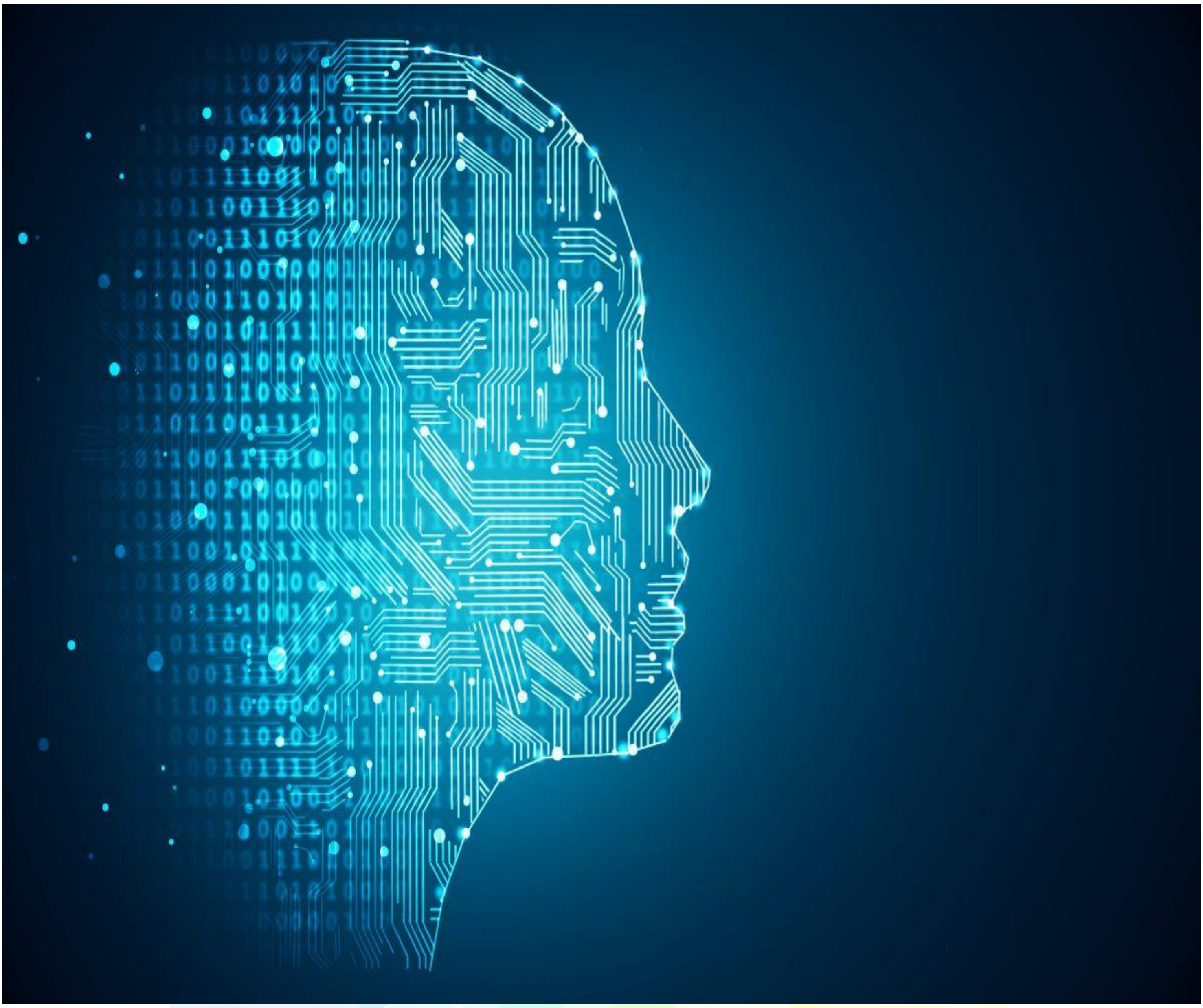
