Introduction to the IBM's applied data science capstone project

This notebook introduces a business problem which is meant to be solved using data science and machine learning and will also walk through the steps on how the data is collected, cleaned and prepared to be fed in the machine learning model.

Step 1: Introduction / Business understanding

I have chosen this data science project as an opportunity to work on a business idea that I formed in my head since college. My dream is to set up a chain of artificial turfs for various sports and activities like football, cricket, fitness workshops etc. This has huge upside potential in India, especially in Kolkata which is my hometown, because:

- The parks and fields available to the public are not well maintained and well suited for a seemless playing experience.
- The parks and playing fields are not always available (to suit the needs of working class) due to lack of infrastructure (proper lighting at night, nets, designated playing areas, level playing fields, etc.)
- Interrputions caused due to religious ceremonies, fairs, political gatherings etc.
- Artificial turfs equipped with sheds will be a bonus as turfs can also be utilized in bad weather (like rainy days or in hot summer days etc.)

If a business model is followed where we offer our target audience a designated playing time with decent infrastructure, people will be more likely to pay fees as people are becoming more and more health conscious and trying hard to be fit and active in their fast paced lives.

One of the most import decisions to make before starting this business is to determine the location where the turf has to be set up to reach maximum potential. In this data science project I have tried to utilize which I have learnt from the 9 courses and designed a machine learning model using the K-means algorithm to find the optitum location to set up the chain of artificial turfs.

Step 2: Business problem

To determine the location or neighbourhood where setting up turfs would be the most beneficial we will use an unsupervised learning algorithm which is k-means clustering. Using foursquare data we will gather data of nearby venues of each neighbourhood or wards. By exploratory data anaylsis we can determine the ratio of occurence of most common venues of each ward.

Then we will feed the dataset into the k-means clustering algorithm to cluster the wards to their likeness to each other. Our goal is to find out which clusters will be beneficial for us to set up artificial turfs.

- First we will discard the cluster where there are the most parks and playgrounds. People living near playgrounds will thick twice paying for artificial turfs when they can play for free.
- Secondly, we will look out for wards which has ease of connectivity. Wards with more number of bus stops, train stations will be beneficial for us.
- We will also look for schools, colleges and commercial complexes and the age group which are most interested in sports are children, young adults and adults.

Step 2: Data Understanding.

We will use the K-means clustering algorithm to segment and cluster neighbourhoods based on the nearby venues which we will obtain from the Foursquare API.

To use the foursquare API we will need to get hold of a dataset which will contain the list of neighbourhoods along with latitude and longitude values of the neighbourhoods. But, as the neighbourhoods in the city of kolkata are quite large in area, we will instead use Wards (smallest unit areas designated by the Kolkata Municipal Corporation of ease of governance).

The wards are governed by local councillors elected by the public in the municipality elections. There are 143 wards in kolkata maintained the Kolkata Municipal Corporation (KMC). The wards are designated by numbers. The KMC website contains the information regarding the wards like Name of the councillors and office address. We will need the office Address to figure out accurate latitude and longitude values of the wards.

Step 3: Data Collection.

It was difficult to scrape data frome KMC website because the dataset was divided into 15 pages. I searched on google and find out that someone already prepared the same dataset I was looking for.

Here's the link to the dataset:

http://dev.opencity.in/dataset/50dad059-13b7-499d-aa61-9bf78fef7267/resource/dfdbde68-68e2-4c63-ac2d-faea1deb998a/download/kolkota-kmc-councillors-2018-1.csv

Copy paste this link in a new tab of your browser.

Let us first import all the libraries.

```
pip install folium
```

```
Requirement already satisfied: folium in /usr/local/lib/python3.6/dist-packages (0.8.3)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from foliu
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from fol
Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: jinja2 in /usr/local/lib/python3.6/dist-packages (from fo
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-package
```

```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files

from geopy.geocoders import Nominatim # convert an address into latitude and longitude values

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors
import matplotlib.pyplot as plt


# import k-means from clustering stage
from sklearn.cluster import KMeans

import folium # map rendering library

# libraries to read dataset from google drive
```

```
import requests
from io import StringIO

print('Libraries imported.')
```

```
Libraries imported.
```

Now we will install Opencage which is a free geocoding API. We will use forward geocoding which is converting Address into Latitude and Longitude values.

```
pip install opencage
```

```
Collecting opencage
  Downloading https://files.pythonhosted.org/packages/44/56/e912b950ab7b05902c08ebc3eb6
Requirement already satisfied: Requests>=2.2.0 in /usr/local/lib/python3.6/dist-packages
Collecting backoff>=1.10.0
  Downloading https://files.pythonhosted.org/packages/f0/32/c5dd4f4b0746e9ec05ace2a5045
Collecting pyopenssl>=0.15.1
  Downloading https://files.pythonhosted.org/packages/b2/5e/06351ede29fd4899782ad335c2e
     |████████████████████████████████| 61kB 4.9MB/s
Requirement already satisfied: six>=1.4.0 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/li
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-pack
Collecting cryptography>=3.2
  Downloading https://files.pythonhosted.org/packages/fa/af/fe27c2cd875bb0621d7fedd8b10
     |████████████████████████████████| 3.2MB 6.2MB/s
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.6/dist-packages (fr
Collecting setuptools-rust>=0.11.4
  Downloading https://files.pythonhosted.org/packages/e5/e3/ede8b8545e7ce7f5fde88d1a89c
Requirement already satisfied: pycparser in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: toml>=0.9.0 in /usr/local/lib/python3.6/dist-packages (f
Collecting semantic-version>=2.6.0
  Downloading https://files.pythonhosted.org/packages/a5/15/00ef3b7888a10363b7c402350ed
Installing collected packages: backoff, semantic-version, setuptools-rust, cryptography
Successfully installed backoff-1.10.0 cryptography-3.4.1 opencage-1.2.2 pyopenssl-20.0.
```

```
from opencage.geocoder import OpenCageGeocode
```

```
# @hidden_cell

key = 'c0ca14c914034184a6792d41901f6a1e'
geocoder = OpenCageGeocode(key)
```

