Coursera Capstone Project : Applied Data Science

1 Introduction

For the last decade, the United Kingdom’s grocery landscape has been dominated by the ‘big four’ supermarket chains: Tesco, Asda, Sainsbury’s and Morrisons. However, on the back of the economic recession, rising food prices and tightened belts, the market has been shaken by British consumers' search for value.

In 2015, IGD valued the grocery retail market at 178 billion British pounds, predicting this figure to rise annually up to 2021. Additional figures from the Office of National Statistics show that despite an increase in the value of grocery store sales, the volume of goods purchased by consumers shows negligible change.

Although rising food prices have not caused the quantity of goods purchased to fall, consumers seem more likely than ever to search for cheaper alternatives to the ‘big four.’ Discount supermarkets are enjoying a surge in popularity among food shoppers. According to figures from Kantar Worldpanel, all of the leading four supermarket brands have lost market share in the three months to August 2016.

Therefore the greater accessibility to these areas fueled by long term growth plans are necessary for the revival of these supermarkets along with greater development of the neighboring areas.

2 Business Problem

Consumer surveys indicate a shift in thinking among shoppers. According to recent evaluations, the number of consumers that use discounters is still increasing, while the number that never use them has decreased: they are chosen over supermarkets

because of their public perception as cheaper and, ironically, to avoid the complexity of over-promotion. Meanwhile, online grocery shopping could further revolutionize the market as e-commerce gains popularity among shoppers in the United Kingdom. Currently, online grocery sales in the United Kingdom take 6.9 percent of the global e-commerce market, and is thus the largest online grocery market in Europe. Thus in hope of the supermarkets to keep up their revenue , the main objective of this project is to find the best locations to situate the supermarkets so that it has greater accessibility of buyers and draws more attention along with keeping in mind the surrounding neighborhoods so that that area also indirectly thrives due to the supermarkets like food joints , movie-halls , etc.

3 Data

The data for this project has been retrieved and processed through multiple sources, giving careful considerations to the accuracy of the methods used.

3.1 Neighbourhoods

The data of the neighbourhoods in Greater Manchester can be extracted out by web scraping using BeautifulSoup library for Python. The neighbourhood data is scraped from a Wikipedia webpage.

```
from bs4 import BeautifulSoup
import requests

source = requests.get('https://en.wikipedia.org/wiki/Category:Areas_of_Greater_Manchester').text
soup = BeautifulSoup(source, 'lxml')
dummy = soup.find_all(class_='mw-category-group')
man = []
for i in range(len(dummy)):
    lists = dummy[i].find_all('a')
    for list in lists:
        temp = list.get('title')
        man.append(temp)

man

for i in range(len(man)):
    if i<=20:
        man.remove(man[i])

print(man)
```

3.2 Geocoding

The file contents are retrieved into a Pandas DataFrame. The latitude and longitude of the neighbourhoods are retrieved using OpenCage Geocoding API. The geometric location values are then stored into the initial dataframe.

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

```
key = 'ac2a83debe8745ac945cf623945274fb'
geocoder = OpenCageGeocode(key)
```

```
enco = ' , Greater Manchester'

lat = []
lon = []

for name in df['Area Of Manchester']:
    query = str(name) + enco
    result = geocoder.geocode(query)
    lat.append(result[0]['geometry']['lat'])
    lon.append(result[0]['geometry']['lng'])

df['Latitudes'] = lat
df['Longitudes'] = lon
```

