Business Problem
In this scenario, it is urgent to adopt machine learning tools in order to assist home buyers clients in London to make wise and effective decisions. As a result, the business problem we are currently posing is: how could we provide support to home buyers clients in to purchase a suitable real estate in London in this uncertain economic and financial scenario?

To solve this business problem, we are going to cluster London neighborhoods in order to recommend venues and the current average price of real estate where home buyers can make a real estate investment. We will recommend profitable venues according to amenities and essential facilities surrounding such venues i.e. elementary schools, high schools, hospitals & grocery stores.

Data section
Data on London properties and the relative price paid data were extracted from the HM Land Registry (http://landregistry.data.gov.uk/ (http://landregistry.data.gov.uk/)).

To explore and target recommended locations across different venues according to the presence of amenities and essential facilities, we will access data through FourSquare API interface and arrange them as a dataframe for visualization. By merging data on London properties and the relative price paid data from the HM Land Registry and data on amenities and essential facilities surrounding such properties from FourSquare API interface, we will be able to recommend profitable real estate investments.

Methodology section
The Methodology section will describe the main components of our analysis and predication system. The Methodology section comprises four stages:

1. Collect Inspection Data
2. Explore and Understand Data
3. Data preparation and preprocessing
4. Modeling

After importing the necessary libraries, we download the data from the HM Land Registry website as follows:

In [2]:
```
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
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values

!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't completed the Foursquare API lab
import folium # map rendering library

print('Libraries imported.')
```

```
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.5.11
  latest version: 4.7.11

Please update conda by running

    $ conda update -n base -c defaults conda



# All requested packages already installed.

Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.5.11
  latest version: 4.7.11

Please update conda by running

    $ conda update -n base -c defaults conda



# All requested packages already installed.

Libraries imported.
```

In [3]:
```
import os
import datetime as dt # Datetime
import json # library to handle JSON files

import requests # library to handle requests
```

```
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
```

In [4]:
```
df_ppd = pd.read_csv("http://prod2.publicdata.landregistry.gov.uk.s3-website-eu-west-1.amazonaws.com/pp-2018.csv")
```

2. Explore and Understand Data

In [5]:
```
df_ppd.head(5)
```

Out[5]:

	{666758D7-43A9-3363-E053-6B04A8C0D74E}	405000	2018-01-25 00:00	WR15 8LH	D	N	F	RAMBLERS WAY	Unnamed: 8	Unnamed: 9	BORASTON	TENBURY WELLS	SH
0	{666758D7-43AA-3363-E053-6B04A8C0D74E}	315000	2018-01-23 00:00	SY7 8QA	D	N	F	MONT CENISE	NaN	NaN	CLUN	CRAVEN ARMS	SH
1	{666758D7-43AD-3363-E053-6B04A8C0D74E}	165000	2018-01-19 00:00	SY1 2BF	T	Y	F	42	NaN	PENSON WAY	NaN	SHREWSBURY	SH
2	{666758D7-43B0-3363-E053-6B04A8C0D74E}	370000	2018-01-22 00:00	SY8 4DF	D	N	F	WILLOW HEY	NaN	NaN	ASHFORD CARBONEL	LUDLOW	SH
3	{666758D7-43B3-3363-E053-6B04A8C0D74E}	320000	2018-01-19 00:00	TF10 7ET	D	N	F	3	NaN	PRINCESS GARDENS	NaN	NEWPORT	WF
4	{666758D7-43B4-3363-E053-6B04A8C0D74E}	180000	2018-01-31 00:00	SY3 0NQ	S	N	F	79	NaN	LYTHWOOD ROAD	BAYSTON HILL	SHREWSBURY	SH

In [6]:
```
df_ppd.shape
```

Out[6]: (1021214, 16)