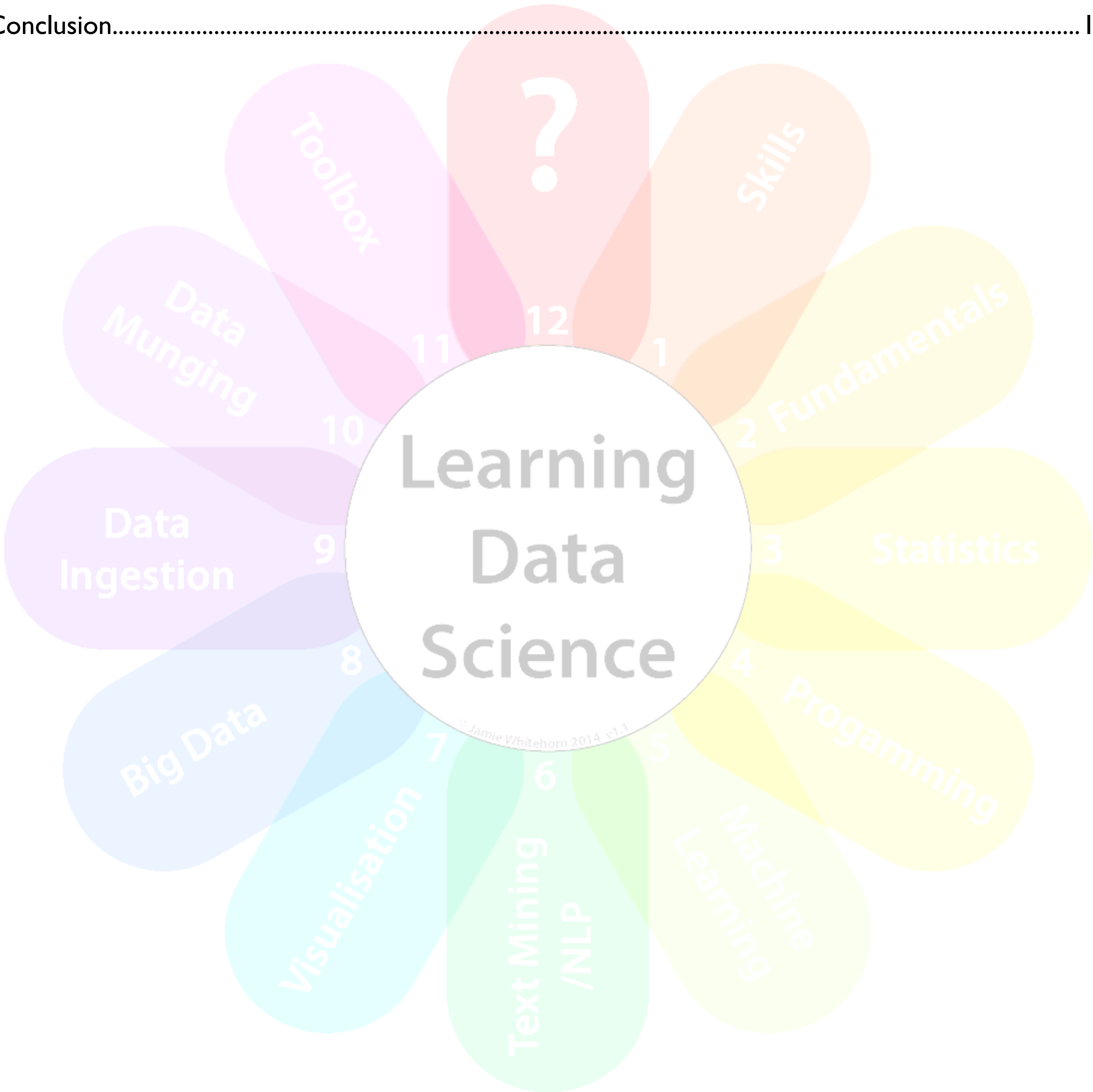