Looking at the office addresses of the KMC dataset, I saw that some of the addresses were not written properly to be fed into the OpenCage API. So I cleaned the dataset and upload it on my Github.

```
url = 'https://raw.githubusercontent.com/dipayan10/Kolkata_dataset/main/df_kolkata_final.csv'
```

Now we read the CSV file from the url of my GitHub and name it df_kolkata.

```
df_kolkata = pd.read_csv(url)
df_kolkata.head()
```

	Ward No.	Councillors	Office Address
0	1	SMT. SITA JAISWARA	9A, Gopi Mondal Lane, Kolkata-700002
1	2	SMT. PUSPALI SINHA	10, Rani Debendra Bala Rd Rishi, Paikpara Kolk...
2	3	DR SANTANU SEN	Seth Lane, Kolkata-50
3	4	SHRI GOUTAM HALDAR	9A, Raja Manindra Road, Kolkata-37
4	5	SHRI TARUN SAHA	67B, Khelat Babu Lane, Kolkata-37

Now we try to learn more about the dataset, shape of the dataset and the data types.

```
df_kolkata.shape
```

```
(144, 3)
```

```
df_kolkata.dtypes
```

```
Ward No.           int64
Councillors       object
Office Address    object
dtype: object
```

Now we try to run a trial of the Opencage API to see if it works.

```
query = df_kolkata['Office Address'][0]
query
```

```
'9A, Gopi Mondal Lane, Kolkata-700002'
```

```
api_url = f'https://api.opencagedata.com/geocode/v1/json?q={query}&key=c0ca14c914034184a6792d
data = requests.get(api_url).json()
print(data['results'])
```

[{'annotations': {'DMS': {'lat': "22° 33' 45.46800'' N", 'lng': "88° 21' 46.94400'' E"},

As we can see, the 'results' key of the json contains all the information. Then we try to seperate the lat and long values from the rest of the json.

```
demo_Latitude = data['results'][0]['geometry']['lat']
demo_Longitude = data['results'][0]['geometry']['lng']
```

Now we create a new dataframe called latlong_dataframe and append a series in it which contains the values of the following columns.

```
my_columns = ['Office Address', 'Latitude', 'Longitude']
latlong_dataframe = pd.DataFrame(columns = my_columns)
```

```
latlong_dataframe.append(
       pd.Series(
    [
           df_kolkata['Office Address'][0],
           demo_Latitude,
           demo_Longitude
       ],
   index = my_columns
),
  ignore_index= True
)
```

	Office Address	Latitude	Longitude
0	9A, Gopi Mondal Lane, Kolkata-700002	22.56263	88.36304

Now it's time to populate the dataframe using the rest of the office addresses. We will loop over the Opencage API call, seperate out the lat long values and append them to the latlong_dataframe.

```
latlong_dataframe = pd.DataFrame(columns = my_columns)
for address in df_kolkata['Office Address']:
    api_url = f'https://api.opencagedata.com/geocode/v1/json?q={address}&key=c0ca14c914034184
    data = requests.get(api_url).json()
    latlong_dataframe = latlong_dataframe.append(
         pd.Series(
         [
             address,
             data['results'][0]['geometry']['lat'],
             data['results'][0]['geometry']['lng']
         ],
         index = my_columns),
    ignore_index = True
```

```
    )
```

```
latlong_dataframe.head()
```

	Office Address	Latitude	Longitude
0	9A, Gopi Mondal Lane, Kolkata-700002	22.562630	88.363040
1	10, Rani Debendra Bala Rd Rishi, Paikpara Kolk...	22.562630	88.363040
2	Seth Lane, Kolkata-50	22.586374	88.353731
3	9A, Raja Manindra Road, Kolkata-37	22.612521	88.383454
4	67B, Khelat Babu Lane, Kolkata-37	22.562630	88.363040

And finally we merge the two dataframes into one on the column "Office Address".

```
df_kolkata = pd.merge(df_kolkata, latlong_dataframe, on="Office Address")
```

```
df_kolkata.head(20)
```

	Ward No.	Councillors	Office Address	Latitude	Longitude
0	1	SMT. SITA JAISWARA	9A, Gopi Mondal Lane, Kolkata-700002	22.562630	88.363040

```
df_kolkata.drop(index=16, inplace = True)
```

Now our updated dataframe contains the Ward numbers, Office addresses, latitudes and longitudes.

Our next task is to collect nearby venues data from Foursquare API. To call a get request we must obtain the latitude and longitude values of the center point of kolkata which is Park Street. We will use the GeoPy library for this task.

```
address = 'Park Street, Kolkata, India'

geolocator = Nominatim(user_agent="kolkata_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Kolkata are {}, {}.'.format(latitude, longitude))
```

```
The geograpical coordinate of Kolkata are 22.5551591, 88.3501171.
```

Now we write the foursquare API credentials. As this information is sensitive , i am going to hide this cell.

```
# @hidden_cell

CLIENT_ID = 'ZBVEI4BPRCOH0V4IAXIJCUNQFJPXGRGOOG2Q5X1LFFASTZUY' # your Foursquare ID
CLIENT_SECRET = 'WBXHFMP4ESAIKI3DFPQWVRGTKMAKZ3VHK5MLJJUMPUAHX3DK' # your Foursquare Secret
AUTH_CODE = 'C4IXNEHBHLPLX4QWHBNGNGZYYOU0L1SIY4XIR54OFARKEUWW'   # your authorization code
VERSION = '20180605' # Foursquare API version
LIMIT = 100    # A default Foursquare API limit value
```