3. Data preparation and preprocessing

At this stage, we prepare our dataset for the modeling process

In [7]:
```
# Assign meaningful column names
df_ppd.columns = ['TUID', 'Price', 'Date_Transfer', 'Postcode', 'Prop_Type', 'Old_New', 'Duration', 'PAON', \
            'SAON', 'Street', 'Locality', 'Town_City', 'District', 'County', 'PPD_Cat_Type', 'Record_Status']
```

In [8]:
```
# Format the date column
df_ppd['Date_Transfer'] = df_ppd['Date_Transfer'].apply(pd.to_datetime)

# Delete all obsolete transactions which were done before 2016
df_ppd.drop(df_ppd[df_ppd.Date_Transfer.dt.year < 2016].index, inplace=True)

# Sort by Date of Sale
df_ppd.sort_values(by=['Date_Transfer'],ascending=[False],inplace=True)
```

In [9]:
```
df_ppd_london = df_ppd.query("Town_City == 'LONDON'")

# Make a list of street names in LONDON
streets = df_ppd_london['Street'].unique().tolist()
```

In [10]:
```
df_grp_price = df_ppd_london.groupby(['Street'])['Price'].mean().reset_index()

# Give meaningful names to the columns
df_grp_price.columns = ['Street', 'Avg_Price']
```

In [11]:
```
#Input your Budget's Upper Limit and Lower Limit - Find the locations df_grp_price which fits your budget
df_affordable = df_grp_price.query("(Avg_Price >= 2200000) & (Avg_Price <= 2500000)")
```

In [13]:
```
df_affordable.head(10)
```

Out[13]:

	Street	Avg_Price
196	ALBION SQUARE	2.450000e+06
391	ANHALT ROAD	2.435000e+06
406	ANSDELL TERRACE	2.250000e+06
421	APPLEGARTH ROAD	2.400000e+06
699	AYLESTONE AVENUE	2.286667e+06
853	BARONSMEAD ROAD	2.375000e+06
979	BEAUCLERC ROAD	2.480000e+06
1100	BELVEDERE DRIVE	2.340000e+06
1213	BICKENHALL STREET	2.208500e+06
1251	BIRCHLANDS AVENUE	2.217000e+06

In [ ]:
```
import hmac
from geopy.distance import vincenty
# import k-means from clustering stage
from sklearn.cluster import KMeans
```

In [14]:
```
geolocator = Nominatim()
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_launcher.py:1: DeprecationWarning: Using No
minatim with the default "geopy/1.20.0" `user_agent` is strongly discouraged, as it violates Nominatim's ToS https://opera
tions.osmfoundation.org/policies/nominatim/ and may possibly cause 403 and 429 HTTP errors. Please specify a custom `use
r_agent` with `Nominatim(user_agent="my-application")` or by overriding the default `user_agent`: `geopy.geocoders.opti
ons.default_user_agent = "my-application"`. In geopy 2.0 this will become an exception.
  """Entry point for launching an IPython kernel.
```