IBM Data Science Applied Capstone Project

Analyzing Johannesburg venues, By Nkululeko Nhlapo

Nkululeko Nhlapo 7/15/20 IBM Data Science

Table of Contents



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. 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.

Johannesburg is the largest single metropolitan contributor to the national economic product. National average growth in gross domestic product (GDP) has been 1,8% over the past 10 years, with Johannesburg marginally outpacing that growth with an average growth per annum of 2%. The city's contribution to the national economy is almost 16%, while it is 40% to the Gauteng province.

Per capita gross geographic product (GGP) in Johannesburg is R31 000. This compares to World Bank-designated middle-income countries whose average GGP per capitais R33 000.

Economical Stats have demonstrated Johannesburg City as one of the better capitalist city for any small business entrepreneur who wishes to venture his/her business in Johannesburg. Therefore this project is created to assist such new business to succeed.

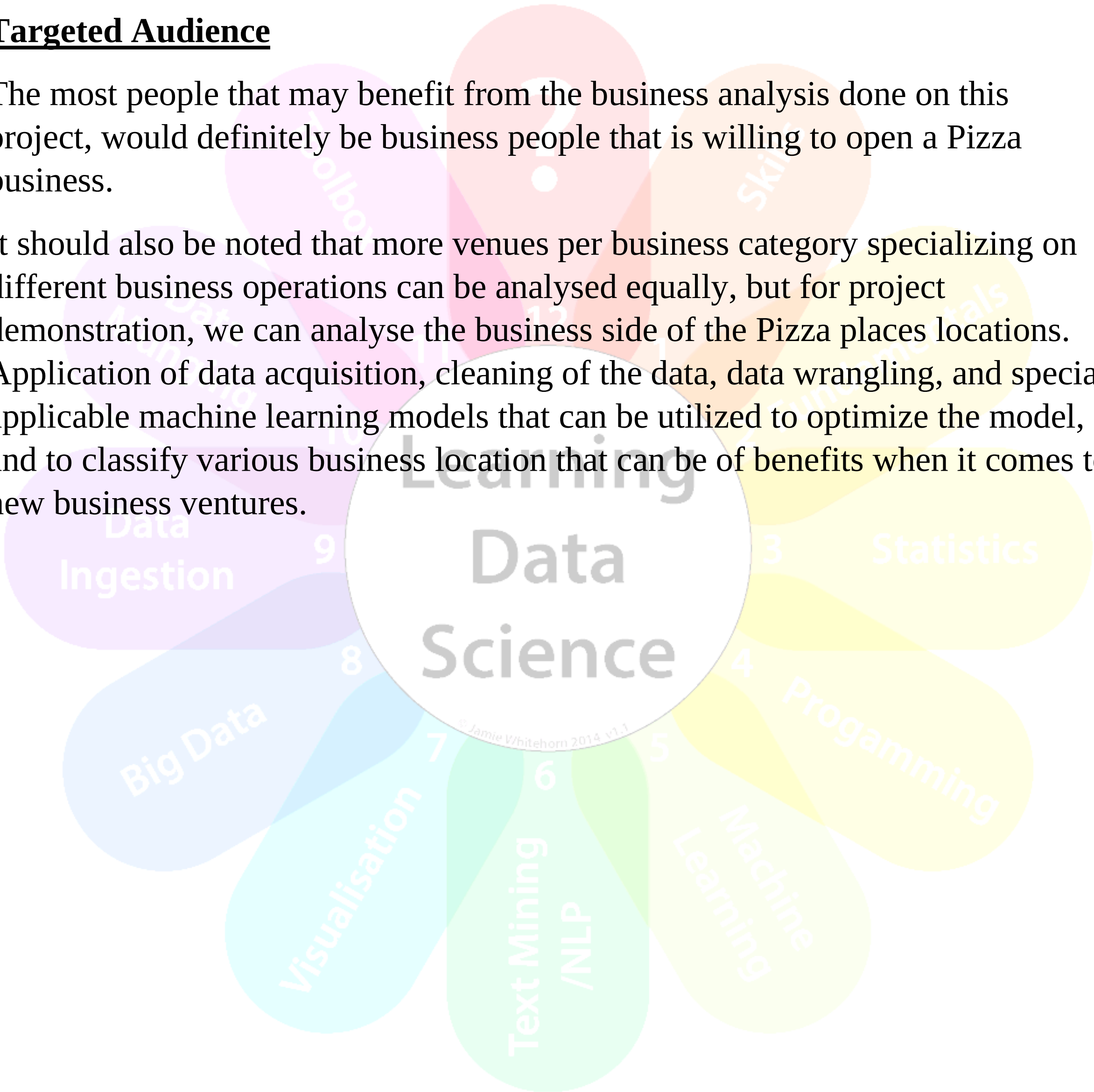
Business Problem

The objective of this capstone project is to analyse and select the best locations in the city of Johannesburg, in a Gauteng Province of South Africa, in order to assist business people that hope to open any type of business in the location of Johannesburg city.

<u>Targeted Audience</u>

The most people that may benefit from the business analysis done on this project, would definitely be business people that is willing to open a Pizza business.

It should also be noted that more venues per business category specializing on different business operations can be analysed equally, but for project demonstration, we can analyse the business side of the Pizza places locations. Application of data acquisition, cleaning of the data, data wrangling, and special applicable machine learning models that can be utilized to optimize the model, and to classify various business location that can be of benefits when it comes to new business ventures.



Data Tools

The data for this project has been retrieved and processed through multiple sources. It has to be noted that the sources of data are scarce for South Africa. Therefore, to do data acquisition I had to fill data manually from external sources:

Neighbourhoods

The internet seems to have less data either in a form of a list for listing all the Johannesburg suburbs, as result I visited a webpage: https://www.roomsforafrica.com/dest/south africa/gauteng/johannesburg.jsp?tab=3, to search for Johannesburg's suburbs name. The suburbs will be used an alternative word for Neighbourhoods.

For each suburbs the geographical location for each was needed. I used the following webpage to retrieve the geographical coordinates for each listed Johannesburg suburbs: https://www.gps-coordinates.net/.

- The latitude and longitude of the neighbourhoods are retrieved using **Google Maps Geocoding API**. The geometric location values are then stored into the initial data frame.
- The venue data for each suburb in a data frame will be retrieved using a **Fouraquare API** and creating another data frame to contain all the venues details regarding its number of visits as compared with other venues within certain radius, and so forth.

Data scraping using **BeautifulSoup** library is one of the pleasurable data acquisition methods that can ease some cumbersome task for data scraping, but lack thereof of data which forms as a vitality for data analysis enforced me to create a json file from scratch, using acquired data through Jupyter Notebooks.

Methodology

The following are procedural methods that were applied in the project in order to analyse Johannesburg:

Importing the Library

```
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)

!conda install -c conda-forge geopy --yes

!conda install -c conda-forge folium=0.5.0 --yes

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 k-means from clustering stage
from sklearn.cluster import KMeans
import folium # map rendering library

print('Libraries imported.')
```

Figure 1: Importing the libraries

Creating the json.

```
joburg_data = {
  "type": "FeatureCollection",
   "features" : [
      {
         "type" : "Feature",
         "geometry" : {
            "type" : "Point",
            "coordinates" : [
               28.0459,
               -26.20145
            ]
         },
         "properties" : {
            "name": "Brakpan",
            "borough": "Johannesburg"
         }
```