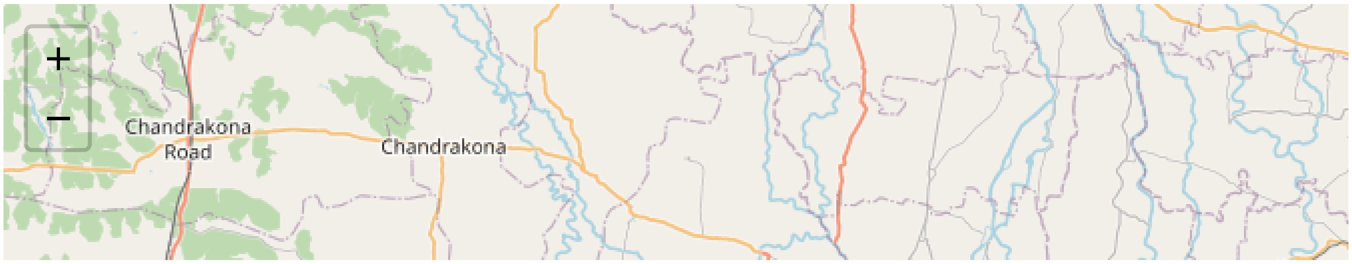
To visualise the data we have collected so far, we superimpose the wards on a follium map.

```
# create map of New York using latitude and longitude values
map_kolkata = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, ward in zip(df_kolkata['Latitude'], df_kolkata['Longitude'], df_kolkata['Ward N
    label = '{}'.format(ward)
```

```
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_kolkata)

map_kolkata
```



Now we write a function called 'getNearbyVenues' where we take the location values from our df_kolkata dataset, make an api call, recieve a json response of the nearby venues, parse the json specifically for venue categories and venue location data and then append all the data on a pandas dataframe.


```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v=
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['Ward',
                  'Ward Latitude',
                  'Ward Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']
```

```
    return(nearby_venues)
```

Now we call the function and create a dataframe called kolkata_venues.

```
# type your answer here
kolkata_venues = getNearbyVenues(names=df_kolkata['Ward No.'],
                                   latitudes=df_kolkata['Latitude'],
                                   longitudes=df_kolkata['Longitude']
                                  )
```

```
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
```

```
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
```

We have now reached the end of our data collection stage. In the final dataframe we have ward numbers, ward location data, venues, venue categories, and venue location data. We have all the features we need to perform exploratory data analysis and feed the data into our machine learning model.

```
kolkata_venues.shape
```

```
(720, 7)
```

```
kolkata_venues.head(10)
```

	Ward	Ward Latitude	Ward Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	1	22.56263	88.36304	Candid Photographer	22.563947	88.362629	Photography Studio
1	1	22.56263	88.36304	Raja Subodh Mallick Square	22.563610	88.359668	Park
2	1	22.56263	88.36304	Georgian Inn	22.559555	88.361805	Hotel
3	1	22.56263	88.36304	Hind INOX	22.564914	88.359152	Multiplex
4	1	22.56263	88.36304	Santosh Mitra Square	22.566008	88.365719	Park
5	1	22.56263	88.36304	Entally Market	22.559952	88.366698	Market
6	2	22.56263	88.36304	Candid Photographer	22.563947	88.362629	Photography Studio

Step 4: Data Science Methodology

The data science methodology consists of many processes which include Exploratory Data analysis, data preparation, model building, model deployment, model development etc. We know that data science methodology is an itterative process and it is key that we should keep repeating the steps untill we get desired results.

Exploratory Data Analysis

We will use the groupy method to get a scenario of number of venues in each ward.

```
kolkata_venues.groupby('Ward').count()
```

Ward	Ward Latitude	Ward Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
1	6	6	6	6	6	6
10	5	5	5	5	5	5
100	6	6	6	6	6	6
101	16	16	16	16	16	16
102	4	4	4	4	4	4
103	1	1	1	1	1	1
104	6	6	6	6	6	6
105	2	2	2	2	2	2
106	5	5	5	5	5	5
107	5	5	5	5	5	5
108	1	1	1	1	1	1
109	4	4	4	4	4	4
11	6	6	6	6	6	6
110	2	2	2	2	2	2
111	6	6	6	6	6	6
112	6	6	6	6	6	6
113	6	6	6	6	6	6
114	3	3	3	3	3	3
115	6	6	6	6	6	6
116	6	6	6	6	6	6
117	6	6	6	6	6	6
118	6	6	6	6	6	6
119	6	6	6	6	6	6
12	6	6	6	6	6	6
120	6	6	6	6	6	6
121	4	4	4	4	4	4
122	5	5	5	5	5	5
123	6	6	6	6	6	6
124	6	6	6	6	6	6

125	6	6	6	6	6	6
126	6	6	6	6	6	6
127	6	6	6	6	6	6
128	1	1	1	1	1	1
129	6	6	6	6	6	6
130	4	4	4	4	4	4
131	4	4	4	4	4	4
132	6	6	6	6	6	6
133	6	6	6	6	6	6
134	1	1	1	1	1	1
136	6	6	6	6	6	6
137	6	6	6	6	6	6
138	6	6	6	6	6	6
139	6	6	6	6	6	6
14	3	3	3	3	3	3
140	6	6	6	6	6	6
141	2	2	2	2	2	2
142	6	6	6	6	6	6
143	4	4	4	4	4	4
144	4	4	4	4	4	4
15	6	6	6	6	6	6
16	4	4	4	4	4	4
18	16	16	16	16	16	16
19	1	1	1	1	1	1
2	6	6	6	6	6	6
20	5	5	5	5	5	5
21	6	6	6	6	6	6
22	6	6	6	6	6	6
23	6	6	6	6	6	6
24	6	6	6	6	6	6
25	6	6	6	6	6	6