3.3 Venue Data

From the location data obtained after Web Scraping and Geocoding, the venue data is found out by passing in the required parameters to the FourSquare API, and creating another DataFrame to contain all the venue details along with the respective neighbourhoods.

```
explore_df_list = []

for i, nbd_name in enumerate(df['Neighborhoods']):
    try :
        ### Getting the data of neighbourhood
        nbd_name = df.loc[i, 'Neighborhoods']
        nbd_lat = df.loc[i, 'Latitude']
        nbd_lng = df.loc[i, 'Longitude']

        radius = 1000 # Setting the radius as 1000 metres
        LIMIT = 30 # Getting the top 30 venues

        url = 'https://api.foursquare.com/v2/venues/explore?client_id={} \
        &client_secret={}&ll={},{}&v={}&radius={}&limit={}'\
        .format(CLIENT_ID, CLIENT_SECRET, nbd_lat, nbd_lng, VERSION, radius, LIMIT)

        results = json.loads(requests.get(url).text)
        results = results['response']['groups'][0]['items']

        nearby = json_normalize(results) # Flattens JSON

         # Filtering the columns
        filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
        nearby = nearby.loc[:, filtered_columns]

        # Renaming the columns
        columns = ['Name', 'Category', 'Latitude', 'Longitude']
        nearby.columns = columns

        # Gets the categories
        nearby['Category'] = nearby.apply(get_category_type, axis=1)

        # Gets the data required
        for i, name in enumerate(nearby['Name']):
            s_list = nearby.loc[i, :].values.tolist()  # Converts the numpy array to a python list
            f_list = [nbd_name, nbd_lat, nbd_lng] + s_list
            explore_df_list.append(f_list)

    except Exception as e:
        pass
```

4 Methodology

A thorough analysis of the principles of methods, rules, and postulates employed have been made in order to ensure the inferences to be made are as accurate as possible.

4.1 Accuracy of the Geocoding API

In the initial development phase with Google Maps Geocoder API, the number of erroneous results were of an appreciable amount, which led to the development of an algorithm to analyze the accuracy of the Geocoding API used. In the algorithm developed, Geocoding API from various providers were tested, and in the end, OpenCage Geocoder API turned out to have the least number of collisions.

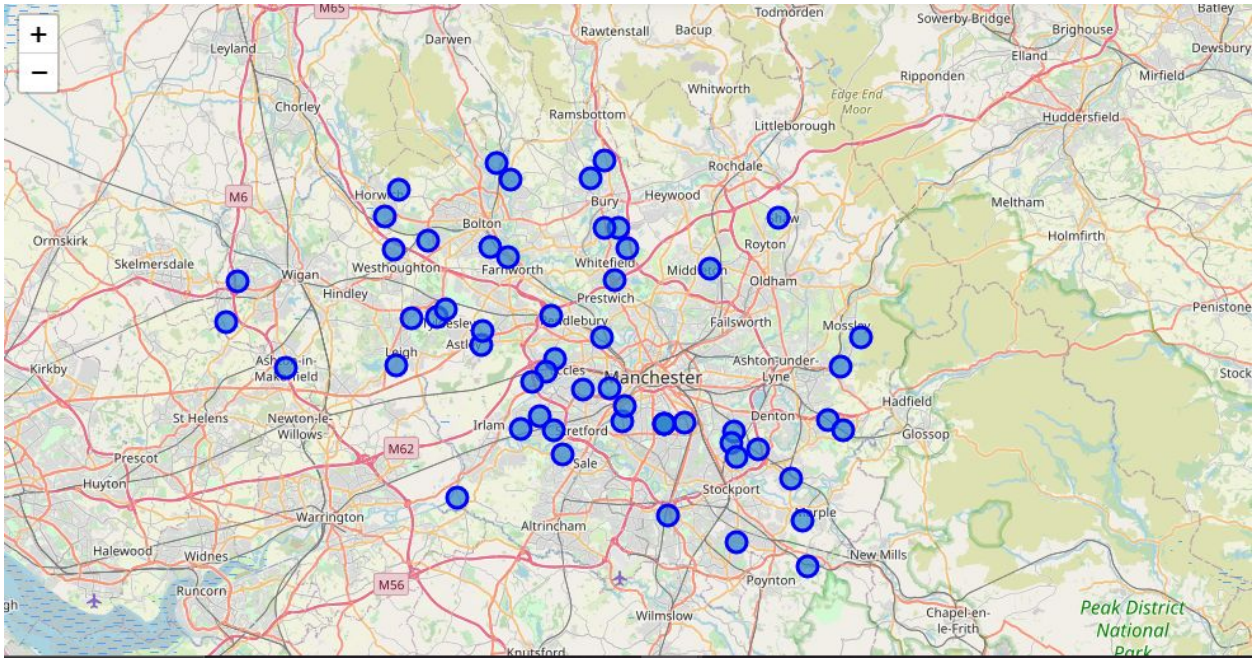
```
col = 0
explored_lat_lng = []
for lat, lng, neighbourhood in zip(df['Latitude'], df['Longitude'], df['Neighborhoods']):
    if (lat, lng) in explored_lat_lng:
        col = col + 1
    else:
        explored_lat_lng.append((lat, lng))

print("Collisions : ", col)
```

```
Collisions :  2
```