Figure 2: Creating the json

Creating the data frame

```
Joburg_Suburbs = joburg_data['features']
```

```
Joburg_Suburbs[0]
```

```
]: {'type': 'Feature',
    'geometry': {'type': 'Point', 'coordinates': [28.0459, -26.20145]},
    'properties': {'name': 'Brakpan', 'borough': 'Johannesburg'}}
```

```
# define the dataframe columns
column_names = ['Borough', 'Suburb Name', 'Latitude', 'Longitude']

# instantiate the dataframe
suburbs = pd.DataFrame(columns=column_names)
```

```
for data in Joburg_Suburbs:
    suburbs_category = suburb_name = data['properties']['borough']
    suburbs_name = data['properties']['name']

    suburbs_latlon = data['geometry']['coordinates']
    suburbs_lat = suburbs_latlon[1]
    suburbs_lon = suburbs_latlon[0]

    suburbs  = suburbs.append({'Borough': suburbs_category ,
                               'Suburb Name':  suburbs_name,
                               'Latitude': suburbs_lat,
                               'Longitude': suburbs_lon}, ignore_index=True)
```

Figure 3: creating the data frame list

Using Google Maps Geocoder API to locate Johannesburg

```
address = 'Johannesburg'

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

Figure 4: Using Geocoder

Using Follium: All cluster visualization are done with help of Folium which in turn generates a Leaflet map made using OpenStreetMap technology.

```
map_joburg = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, suburb in zip(suburbs['Latitude'], suburbs['Longitude'], suburbs['Suburb Name']):
    suburb = folium.Popup(suburb, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=suburb,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_joburg)

map_joburg
```

Figure 5: Using Follium

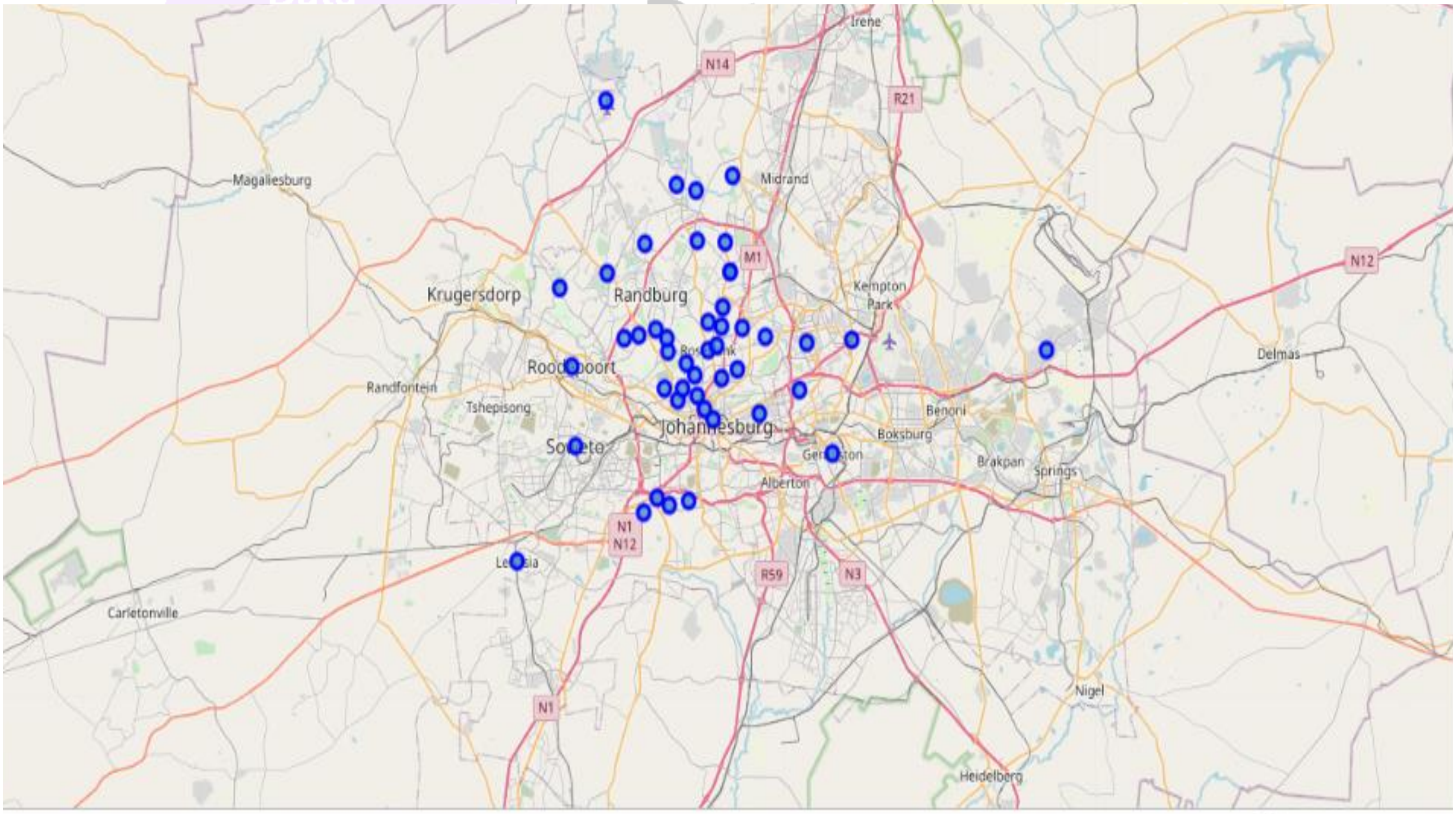


Figure 6: Resulted suburb point

Using of **Foursquare**