26	4	4	4	4	4	4
28	3	3	3	3	3	3
29	6	6	6	6	6	6
3	5	5	5	5	5	5
30	6	6	6	6	6	6
31	6	6	6	6	6	6
32	6	6	6	6	6	6
33	4	4	4	4	4	4
34	3	3	3	3	3	3
35	4	4	4	4	4	4
36	10	10	10	10	10	10
37	6	6	6	6	6	6
38	6	6	6	6	6	6
39	16	16	16	16	16	16
4	5	5	5	5	5	5
40	1	1	1	1	1	1
41	6	6	6	6	6	6
42	8	8	8	8	8	8
43	6	6	6	6	6	6
44	6	6	6	6	6	6
45	6	6	6	6	6	6
46	6	6	6	6	6	6
47	6	6	6	6	6	6
48	4	4	4	4	4	4
49	6	6	6	6	6	6
5	6	6	6	6	6	6
50	6	6	6	6	6	6
51	6	6	6	6	6	6
52	6	6	6	6	6	6
53	6	6	6	6	6	6
54	6	6	6	6	6	6
55	6	6	6	6	6	6

56	4	4	4	4	4	4
57	6	6	6	6	6	6
58	6	6	6	6	6	6
6	6	6	6	6	6	6
61	6	6	6	6	6	6
62	5	5	5	5	5	5
63	5	5	5	5	5	5
64	6	6	6	6	6	6
65	6	6	6	6	6	6
66	4	4	4	4	4	4
67	1	1	1	1	1	1
68	9	9	9	9	9	9
69	5	5	5	5	5	5
7	6	6	6	6	6	6
70	11	11	11	11	11	11
74	6	6	6	6	6	6

The one hot encoding method is used to convert categorical variables into binary which better suited to be fitted into our machine learning model.

```
# one hot encoding
kolkata_onehot = pd.get_dummies(kolkata_venues[['Venue Category']], prefix="", prefix_sep="")

# add Ward column back to dataframe
kolkata_onehot['Ward'] = kolkata_venues['Ward']

# move Ward column to the first column
fixed_columns = [kolkata_onehot.columns[-1]] + list(kolkata_onehot.columns[:-1])
kolkata_onehot = kolkata_onehot[fixed_columns]

kolkata_onehot.head()
```

	Ward	ATM	American Restaurant	Art Museum	Asian Restaurant	Athletics & Sports	Awadhi Restaurant	Bakery	Bank	Be Resta
2	1	0	0	0	0	0	0	0	0	

Now let's group the wards together and take the mean of the binary values. This will help us understand which venues are more frequent in their respective wards and name the new dataframe kolkata_grouped.

```
kolkata_grouped = kolkata_onehot.groupby('Ward').mean().reset_index()
kolkata_grouped
```

	Ward	ATM	American Restaurant	Art Museum	Asian Restaurant	Athletics & Sports	Awadhi Restaurant	Bakery	
0	1	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
1	10	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
2	100	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
3	101	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.125000	C
4	102	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
5	103	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
6	104	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
7	105	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
8	106	0.200000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
9	107	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
10	108	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
11	109	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
12	11	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
13	110	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
14	111	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
15	112	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
16	113	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
17	114	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
18	115	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
19	116	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
20	117	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
21	118	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
22	119	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
23	12	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
24	120	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
25	121	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
26	122	0.000000	0.000000	0.200000	0.00	0.0	0.00	0.000000	C
27	123	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
28	124	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C

29	125	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
30	126	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
31	127	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
32	128	0.000000	0.000000	0.000000	0.00	1.0	0.00	0.000000	
33	129	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
34	130	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
35	131	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
36	132	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
37	133	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
38	134	1.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
39	136	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
40	137	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
41	138	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
42	139	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
43	14	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
44	140	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
45	141	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
46	142	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
47	143	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.250000	
48	144	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.250000	
49	15	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.166667	
50	16	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.250000	
51	18	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.125000	
52	19	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
53	2	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
54	20	0.000000	0.000000	0.200000	0.00	0.0	0.00	0.000000	
55	21	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
56	22	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
57	23	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
58	24	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	
59	25	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	

60	26	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
61	28	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
62	29	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
63	3	0.600000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
64	30	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
65	31	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
66	32	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
67	33	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
68	34	0.333333	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
69	35	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
70	36	0.100000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
71	37	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
72	38	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
73	39	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.125000	C
74	4	0.200000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
75	40	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
76	41	0.000000	0.000000	0.166667	0.00	0.0	0.00	0.000000	C
77	42	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
78	43	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
79	44	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
80	45	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
81	46	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
82	47	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
83	48	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
84	49	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
85	5	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
86	50	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
87	51	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
88	52	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
89	53	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
90	54	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
91	55	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C

92	56	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
93	57	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
94	58	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
95	6	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
96	61	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
97	62	0.000000	0.000000	0.000000	0.00	0.0	0.20	0.000000	C
98	63	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
99	64	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
100	65	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
101	66	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
102	67	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
103	68	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
104	69	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C
105	7	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C

```
kolkata_grouped.shape
```

```
(135, 80)
```