4.2 Folium

Folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the leaflet.js library. All cluster visualizations are done with help of Folium which in turn generates a Leaflet map made using OpenStreetMap technology.

```
gman_lat = 53.4576
gman_lng = -2.1578

# Creates map of Greater Manchester using latitude and longitude values
map_man = folium.Map(location=[gman_lat, gman_lng], zoom_start=10)

# Add markers to map
for lat, lng, neighborhood in zip(df['Latitude'], df['Longitude'], df['Neighborhoods']):
    label = '{}'.format(neighborhood)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=8,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_man)

map_man
```



4.3 One hot encoding

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML 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
man_onehot = pd.get_dummies(explore_df[['Venue Category']], prefix="", prefix_sep="")

# Add neighborhood column back to dataframe
man_onehot['Neighbourhood'] = explore_df['Neighbourhood']

# Move neighborhood column to the first column
fixed_columns = [man_onehot.columns[-1]] + man_onehot.columns[:-1].values.tolist()
man_onehot = man_onehot[fixed_columns]

man_onehot
```

4.4 Top 10 most common venues

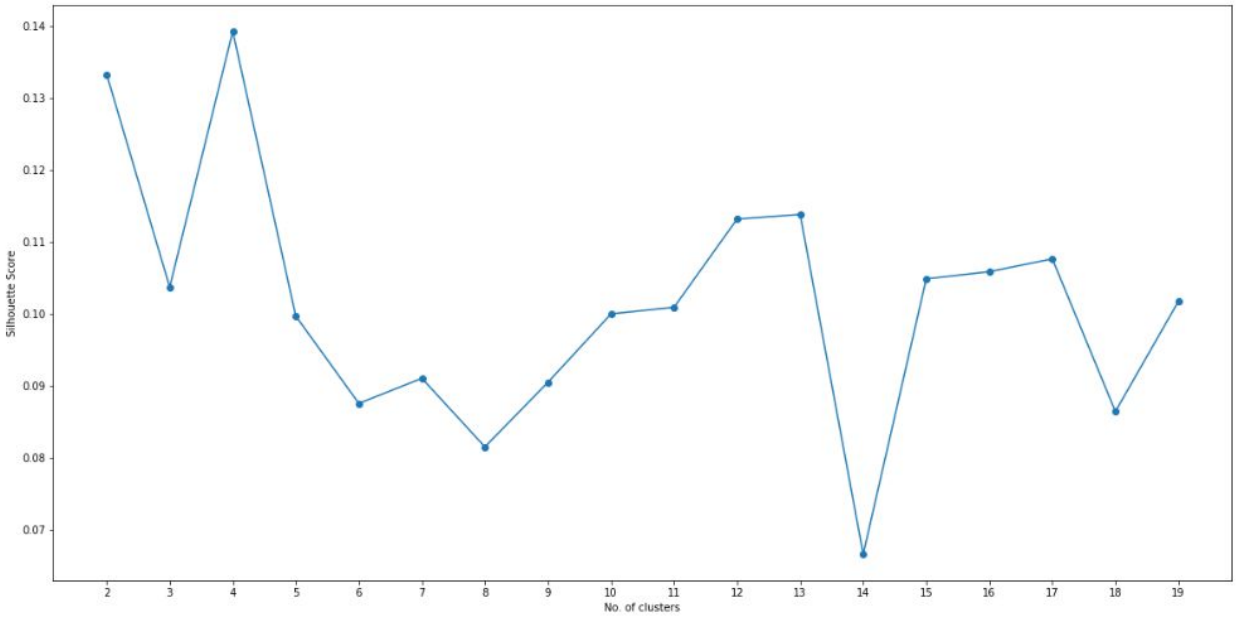
Due to high variety in the venues, only the top 10 common venues are selected and a new DataFrame is made, which is used to train the K-means Clustering Algorithm.

