Emami Agrotech – Live Project

1. **Project Title –** Consolidation of Primary Drop point/ consolidation
2. **Background –** Emami Aggrotech is in a commodity goods business where the finished goods are transported from the 4 factories to the 3000+ distributors all across India. Following are the two categories of product -
   1. Best Choice – Mass Product (70% of total volume)
   2. Healthy and Tasty – Premium (30% of total volume)

For this RFQs are raised separately for all the nodes. This is resulting in a large number of quotes, which is impacting high bidding time and making the process of Reverse Auction cumbersome.

3. **Problem Statement -** Reducing the number of lanes for quotation to reduce time and increase efficiency of Reverse Auction. Preferably 15-20% reduction in number of lanes.
4. **Objective –** Dividing the distributor locations into clusters of radius 50km so that a single quotation (RFQ) can be taken from the vendors for each cluster.
5. **Methodology and Project Plan –**

5.1. Extracting Latitude and Longitude data from the distributor data

This is done by using the geopy library of python. The distribution location data is plotted on the map of India to get a better idea on the data spread.

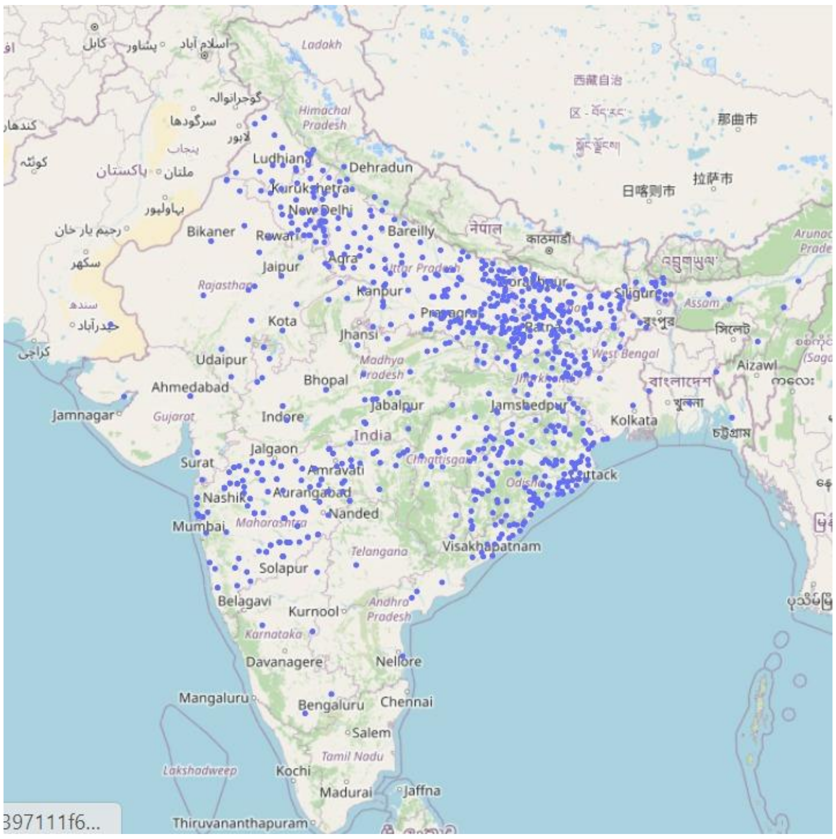

Fig 1. – Distributor Locations

5.2. K Means clustering

This technique will be applied on the data set to cluster the locations based on their proximity. Python will be used to process the data. Also, the mean distance among the nodes of each cluster is to be examined to judge the feasibility of the vendor to provide a single quote for the entire cluster. Subsequently we can think of increasing or decreasing the number of clusters.

5.2.1. Iteration 1 – 3 No. of Clusters

Starting with 3 number of clusters, K Means clustering is applied on the data and the maximum distance between two points is calculated.

- If this distance is more than 100 km, we have to increase the number of cluster to reduce the max inter-point distance to less than 100 km (to achieve a cluster of radius 50 km).
- If this distance is less than 100 km, we have to decrease the number of cluster and recheck the max inter-point distance in each cluster.