```
CLIENT_ID = 'IDSREVRMT1XCAZAX0MZKRUJHTGL35OCGRPVJ3X04JRTH3LQP' # your Foursquare ID
CLIENT_SECRET = 'SMMMLX3RG1YE3URNYCMULJW0NK55RQD2KS3RGMELRKE1CWI1' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

Figure 7: Logging to foursquare

Using one hot encoding

To process data by which categorical variables are converted into a form that could be provided to Machine Learning models/algorithms to do a better job in prediction. For the K-means Clustering Algorithm, all unique items under Venue Category are one-hot encoded.

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

# add neighborhood column back to dataframe
suburbs_onehot['Suburb'] = suburbs_venues['Suburb']

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

suburbs_onehot.head()
```

Figure 8: One-Hot encoding

Getting Top 10 Venues

```python
num_top_venues = 10

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

# create columns according to number of top venues
columns = ['Suburb']
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
suburbs_venues_sorted = pd.DataFrame(columns=columns)
suburbs_venues_sorted['Suburb'] = suburbs_grouped['Suburb']

for ind in np.arange(suburbs_grouped.shape[0]):
    suburbs_venues_sorted.iloc[ind, 1:] = return_most_common_venues(suburbs_grouped.iloc[ind, :], num_top_venues)

suburbs_venues_sorted.head()
```

Figure 9: Getting Top 10 Venues

Setting 5 clusters for 40 places

```python
# set number of clusters
kclusters = 5

suburbs_grouped_clustering = pizza_places.drop('Suburb', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:39]
```

Figure 10: Setting 5 clusters

The merging the table with the original data frame and mapping the clusters per geographical coordinates, using Follium.