In [15]:
```
df_affordable['city_coord'] = df_affordable['Street'].apply(geolocator.geocode).apply(lambda x: (x.latitude, x.longitude))
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [16]:
```
df_affordable.head(10)
```

Out[16]:

	Street	Avg_Price	city_coord
196	ALBION SQUARE	2.450000e+06	(-41.27375755, 173.289393239104)
391	ANHALT ROAD	2.435000e+06	(51.4803265, -0.1667607)
406	ANSDELL TERRACE	2.250000e+06	(51.4998899, -0.1891027)
421	APPLEGARTH ROAD	2.400000e+06	(53.749244, -0.32678)
699	AYLESTONE AVENUE	2.286667e+06	(51.5409157, -0.2178742)
853	BARONSMEAD ROAD	2.375000e+06	(51.4773147, -0.239457)
979	BEAUCLERC ROAD	2.480000e+06	(51.4995771, -0.2290331)
1100	BELVEDERE DRIVE	2.340000e+06	(44.7628418, -63.6692314)
1213	BICKENHALL STREET	2.208500e+06	(51.5211969, -0.1589341)
1251	BIRCHLANDS AVENUE	2.217000e+06	(51.4483941, -0.1604676)

In [17]:
```
df_affordable[['Latitude', 'Longitude']] = df_affordable['city_coord'].apply(pd.Series)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/pandas/core/frame.py:3489: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  self[k1] = value[k2]
```

In [18]:
```
df_affordable.head(10)
```

Out[18]:

	Street	Avg_Price	city_coord	Latitude	Longitude
196	ALBION SQUARE	2.450000e+06	(-41.27375755, 173.289393239104)	-41.273758	173.289393
391	ANHALT ROAD	2.435000e+06	(51.4803265, -0.1667607)	51.480326	-0.166761

406	ANSDELL TERRACE	2.250000e+06	(51.4998899, -0.1891027)	51.499890	-0.189103
421	APPLEGARTH ROAD	2.400000e+06	(53.749244, -0.32678)	53.749244	-0.326780
699	AYLESTONE AVENUE	2.286667e+06	(51.5409157, -0.2178742)	51.540916	-0.217874
853	BARONSMEAD ROAD	2.375000e+06	(51.4773147, -0.239457)	51.477315	-0.239457
979	BEAUCLERC ROAD	2.480000e+06	(51.4995771, -0.2290331)	51.499577	-0.229033
1100	BELVEDERE DRIVE	2.340000e+06	(44.7628418, -63.6692314)	44.762842	-63.669231
1213	BICKENHALL STREET	2.208500e+06	(51.5211969, -0.1589341)	51.521197	-0.158934
1251	BIRCHLANDS AVENUE	2.217000e+06	(51.4483941, -0.1604676)	51.448394	-0.160468

In [19]:
```
df = df_affordable.drop(columns=['city_coord'])
```

In [20]:
```
df.head(8)
```

Out[20]:

	Street	Avg_Price	Latitude	Longitude
196	ALBION SQUARE	2.450000e+06	-41.273758	173.289393
391	ANHALT ROAD	2.435000e+06	51.480326	-0.166761
406	ANSDELL TERRACE	2.250000e+06	51.499890	-0.189103
421	APPLEGARTH ROAD	2.400000e+06	53.749244	-0.326780
699	AYLESTONE AVENUE	2.286667e+06	51.540916	-0.217874
853	BARONSMEAD ROAD	2.375000e+06	51.477315	-0.239457
979	BEAUCLERC ROAD	2.480000e+06	51.499577	-0.229033
1100	BELVEDERE DRIVE	2.340000e+06	44.762842	-63.669231

In [21]:
```
address = 'London, UK'

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

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_launcher.py:3: DeprecationWarning: Using No
minatim with the default "geopy/1.20.0" `user_agent` is strongly discouraged, as it violates Nominatim's ToS https://opera
tions.osmfoundation.org/policies/nominatim/ and may possibly cause 403 and 429 HTTP errors. Please specify a custom `use
r_agent` with `Nominatim(user_agent="my-application")` or by overriding the default `user_agent`: `geopy.geocoders.opti
ons.default_user_agent = "my-application"`. In geopy 2.0 this will become an exception.
  This is separate from the ipykernel package so we can avoid doing imports until
```

The geograpical coordinate of London City are 51.4893335, -0.144055084527687.

In [22]:
```
# create map of London using latitude and longitude values
map_london = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, price, street in zip(df['Latitude'], df['Longitude'], df['Avg_Price'], df['Street']):
    label = '{}, {}'.format(street, price)
    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_london)

map_london
```

Out[22]:

In [23]:
```
#Define Foursquare Credentials and Version
CLIENT_ID = 'QIHHLX0WCGDOCISZEUFQKQBNCE5SQKSRSNN0KMBI0LGREOXY' # Foursquare ID
CLIENT_SECRET = 'CW1CW2XFGFIOXV5BGMB1GED5CIPHVVMFAOBFYBTNJ131A0JB' # Foursquare Secret
VERSION = '20180602' # Foursquare API version
```

In [24]:
```
def getNearbyVenues(names, latitudes, longitudes, radius=500, LIMIT=100):
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.fo
rmat(
```