The figure below shows the clustered distributor data on the map of India

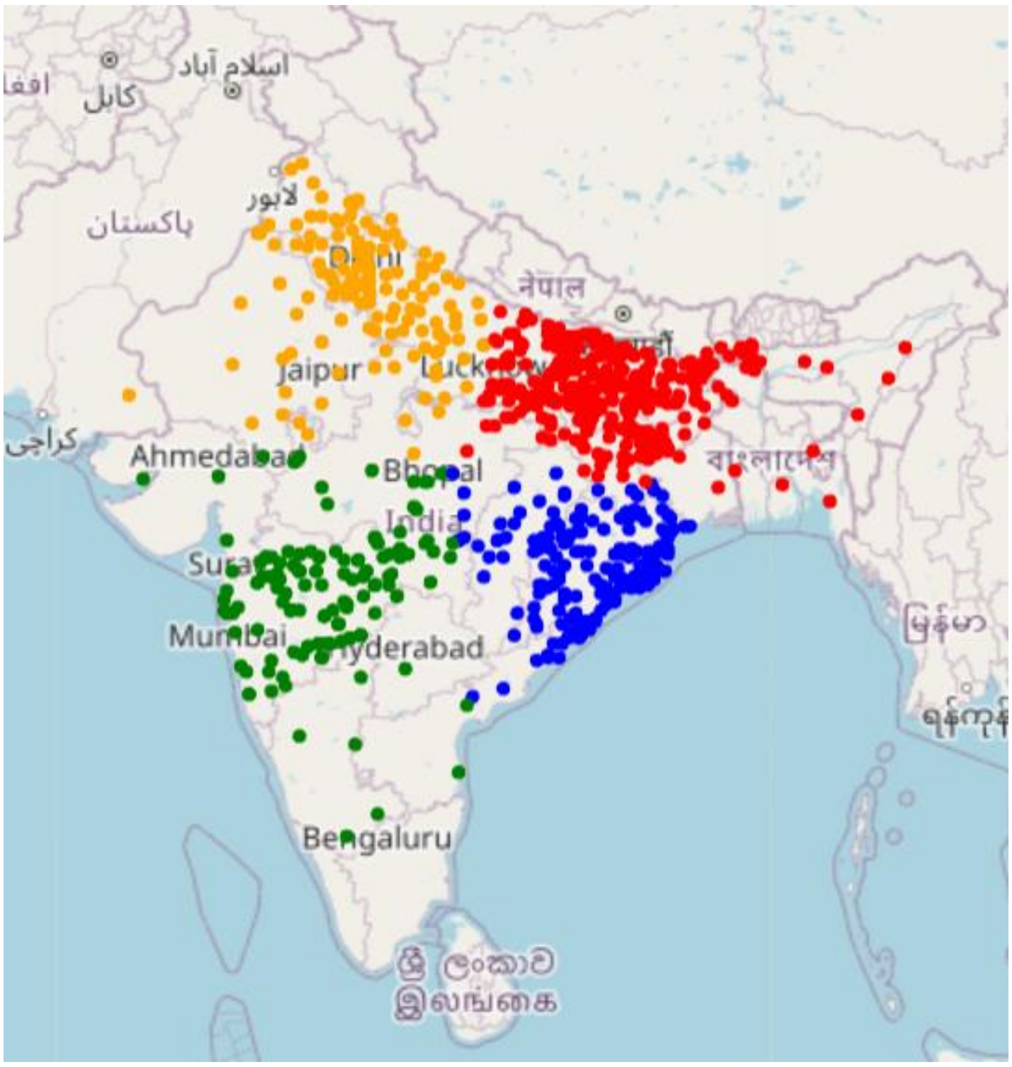

Fig 2. – Clustered distributor locations with n = 4

Following are the maximum inter-point distance in each cluster.

Cluster 1 – 1427.67 km

Cluster 2 - 1422.94 km

Cluster 3 – 911.26 km

Cluster 4 – 1133.18 km

It is very clear from the result that we need to significantly increase the number of clusters.