```
# create map
map_suburbs = 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(suburbs_merged['Latitude'], suburbs_merged['Longitude'], suburbs_merged['Suburb'], suburbs_merged['Cluster Labels']):
    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_suburbs)

map_suburbs
```

Figure 11: Mapping the clusters

Results

After the use of the folium we can review the mapped clustered data as demonstrated below.

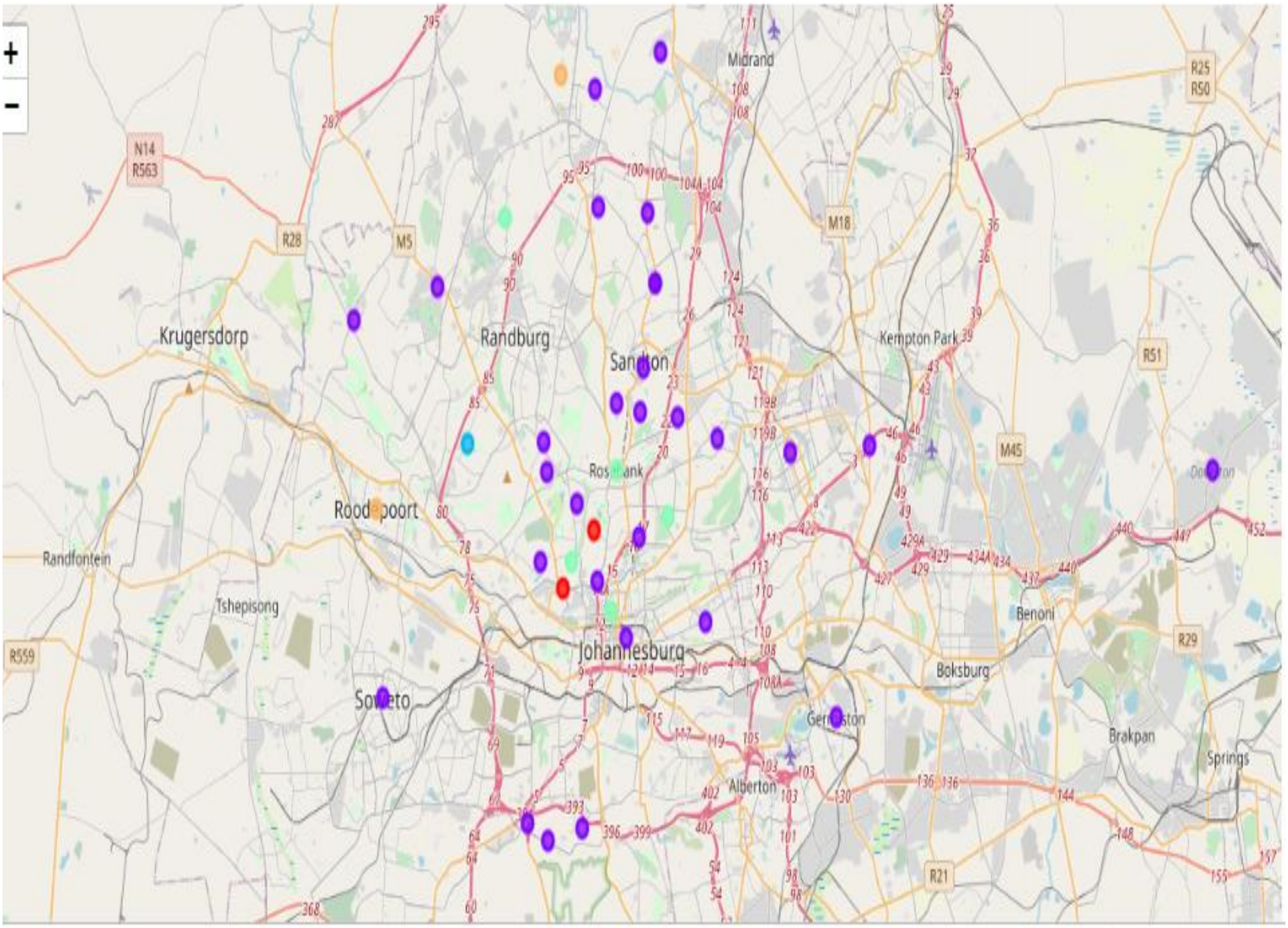


Figure 12: Cluster mapped

Discussion