```
            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 = ['Street',
                'Street Latitude',
                'Street Longitude',
                'Venue',
                'Venue Latitude',
                'Venue Longitude',
                'Venue Category']

    return(nearby_venues)
```

In [25]:

```
# Run the above function on each location and create a new dataframe called location_venues and display it.
location_venues = getNearbyVenues(names=df['Street'],
                                  latitudes=df['Latitude'],
                                  longitudes=df['Longitude'])
```

```
ALBION SQUARE
ANHALT ROAD
ANSDELL TERRACE
APPLEGARTH ROAD
AYLESTONE AVENUE
BARONSMEAD ROAD
BEAUCLERC ROAD
BELVEDERE DRIVE
BICKENHALL STREET
BIRCHLANDS AVENUE
BRAMPTON GROVE
BRIARDALE GARDENS
BROOKWAY
BURBAGE ROAD
BURY WALK
CALLCOTT STREET
CAMPDEN HILL ROAD
CAMPION ROAD
CANNING PLACE
CARLISLE ROAD
CARLTON GARDENS
CARLYLE COURT
CHALCOT SQUARE
CHARLES LANE
CHELSEA CRESCENT
CHESTER CLOSE NORTH
CHEYNE COURT
CHEYNE ROW
CHISWICK MALL
CITY ROAD
CLARENDON STREET
CLONCURRY STREET
COLBECK MEWS
CORNWALL TERRACE MEWS
COURT LANE GARDENS
CRESCENT GROVE
DALEBURY ROAD
DEWHURST ROAD
DORIA ROAD
DOWNSHIRE HILL
DUCHESS WALK
ECCLESTON SQUARE MEWS
EGBERT STREET
EGERTON PLACE
ELM PARK ROAD
FRANK DIXON WAY
FULTON MEWS
GERARD ROAD
GERRARD ROAD
GIRDLERS ROAD
```

GLOUCESTER CRESCENT
GORDON PLACE
GRAFTON SQUARE
GRAHAM TERRACE
HARMAN DRIVE
HARRIS STREET
HAVANNAH STREET
HAZLEWELL ROAD
HEREFORD MEWS
HERONDALE AVENUE
HIGHGATE HIGH STREET
HIGHWOOD HILL
HILLGATE PLACE
HOLLYCROFT AVENUE
HOLLYWOOD MEWS
HONEYWELL ROAD
HORTENSIA ROAD
HOXTON SQUARE
HUNTER ROAD
JACKSONS LANE
JOHN STREET
KINNERTON STREET
KNARESBOROUGH PLACE
KNOX STREET
LADBROKE GROVE
LANCASTER MEWS
LANSDOWNE ROAD
LATIMER INDUSTRIAL ESTATE
LAXTON PLACE
LINCOLN AVENUE
LINGFIELD ROAD
LISSON STREET
LIVERPOOL GROVE
LONGWOOD DRIVE
LONSDALE SQUARE
MAZE HILL
MIDDLESEX PASSAGE
MONTPELIER AVENUE
MONTPELIER WALK
MULTON ROAD
MUNDEN STREET
NORFOLK CRESCENT
NORTH CIRCULAR ROAD
NOTTINGHAM STREET
OAKLEY STREET
OAKWOOD COURT
OBSERVATORY GARDENS
OLD COURT PLACE
ONSLOW MEWS WEST
PALACE PLACE
PANTON STREET
PARK CRESCENT
PARK LANE
PARKE ROAD
PARKFIELDS
PARTHENIA ROAD
PAVILION ROAD
PEMBRIDGE MEWS
PEMBRIDGE ROAD
PEMBROKE STUDIOS
PENCOMBE MEWS
PETERSHAM PLACE
PHILLIMORE GARDENS
PHYSIC PLACE
PITFIELD STREET
PRINCES GATE
PRIORY ROAD
PROTHERO GARDENS
PUTNEY HIGH STREET
QUARRENDON STREET
RADSTOCK STREET
RANELAGH AVENUE
REDCLIFFE ROAD
REEVES MEWS
RHEIDOL MEWS
RINGWOOD AVENUE
RODERICK ROAD
ROPEMAKERS FIELDS
ROYAL CRESCENT
ROYAL HILL
RUSSELL GARDENS MEWS
SETTLES STREET
SHELDON AVENUE
SOUTH END ROW
SOUTHWOOD LAWN ROAD
SOVEREIGN PARK
ST MARGARETS CRESCENT