108	73	0.000000	0.000000	0.000000	0.00	0.0	0.00	0.000000	C

```
kolkata_grouped['Ward'] = kolkata_grouped['Ward'].astype(str)
kolkata_grouped.dtypes
```

```
Dhaba                                        float64
Electronics Store                            float64
Fast Food Restaurant                         float64
Flea Market                                  float64
Furniture / Home Store                       float64
Grocery Store                                float64
Gym                                          float64
Gym / Fitness Center                         float64
Harbor / Marina                              float64
Historic Site                                float64
History Museum                               float64
Hostel                                       float64
Hotel                                        float64
IT Services                                  float64
Ice Cream Shop                               float64
Indian Restaurant                            float64
Indian Sweet Shop                            float64
Indie Movie Theater                          float64
Insurance Office                             float64
Italian Restaurant                           float64
Jewelry Store                                float64
Juice Bar                                    float64
Kerala Restaurant                            float64
Lake                                         float64
```

```
Lake                                         float64
Lounge                                       float64
Market                                       float64
Men's Store                                  float64
Metro Station                                float64
Mobile Phone Shop                            float64
Movie Theater                                float64
Mughlai Restaurant                           float64
Multiplex                                    float64
Music Venue                                  float64
Nightclub                                    float64
Park                                         float64
Performing Arts Venue                        float64
Pharmacy                                     float64
Photography Studio                           float64
Pier                                         float64
Pizza Place                                  float64
Playground                                   float64
Plaza                                        float64
Pool                                         float64
Residential Building (Apartment / Condo)     float64
Restaurant                                   float64
River                                        float64
Sausage Shop                                 float64
Shipping Store                               float64
Shoe Store                                   float64
Shopping Mall                                float64
Snack Place                                  float64
Spa                                          float64
Supermarket                                  float64
Tea Room                                     float64
Train Station                                float64
Tram Station                                 float64
Vegetarian / Vegan Restaurant                float64
Women's Store                                float64
dtype: object
```

To use the clustering alogrithm we need to perform exploratory data analysis on our dataset. The algorithm will cluster the wards according to the frequency of the venues occuring. That's why we now create a function which will list all the wards with their respective top 5 venues.

```
num_top_venues = 5

for wards in kolkata_grouped['Ward']:
    print("----"+wards+"----")
    temp = kolkata_grouped[kolkata_grouped['Ward'] == wards].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venue
    print('\n')
```

```
2              Market  0.17
```

```
3               Hotel  0.17
4           Multiplex  0.17


----93----
                venue  freq
0         Pizza Place  0.25
1     Asian Restaurant  0.25
2            Tea Room  0.25
3       Shopping Mall  0.25
4   Kerala Restaurant  0.00


----94----
                venue  freq
0                Park  0.33
1  Photography Studio  0.17
2              Market  0.17
3               Hotel  0.17
4           Multiplex  0.17


----95----
                venue  freq
0    Department Store  0.17
1   Convenience Store  0.17
2                Park  0.17
3                 Spa  0.17
4   Mobile Phone Shop  0.17


----97----
                venue  freq
0                Park  0.33
1  Photography Studio  0.17
2              Market  0.17
3               Hotel  0.17
4           Multiplex  0.17



----98----
                   venue  freq
0                    ATM   0.5
1          Metro Station   0.5
2          Movie Theater   0.0
3               Pharmacy   0.0
4  Performing Arts Venue   0.0


----99----
                   venue  freq
0            Pizza Place  0.25
1  Furniture / Home Store  0.25
2             Supermarket  0.25
3           Metro Station  0.25
4                     ATM  0.00
```

Now we write a function to sort the venues in descending order.

```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

Now let's use the function and show the top 10 venues in a pandas dataframe.

```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Ward']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
Wards_venues_sorted = pd.DataFrame(columns=columns)
Wards_venues_sorted['Ward'] = kolkata_grouped['Ward']

for ind in np.arange(kolkata_grouped.shape[0]):
    Wards_venues_sorted.iloc[ind, 1:] = return_most_common_venues(kolkata_grouped.iloc[ind, :

Wards_venues_sorted.head()
```

	Ward	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
0	1	Park	Photography Studio	Hotel	Multiplex	Market	Grocery Store	Dhaba	Electronics Store
1	10	Indian Sweet Shop	Fast Food Restaurant	Bank	Market	Metro Station	Gym / Fitness Center	Dhaba	Electronics Store
2	100	Park	Photography Studio	Hotel	Multiplex	Market	Grocery Store	Dhaba	Electronics Store
3	101	Café	Bakery	Ice Cream	Performing Arts Venue	Indian Restaurant	Pizza Place	Restaurant	Nightclub

Clustering Neighborhoods

Now we enter the stage where we use our machine learning algorithm which is K-means clustering.

K-means will group the wards into 5 groups depending on the likeness of the wards with respect to each other.

```
# set number of clusters
kclusters = 5

kolkata_grouped_clustering = kolkata_grouped.drop('Ward', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(kolkata_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:100]
```

```
array([1, 2, 1, 4, 2, 0, 1, 2, 2, 2, 4, 4, 1, 2, 1, 1, 1, 4, 1, 1, 1, 1,
       1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 1, 2, 4, 1, 1, 3, 1, 1, 1, 1, 2,
       1, 2, 1, 4, 4, 2, 2, 4, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 3, 1, 1,
       1, 2, 2, 2, 2, 1, 1, 4, 2, 2, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1,
       1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1], dtype=int32)
```

Now we add the 'Cluster Labels' to the dataframe by creating a new column and merging it with the 'Ward_venues_sorted' and name the new dataframe 'kolkata_merged'.

```
Wards_venues_sorted.dtypes
```

```
Ward                      object
1st Most Common Venue     object
2nd Most Common Venue     object
3rd Most Common Venue     object
4th Most Common Venue     object
5th Most Common Venue     object
6th Most Common Venue     object
7th Most Common Venue     object
8th Most Common Venue     object
9th Most Common Venue     object
10th Most Common Venue    object
dtype: object
```

Which ward belongs to which group will be determined by the cluster labels.

