. We are preparing for KMeans analysis.

Let us take no of cluster=8.

Now let us create a dataframe that contains the 'Name of the cities', 'Latitudes', 'Longitudes' and 'Cluster Labels'. This will be useful for plotting the city clusters using folium module.

Folium is a Python library used for visualizing geospatial data. We will be using it to aid us in generating a map containing all the cities clustered together w.r.t their similarities.

RESULTS & DISCUSSIONS

Our analysis shows that most of the cities fall under the clusters 0,2 and 6. And these cities' locations range across all of India.

The only city located in the extreme North, i.e., Srinagar falls under a completely separate cluster (i.e., Cluster 7). It was quite obvious as in contrast to other states of India, Jammu and Kashmir had its own constitution, flag and administrative autonomy. Srinagar is a hilly region together with a majority of Muslim population. All these factors add up and make Srinagar a completely unique city.

Similarly, the city in the extreme North-east part of India, i.e., Guwahati also falls under a completely separate cluster (i.e., Cluster 5) together with Aurangabad being the only other city to fall in its cluster. On further observation, we will find that these two cities are the ones with the least sex-ratio.

Most of the cities located in the Central part of India fall under the cluster 6 category. This is another observation from the cluster map.

The beauty of machine learning algorithms is that it can identify such relationships among the variables OR in our case similarities and dissimilarities between the cities, which can be super hard or sometimes even impossible to observe through human analysis (as data is huge most of the times). Similarly, we can observe that most of the cities are grouped into clusters on the basis of some underlying relationships. In order to figure it out, the interested companies can carry on further analysis.

CONCLUSION

Purpose of this project was to provide the interested companies with an initial option to target different cities corresponding to their different products. As the similar cities are grouped into clusters now, the companies can have different plans for cities falling under different clusters.

Some outliers which fall into completely separate clusters can be skipped by these companies.

Final decision on optimal business decisions will be made by stakeholders based on the specific characteristics of clusters.

THE END.