Now by calculating the max interpoint distance for each cluster we have to determine the optimum number of clusters for which the interpoint distance for each cluster will be less than 100 km. Once that is fixed we can proceed with the clustering of the data and plot them on the map of India and finally extract the clustered data. Below are some of the comparative data of number of cluster versus the mean of inter-point distances and max of inter-point distance

N = 50, Mean (Inter-point distance) = 210, Max (Inter-point distance) = 387

N = 100, Mean (Inter-point distance) = 107, Max (Inter-point distance) = 222

N = 150, Mean (Inter-point distance) = 64, Max (Inter-point distance) = 157

N = 200, Mean (Inter-point distance) = 44, Max (Inter-point distance) = 129

N = 220, Mean (Inter-point distance) = 38, Max (Inter-point distance) = 97

So, 220 seems to be a optimum number of cluster, where the max inter-point distance between a cluster id 97km, which is satisfying our objective of having clusters with radius of 50km. So, we can proceed with 220 number of clusters.

5.2.2. Clustering with n = 220.

By clustering with n = 220, we will get clusters in which the max inter-point distance in each cluster will be less than 100km i.e. the clusters will be in a region with radius 50 km.

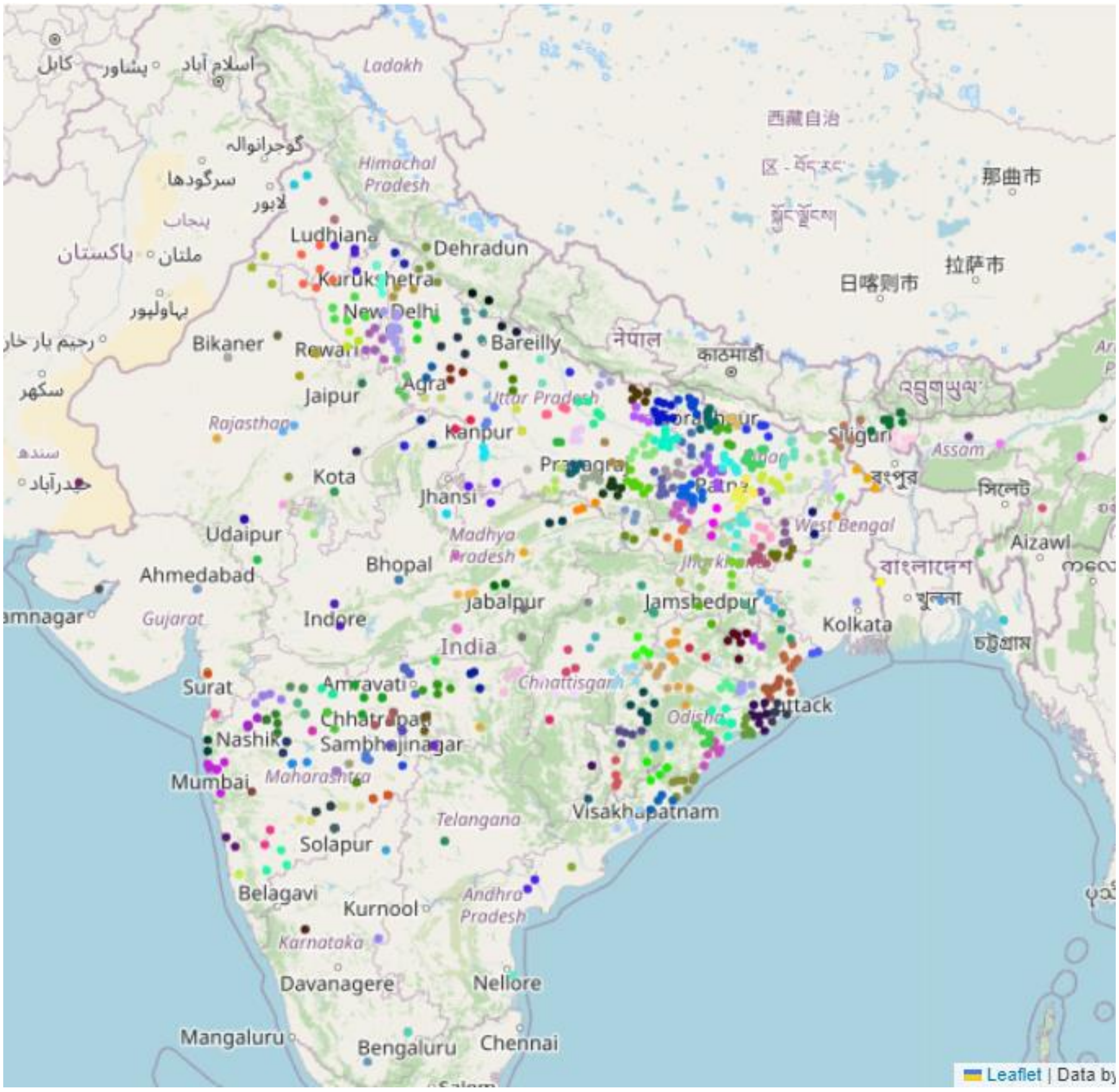

Fig 3. – Clustered distributor locations with n = 220

The data shown above is our final clustered data with each cluster lying in a region of radius less than 50 km.

6. Results

The 775 distributor data points is reduced to 220 clusters.

Appendix

1. Extracting Latitude and Longitude from Address.

```
import pandas as pd
from geopy.geocoders import Nominatim

# Load the Excel file with addresses
df = pd.read_excel(r"C:\Users\HP\Desktop\E\Distributor_Data.xlsx")

# Initialize geocoder
geolocator = Nominatim(user_agent='my_app')

# Define a function to get latitude and longitude from an address
def get_lat_long(address):
    location = geolocator.geocode(address)
    if location:
        return location.latitude, location.longitude
    else:
        return None, None

# Apply the function to the 'address' column in the dataframe
df['latitude'], df['longitude'] = zip(*df['Address'].apply(get_lat_long))

# Save the dataframe to a new Excel file
df.to_excel('file_with_coordinates.xlsx', index=False)
```

2. Data Clustering and estimating Max inter-point distance in each cluster.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from sklearn.cluster import KMeans
from geopy.distance import geodesic

df = pd.read_excel(r"C:\Users\HP\Desktop\E\Distributor_Data.xlsx") # Replace 'data.csv' with the name of your CSV file

X = df[['Latitude', 'Longitude']]
kmeans = KMeans(n_clusters=220, random_state=0).fit(X)
df['cluster'] = kmeans.labels_

def calc_max_distance(cluster_df):
  max_distance = 0
  for i, row in cluster_df.iterrows():
    for j, other_row in cluster_df.iterrows():
      if i < j:
        distance = geodesic((row['Latitude'], row['Longitude']), (other_row['Latitude'], other_row['Longitude'])).km
        if distance > max_distance:
          max_distance = distance
  return max_distance

max_distances = []
for i in range(220): # Replace 3 with the number of clusters
  cluster_df = df[df['cluster'] == i]
  max_distance = calc_max_distance(cluster_df)
  max_distances.append(max_distance)


print(max_distances)
```

3. Plotting the clustered data in the map.

```
import random
colors = ['#%06x' % random.randint(0, 0xFFFFFF) for i in range(220)]


india_map = folium.Map(location=[20.5937, 78.9629], zoom_start=5)


for i in range(len(df)):
  folium.CircleMarker(location=[df.iloc[i]['Latitude'], df.iloc[i]['Longitude']], radius=0.2, color=colors[df.iloc[i]['cluster']]).add_to(india_map)


india_map.save('clustered_dist_1.html')


india_map
```

4. Exporting the clustered data into Excel

```
with pd.ExcelWriter('clustered_data_final.xlsx') as writer:
  for i in range(220):
    cluster_data = df[df['cluster'] == i]
    if len(cluster_data) > 0:
      cluster_data.to_excel(writer, sheet_name=f'Cluster {i}', index=False)
```