```
# add clustering labels
Wards_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)
```

```
Wards_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)
```

```
kolkata_merged = df_kolkata
kolkata_merged.head(10)
```

	Ward No.	Councillors	Office Address	Latitude	Longitude
0	1	SMT. SITA JAISWARA	9A, Gopi Mondal Lane, Kolkata-700002	22.562630	88.363040
1	2	SMT. PUSPALI SINHA	10, Rani Debendra Bala Rd Rishi, Paikpara Kolk...	22.562630	88.363040
2	3	DR SANTANU SEN	Seth Lane, Kolkata-50	22.586374	88.353731
3	4	SHRI GOUTAM HALDAR	9A, Raja Manindra Road, Kolkata-37	22.612521	88.383454
4	5	SHRI TARUN SAHA	67B, Khelat Babu Lane, Kolkata-37	22.562630	88.363040
5	6	SMT. SUMAN SINGH	3/1B ,Turner Road, Kolkata-02	22.562630	88.363040
6	7	SHRI BAPI GHOSH	47A, Bagbazar Street, Kolkata-700003	22.603919	88.366481
7	8	SHRI PARTHA MITRA	3/1E Raja Raj Ballav Street, Kolkata-03	22.562630	88.363040

Double-click (or enter) to edit

```
df_kolkata['Ward No.'] = df_kolkata['Ward No.'].astype(str)
```

```
df_kolkata.dtypes
```

```
Ward No.           object
Councillors        object
Office Address     object
Latitude          float64
Longitude         float64
dtype: object
```

```
kolkata_merged = df_kolkata

# merge kolkata_grouped with kolkata_data to add latitude/longitude for each neighborhood
kolkata_merged = kolkata_merged.join(Wards_venues_sorted.set_index('Ward'), on='Ward No.')
```

Now we prepare our final dataframe kolkata_merged, which will contain our previous dataset df_kolkata (ward numbers, Office Address, latitude, longitude) , cluster labels and the most common venues.

```
kolkata_merged = kolkata_merged.fillna(0)
```

```
kolkata_merged['Cluster Labels'] = kolkata_merged['Cluster Labels'].astype(int)
```

```
kolkata_merged.dtypes
```

```
Ward No.                  object
Councillors               object
Office Address            object
Latitude                 float64
Longitude                float64
Cluster Labels             int64
1st Most Common Venue     object
2nd Most Common Venue     object
3rd Most Common Venue     object
4th Most Common Venue     object
5th Most Common Venue     object
6th Most Common Venue     object
7th Most Common Venue     object
8th Most Common Venue     object
9th Most Common Venue     object
10th Most Common Venue    object
dtype: object
```

```
kolkata_merged.head()
```

	Ward No.	Councillors	Office Address	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3r Mos Comm Venu
0	1	SMT. SITA JAISWARA	9A, Gopi Mondal Lane, Kolkata-700002	22.562630	88.363040	1	Park	Photography Studio	Hot
1	2	SMT. PUSPALI SINHA	10, Rani Debendra Bala Rd Rishi, Paikpara Kolk...	22.562630	88.363040	1	Park	Photography Studio	Hot
2	3	DR SANTANU SEN	Seth Lane, Kolkata-	22.586374	88.353731	3	ATM	Snack Place	Mark

Now we will use follium to visualise the clusters superimposed on terrestrial map. We will select different colours for different clusters which will help us visualise.

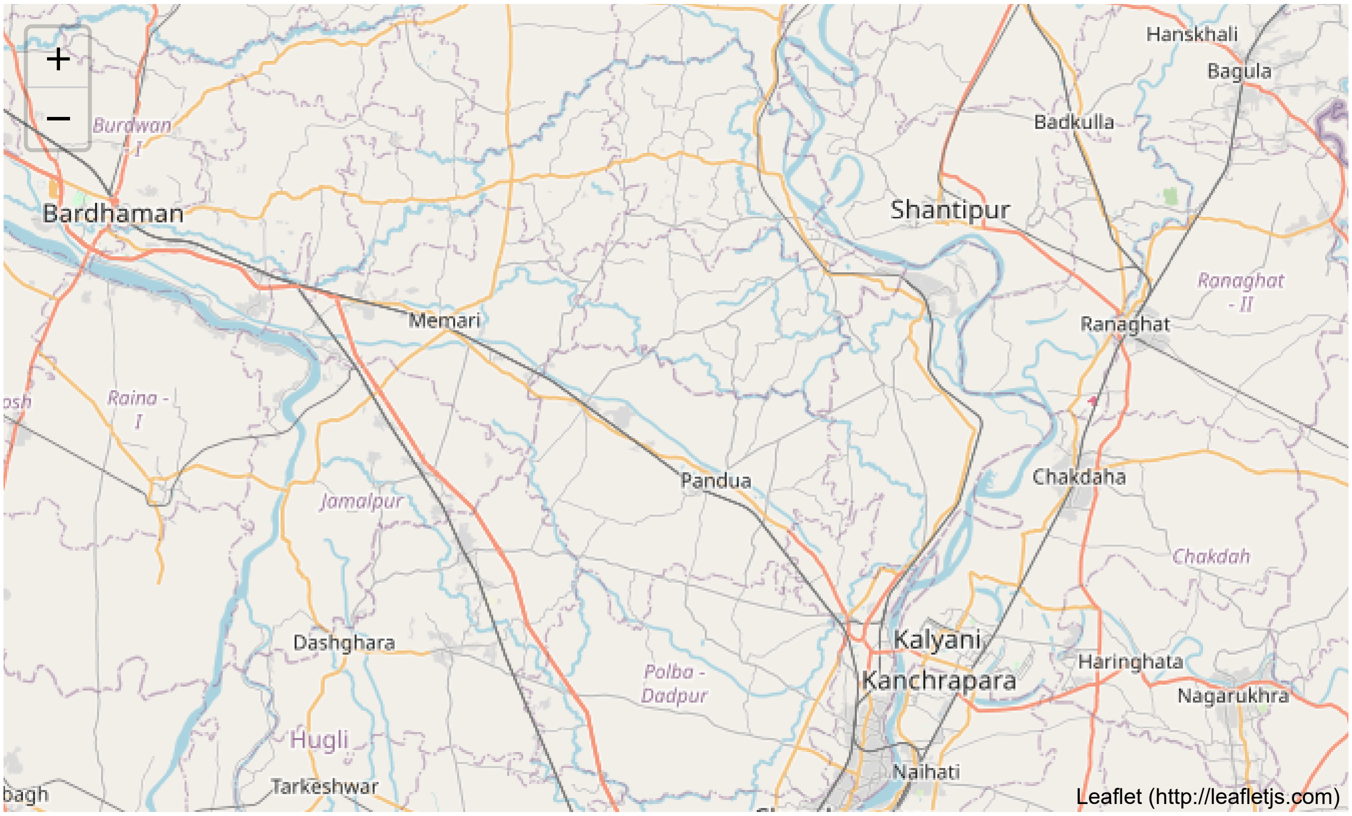
Now let's visualize the clusters