```
num_top_venues = 10
indicators = ['st', 'nd', 'rd']

# Create columns according to number of top venues
columns = ['Neighbourhood']
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
neighbourhoods_venues_sorted = pd.DataFrame(columns=columns)
neighbourhoods_venues_sorted['Neighbourhood'] = man_grouped['Neighbourhood']

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

neighbourhoods_venues_sorted.head()
```

4.5 Optimal number of clusters

Silhouette Score is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. Based on the Silhouette Score of various clusters below 20, the optimal cluster size is determined.

```
import matplotlib.pyplot as plt
%matplotlib inline

def plot(x, y, xlabel, ylabel):
    plt.figure(figsize=(20,10))
    plt.plot(np.arange(2, x), y, 'o-')
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.xticks(np.arange(2, x))
    plt.show()
```




```
from sklearn.metrics import silhouette_samples, silhouette_score

indices = []
scores = []

for kclusters in range(2, max_range) :

    # Run k-means clustering
    kgc = man_grouped_clustering
    kmeans = KMeans(n_clusters = kclusters, init = 'k-means++', random_state = 0).fit_predict(kgc)

    # Gets the score for the clustering operation performed
    score = silhouette_score(kgc, kmeans)

    # Appending the index and score to the respective lists
    indices.append(kclusters)
    scores.append(score)
```

4.6 K-means clustering

The venue data is then trained using K-means Clustering Algorithm to get the desired clusters to base the analysis on. K-means was chosen as the variables (Venue Categories) are huge, and in such situations K-means will be computationally faster than other clustering algorithms.

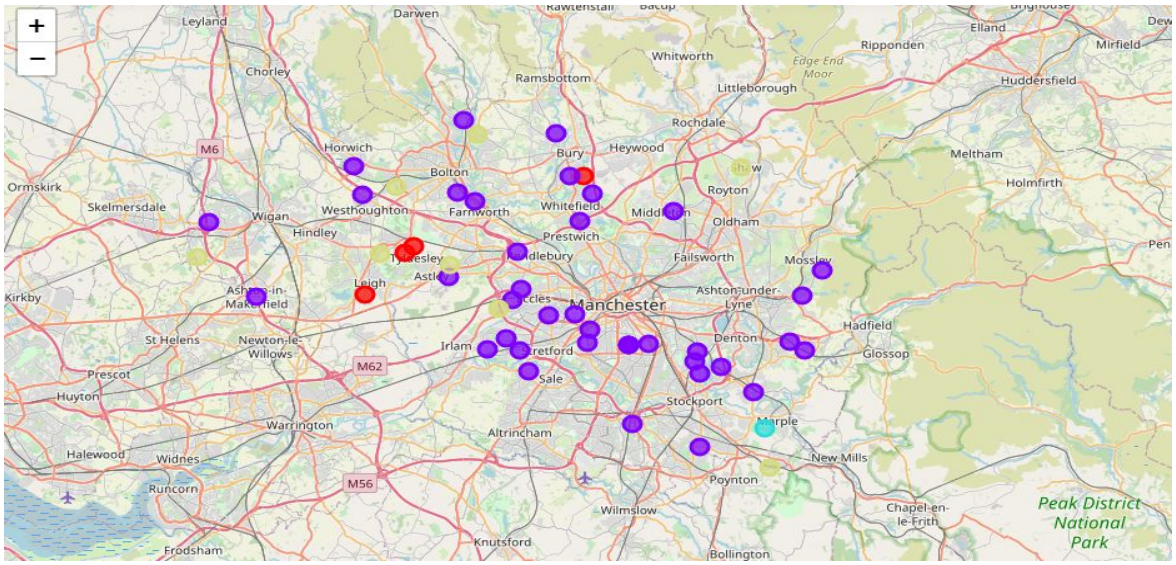
```
kclusters = opt

# Run k-means clustering
kgc = man_grouped_clustering
kmeans = KMeans(n_clusters = kclusters, init = 'k-means++', random_state = 0).fit(kgc)
```

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

5 Results

The neighbourhoods are divided into n clusters where n is the number of clusters found using the optimal approach. The clustered neighbourhoods are visualized using different colours so as to make them distinguishable.