```
map_suburbs.save('map_clusters.html')
```

```
suburbs_merged.loc[suburbs_merged['Cluster Labels'] == 0]
```

32]:

	Suburb	Pizza Place	Cluster Labels	Borough	Latitude	Longitude
0	Auckland Park	0.100000	0	Johannesburg	-26.184631	28.005838
29	Parkview	0.111111	0	Johannesburg	-26.164167	28.025278

Figure 13: Cluster 1

]:
```
suburbs_merged.loc[suburbs_merged['Cluster Labels'] == 1]
```

293]:

	Suburb	Pizza Place	Cluster Labels	Borough	Latitude	Longitude
38	Westdene	0.0	1	Johannesburg	-26.175000	27.990833
21	Lonehill	0.0	1	Johannesburg	-26.009722	28.026111
22	Lyndhurst	0.0	1	Johannesburg	-26.132102	28.104117
24	Mondeor	0.0	1	Johannesburg	-26.272500	27.996111
25	Morningside	0.0	1	Johannesburg	-26.078056	28.064444
25	Morningside	0.0	1	Johannesburg	-26.078056	28.064444
28	Parktown	0.0	1	Johannesburg	-26.181667	28.027778
30	Randburg	0.0	1	Johannesburg	-26.143841	27.995186
31	Rivonia	0.0	1	Johannesburg	-26.053333	28.059444
34	Ruimsig	0.0	1	Johannesburg	-26.090917	27.871933
35	Sandton	0.0	1	Johannesburg	-26.107567	28.056702
36	South Gate	0.0	1	Johannesburg	-26.266403	27.982587
37	Soweto	0.0	1	Johannesburg	-26.222778	27.890000
18	Lanseria	0.0	1	Johannesburg	-25.934450	27.924736
17	Kyalami	0.0	1	Johannesburg	-25.997024	28.067827
20	Linden	0.0	1	Johannesburg	-26.133398	27.993333

Figure 14: Cluster 2

```
suburbs_merged.loc[suburbs_merged['Cluster Labels'] == 2]
```

34]:

	Suburb	Pizza Place	Cluster Labels	Borough	Latitude	Longitude
7	Fairland	0.25	2	Johannesburg	-26.133611	27.944444
19	Lenasia	0.25	2	Johannesburg	-26.319631	27.824432

Figure 15: Cluster 3

```
suburbs_merged.loc[suburbs_merged['Cluster Labels'] == 3]
```

5]:

	Suburb	Pizza Place	Cluster Labels	Borough	Latitude	Longitude
26	North Riding	0.055556	3	Johannesburg	-26.054722	27.968333
27	Norwood	0.050000	3	Johannesburg	-26.158889	28.072500
23	Melville	0.043478	3	Johannesburg	-26.175163	28.010860
33	Rosebank	0.027027	3	Johannesburg	-26.142948	28.039749
1	Braamfontein	0.034483	3	Johannesburg	-26.192321	28.036198

Figure 16: Cluster 4

```
suburbs_merged.loc[suburbs_merged['Cluster Labels'] == 4]
```

]:

	Suburb	Pizza Place	Cluster Labels	Borough	Latitude	Longitude
32	Roodepoort	0.2	4	Johannesburg	-26.156389	27.885833
8	Fourways	0.2	4	Johannesburg	-26.005000	28.003889

Figure 17: Cluster 5

From the clusters as they can be seen, we can then conclude that the most suburbs are without Pizza Place, and as we can notice in cluster number 2. If the business owner hope to open a business in those, there will be certain criteria that should be considered, of which that can include the household income in those areas. I however much of data had acquired in order to conduct economic factors surrounding those areas

Conclusion

Data form as a vital role when it comes to data science, especially if one must leverage some open-source software to analyse a business problem. The data in South Africa still remains insufficient. However, with **Foursqaure** it was very simple to analyse venues per business category. It would be much appreciated to observe buying per each business category that sells, and even though we need to appreciate having sufficient data for top visits.