```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(kolkata_merged['Latitude'], kolkata_merged['Longitude'], ko
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```



Step 5: Generating insights

Examine Clusters

Cluster 1

This cluster mainly contains the wards for which Foursqaure API has no information. So we can ignore this cluster for now.

```
kolkata_merged.loc[kolkata_merged['Cluster Labels'] == 0, kolkata_merged.columns[[1] + list(r
```

	Councillors	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7t
26	SMT. MINAKSHI GUPTA	0	0	0	0	0	0	0	
58	SMT. JALY BOSE	0	0	0	0	0	0	0	

Cluster 2

```
kolkata_merged.loc[kolkata_merged['Cluster Labels'] == 1, kolkata_merged.columns[[1] + list(r
```

	Councillors	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th M Com Ve
0	SMT. SITA JAISWARA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
1	SMT. PUSPALI SINHA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
4	SHRI TARUN SAHA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
5	SMT. SUMAN SINGH	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
7	SHRI PARTHA MITRA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
8	SMT. MITALI SAHA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
10	SHRI ATIN GHOSH	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
11	SMT. PRANATI BHATTACHARJEE	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
19	SHRI VIJAY UPADHYAY	1	Park	Metro Station	Art Museum	History Museum	Women's Store	Electro S
20	SMT. SUJATA SAHA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
21	SMT. MEENA DEVI PUROHIT	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
22	SHRI VIJAY OJHA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
23	SMT. ELLORA SAHA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
24	SMT. SMITA BAKSHI	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
25	SHRI TARAK NATH CHATTOPADHYAY	1	Park	Metro Station	Pizza Place	Women's Store	Dhaba	Electro S
28	SHRI PRAKASH UPADHYAY	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
29	SMT. PAPIYA GHOSH (BISWAS)	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
30	SMT.SUNANDA GUHA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S

31	SHRI SANTI RANJAN KUNDU	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
36	SMT. SOMA CHAUDHURI	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
37	SMT. SADHANA BOSE	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
42	SMT. SHAGUFTA PARVEEN	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
43	REHANA KHATOON	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
44	SHRI SANTOSH KUMAR PATHAK	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
45	SHRI GOPAL CHANDRA SAHA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
46	SMT. SUMAN SINGH	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
48	SMT. APARAJITA DASGUPTA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
50	SMT. SANCHITA MONDAL	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
51	SHRI SANDIPAN SAHA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
52	SMT. INDRANI SAHA BANERJEE	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
53	AMIRUDDIN (BOBBY)	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
54	SHRI ARUN KUMAR DAS	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
56	SHRI JIBAN SAHA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
57	SHRI SWAPAN SAMADDAR	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
60	MANZAR IQBAL	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
62	SMT. SUSMITA BHATTACHARYA (CHATTERJEE)	1	Park	Photography Studio	Hotel	Market	Gym	Dh
63	IQBAL AHMED	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
64	SMT. NIBEDITA SHARMA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S

73	SMT. DEBALINA BISWAS	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
74	BELQUIS BEGUM	1	Park	Shoe Store	Women's Store	Gym	Dhaba	Electro S
75	SHRI SASTI DAS	1	Park	Awadhi Restaurant	Shoe Store	Market	Gym / Fitness Center	Dh
76	SHAMIMA REHAN KHAN	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
79	MD. ANWAR KHAN	1	Park	Tram Station	Awadhi Restaurant	Market	Gym / Fitness Center	Dh
82	SMT. MANJUSREE MAJUMDAR	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
84	SHRI DEBASISH KUMAR	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
87	SMT. MALA ROY	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
91	SMT. MADHUCHHANDA DEB	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
93	SMT. ARCHANA SEN GUPTA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
96	SMT. MITALI BANERJEE	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
99	SMT. SUSMITA DAM	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
103	SHRI TARAKESWAR CHAKRABORTY	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
110	SHRI CHAYAN BHATTACHARYA	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
111	SMT. ANITA KAR MAJUMDAR	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S
112	SHRI GOPAL RAY	1	Park	Photography Studio	Hotel	Multiplex	Market	Gro S

We can see that most of the wards fall under the 2nd cluster. This cluster is the most important because we see that the most common venues are the parks. In the beginning, at the formulation of

the business plan, we decided to avoid areas with the most parks. So we can avoid these wards till
DASGUPTA Studio S

Cluster 3

```
kolkata_merged.loc[kolkata_merged['Cluster Labels'] == 2, kolkata_merged.columns[[1] + list(r
```