```
# Create map
map_clusters = folium.Map(location=[gman_lat, gman_lng], 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(man_merged['Latitude'], man_merged['Longitude'], man_merged['Neighborhoods'], man_merged['Clus
    label = folium.Popup(str(poi) + ' (Cluster ' + str(cluster + 1) + ')', parse_html=True)
    map_clusters.add_child(
        folium.features.CircleMarker(
        [lat, lon],
        radius=7,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7))

map_clusters
```

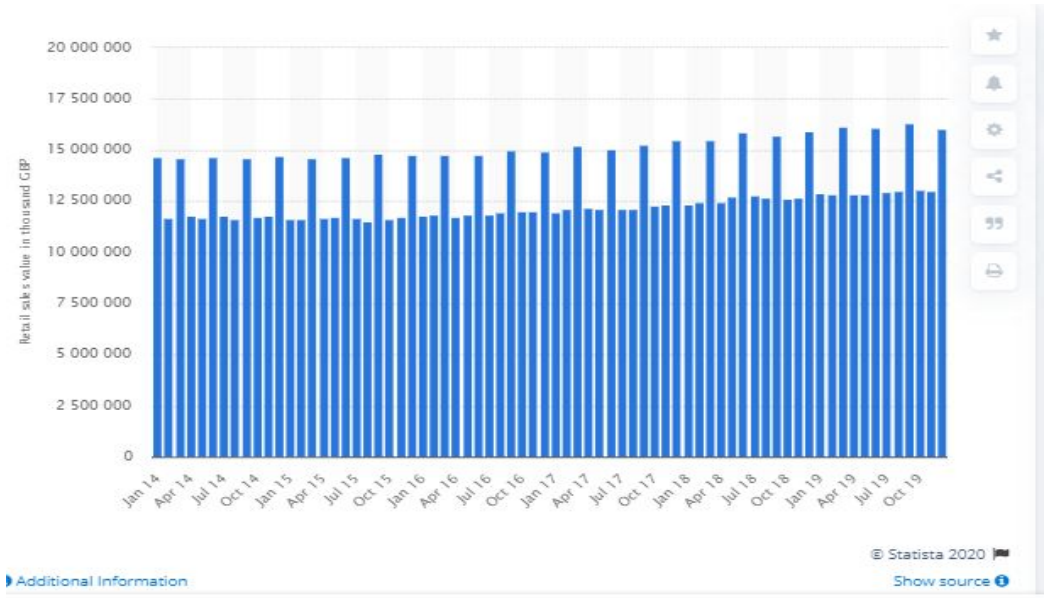


6 Discussion

After analyzing the various clusters produced by the Machine learning algorithm, cluster no. 2 , is a prime fit to solving the problem of finding a cluster with a common venue as a train station mentioned before.

	Neighborhoods	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	9th Most Common Venue	10th Most Common Venue
16	Hindsford	Supermarket	Bar	Roller Rink	Fast Food Restaurant	Soccer Field	Ethiopian Restaurant	Entertainment Service	Event Space	Falafel Restaurant	Fried Chicken Joint
34	Pennington, Greater Manchester	Supermarket	Gym	Movie Theater	Chinese Restaurant	Portuguese Restaurant	Hotel	Sports Club	Stadium	Gastropub	Playground
35	Pilsworth	Supermarket	Gas Station	Hotel	Turkish Restaurant	Fish Market	Fish & Chips Shop	Film Studio	Fast Food Restaurant	Farm	Falafel Restaurant
43	Shakerley	Supermarket	Pub	Train Station	Fast Food Restaurant	Fish Market	Fish & Chips Shop	Film Studio	Farm	Falafel Restaurant	Turkish Restaurant

These four places Hindsford , Pennington , Pilsworth , Shakerley fall in the heart of the Greater Manchester area. This is in direct correlation to them having the highest footfall and daily agglomeration of people of all works of life having different jobs and places to visit.



The above diagram shows Retail sales value monthly in predominantly food stores in Great Britain from January 2014 to December 2019*

The data has been obtained from Statistica website

7 Conclusion

The four identified places in Greater Manchester , United Kingdom have the potential for supermarkets to thrive if developed there. This is due to them being one of the most central and most accessible locations in Manchester having great railway connectivity and bus transport capability. Also these four areas have a large number of small localized markets of different aspects like movie-theaters , gyms , hotels and restaurants which can profit indirectly due to the increased footfall in these areas if supermarkets are localised in these places.