	Councillors	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Mos Commo Venu
3	SHRI GOUTAM HALDAR	2	ATM	Pharmacy	Park	Music Venue	Electronic Stor
6	SHRI BAPI GHOSH	2	Clothing Store	River	Pier	Chinese Restaurant	Bar
9	SMT. KARUNA SENGUPTA	2	Indian Sweet Shop	Fast Food Restaurant	Bank	Market	Metr Static
13	SHRI AMAL CHAKRABORTY	2	Gym	Clothing Store	Historic Site	Women's Store	Dhab
14	SMT. SHUKLA BHORE	2	Clothing Store	Chinese Restaurant	Café	Bakery	Po
15	SHRI SADHAN SAHA	2	Vegetarian / Vegan Restaurant	Indian Restaurant	Plaza	Bakery	Gyı
18	SMT. SIKHA SAHA	2	IT Services	Women's Store	Gym / Fitness Center	Dhaba	Electronic Stor
27	IQBAL AHMED	2	Women's Store	Chinese Restaurant	Department Store	Ice Cream Shop	Grocer Stor
32	SHRI PABITRA BISWAS	2	Chinese Restaurant	Restaurant	Gym / Fitness Center	Pharmacy	Women Stor
33	SMT. ALOANANDA DAS	2	ATM	Pharmacy	Restaurant	Department Store	Dhab
34	SHRI ASHUTOSH DAS	2	Chinese Restaurant	Restaurant	Gym / Fitness Center	Pharmacy	Women Stor
35	SHRI RAJESH KHANNA	2	ATM	Bookstore	Café	Plaza	Po
39	SMT. SWAPNA DAS	2	Men's Store	Women's Store	Gym / Fitness Center	Dhaba	Electronic Stor
40	SMT. REITA CHOWDHURY	2	Park	Metro Station	Art Museum	Hotel	Caf
41	SMT. SUNITA JHAWAR	2	Metro Station	Indian Sweet Shop	Indie Movie Theater	Mughlai Restaurant	Bar

47	SHRI SATYENDRA NATH DEY (BULU DA)	2	Juice Bar	Fast Food Restaurant	Plaza	Breakfast Spot	Women Stor
49	SMT. MOUSUMI DEY	2	Insurance Office	Café	Jewelry Store	Park	Departme Stor
55	SMT. DIPALI DAS	2	Indian Sweet Shop	Chinese Restaurant	Hotel	Bus Station	Gy
61	SANA AHMED	2	Hotel	Awadhi Restaurant	Hostel	Campground	Mughl Restaura
65	FAIZ AHMED KHAN	2	Photography Studio	Lake	Residential Building (Apartment / Condo)	Boutique	Gy
66	SHRI BIJAN LAL MUKHERJEE	2	Pharmacy	Indie Movie Theater	Department Store	Dhaba	Electronic Stor

The 3rd cluster is populated with clothing stores and restaurants.

67	SUDARSHANA	2	Bengali	Dhaba	/ Vegan	Indian Sweet	Chines
69		2				Multiplex	

Cluster 4

```
kolkata_merged.loc[kolkata_merged['Cluster Labels'] == 3, kolkata_merged.columns[[1] + list(r
```

	Councillors	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
2	DR SANTANU SEN	3	ATM	Snack Place	Market	Gym / Fitness Center	Dhaba	Electronics Store
97	SHRI MRITYUNJOY CHAKRABORTY	3	ATM	Metro Station	Indian Restaurant	Ice Cream Shop	Dhaba	Electronics Store
80	BISWAS	2	Snack Place	Store	Store	Fitness	Dhab

This cluster is helpful as the common venues are fitness centers which will attract fitness enthusiasts and metro stations which will ensure ease of transportation.

Cluster 5

	CHATTERJEE		Restaurant	Restaurant	Restaurant		Cent

```
kolkata_merged.loc[kolkata_merged['Cluster Labels'] == 4, kolkata_merged.columns[[1] + list(r
```

	Councillors	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	
17	SHRI SUNANDA SARKAR	4	Café	Bakery	Ice Cream Shop	Performing Arts Venue	Indian Restaurant	
38	MD. JASIMUDDIN	4	Café	Bakery	Ice Cream Shop	Performing Arts Venue	Indian Restaurant	
68	SHRI SUKDEV CHAKRABARTY	4	Café	Ice Cream Shop	Chinese Restaurant	Women's Store	Gym	E
85	SMT. TISHTA BISWAS (DAS)	4	Café	Indian Restaurant	Ice Cream Shop	Hotel	Lounge	
100	SHRI BAPPADITYA DASGUPTA	4	Café	Bakery	Ice Cream Shop	Performing Arts Venue	Indian Restaurant	
107	SHRI SHYAMAL BANERJEE	4	Café	Women's Store	Cosmetics Shop	Dhaba	Electronics Store	F
108	SMT. ANANYA BANERJEE	4	Café	Clothing Store	Movie Theater	Women's Store	Gym	E
113	SHRI BISWAJIT MANDAL	4	Café	Lounge	Movie Theater	Women's Store	Gym	E

The most common venues of this cluster is coffee shops and other restaurants. To think strategically, these areas have footfall and will positively impact our business. As adults and young adults often visit these areas, these locations might be beneficial for our business.

Conclusion

So we are now at this end of this analysis. I have used all the knowledge i could gather from the IBM Professional Data Science Certificate courses. Feel we to comment where you think I can improve this project.

So Far, by my understanding, clusters 5 and 4 are the most important for artificial turfs. This locations will positively impact our business for the reasons I have provided earlier. For cluster 3 i am somewhat skeptical at the moment. I would very much appriciate feedback on how I can use this cluster to my advantage.