```
ST OSWALDS PLACE
ST PETERS SQUARE
STAFFORD TERRACE
SUTHERLAND PLACE
SYDNEY STREET
THAMES BANK
THE HEXAGON
TREDEGAR SQUARE
TRINITY STREET
UPPER HAMPSTEAD WALK
WALPOLE GARDENS
WALPOLE STREET
WARWICK SQUARE
WELBECK WAY
WELLESLEY TERRACE
WELLINGTON STREET
WESTMORELAND PLACE
WHITFIELD STREET
WILFRED STREET
WILLOW BRIDGE ROAD
WILSON STREET
WINCHENDON ROAD
WINGATE ROAD
```

In [26]:
```
location_venues.head()
```

Out[26]:

	Street	Street Latitude	Street Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	ALBION SQUARE	-41.273758	173.289393	The Free House	-41.273340	173.287364	Bar
1	ALBION SQUARE	-41.273758	173.289393	The Indian Cafe	-41.273308	173.286530	Indian Restaurant
2	ALBION SQUARE	-41.273758	173.289393	The Bridge Street Collective	-41.272520	173.285517	Café
3	ALBION SQUARE	-41.273758	173.289393	Queen's Gardens	-41.273671	173.291383	Park
4	ALBION SQUARE	-41.273758	173.289393	Urban	-41.274355	173.286317	New American Restaurant

In [29]:
```
location_venues.groupby('Street').count().head()
```

Out[29]:

	Street Latitude	Street Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Street						
ALBION SQUARE	27	27	27	27	27	27
ANHALT ROAD	14	14	14	14	14	14
ANSDELL TERRACE	60	60	60	60	60	60
APPLEGARTH ROAD	4	4	4	4	4	4
AYLESTONE AVENUE	4	4	4	4	4	4

In [30]:
```
location_venues.shape
```

Out[30]: (6038, 7)

In [31]:
```
# one hot encoding
venues_onehot = pd.get_dummies(location_venues[['Venue Category']], prefix="", prefix_sep="")
# add street column back to dataframe
venues_onehot['Street'] = location_venues['Street']
# move street column to the first column
fixed_columns = [venues_onehot.columns[-1]] + list(venues_onehot.columns[:-1])
#fixed_columns
venues_onehot = venues_onehot[fixed_columns]
venues_onehot.head()
```

Out[31]:

	Street	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	American Restaurant	Antique Shop	Arcade	Argentinian Restaurant	Art Gallery	Art Museum
0	ALBION SQUARE	0	0	0	0	0	0	0	0	0	0
1	ALBION SQUARE	0	0	0	0	0	0	0	0	0	0

2	ALBION SQUARE	0	0	0	0	0	0	0	0	0	0
3	ALBION SQUARE	0	0	0	0	0	0	0	0	0	0
4	ALBION SQUARE	0	0	0	0	0	0	0	0	0	0

In [32]:

```
london_grouped = venues_onehot.groupby('Street').mean().reset_index()
london_grouped.head()
```

Out[32]:

	Street	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	American Restaurant	Antique Shop	Arcade	Argentinian Restaurant	Art Gallery	A M
0	ALBION SQUARE	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.037037	0.
1	ANHALT ROAD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.
2	ANSDELL TERRACE	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.016667	0.
3	APPLEGARTH ROAD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.
4	AYLESTONE AVENUE	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.

In [33]:

```
london_grouped.shape
```

Out[33]: (151, 348)

In [34]:

```
# Define a function to return the most common venues/facilities nearby real estate investments#
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]
```

In [35]:

```
num_top_venues = 10
indicators = ['st', 'nd', 'rd']
# create columns according to number of top venues
columns = ['Street']
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))
```

In [36]:

```
# create a new dataframe
venues_sorted = pd.DataFrame(columns=columns)
venues_sorted['Street'] = london_grouped['Street']

for ind in np.arange(london_grouped.shape[0]):
    venues_sorted.iloc[ind, 1:] = return_most_common_venues(london_grouped.iloc[ind, :], num_top_venues)
```

In [37]:

```
venues_sorted.head()
```

Out[37]:

	Street	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	
0	ALBION SQUARE	Café	Bar	Pub	Indian Restaurant	Restaurant	Coffee Shop	New American Restaurant	Burger Joint	French Restaurant	I
1	ANHALT ROAD	Pub	Grocery Store	Japanese Restaurant	Gym / Fitness Center	Diner	Plaza	Pizza Place	Cocktail Bar	English Restaurant	(
2	ANSDELL TERRACE	Italian Restaurant	Clothing Store	Restaurant	Pub	Café	Hotel	Garden	Indian Restaurant	English Restaurant	I I
3	APPLEGARTH ROAD	Pub	Nightclub	Casino	Food & Drink Shop	Farmers Market	Fast Food Restaurant	Filipino Restaurant	Fish & Chips Shop	Fish Market	I I

4	AYLESTONE AVENUE	Park	Café	Movie Theater	Yoga Studio	Food Court	Fast Food Restaurant	Filipino Restaurant	Fish & Chips Shop	Fish Market	I I

In [38]:
```
venues_sorted.shape
```
Out[38]: (151, 11)

In [39]:
```
london_grouped.shape
```
Out[39]: (151, 348)

In [40]:
```
london_grouped=df
```

After our inspection of venues/facilities/amenities nearby the most profitable real estate investments in London, we could begin by clustering properties by venues/facilities/amenities nearby.

In [44]:
```
from sklearn.cluster import KMeans
```

In [45]:
```
#Distribute in 5 Clusters
# set number of clusters
kclusters = 5

london_grouped_clustering = london_grouped.drop('Street', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:50]
```
Out[45]:
```
array([1, 4, 2, 4, 3, 0, 1, 0, 2, 2, 1, 4, 4, 4, 1, 0, 0, 1, 4, 2, 1, 3,
       3, 4, 1, 1, 2, 4, 3, 1, 2, 4, 0, 0, 0, 3, 4, 4, 0, 2, 1, 0, 3, 2,
       3, 2, 3, 2, 2, 4], dtype=int32)
```

In [46]:
```
london_grouped_clustering=df
london_grouped_clustering.head()
```
Out[46]:

	Street	Avg_Price	Latitude	Longitude
196	ALBION SQUARE	2.450000e+06	-41.273758	173.289393
391	ANHALT ROAD	2.435000e+06	51.480326	-0.166761
406	ANSDELL TERRACE	2.250000e+06	51.499890	-0.189103
421	APPLEGARTH ROAD	2.400000e+06	53.749244	-0.326780
699	AYLESTONE AVENUE	2.286667e+06	51.540916	-0.217874

In [47]:
```
london_grouped_clustering.shape
```
Out[47]: (160, 4)

In [48]:
```
london_grouped_clustering['Cluster Labels'] = kmeans.labels_
# merge london_grouped with london_data to add latitude/longitude for each neighborhood
london_grouped_clustering = london_grouped_clustering.join(venues_sorted.set_index('Street'), on='Street')
london_grouped_clustering.head(8)
```
Out[48]:

	Street	Avg_Price	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th M Comr Venu
196	ALBION SQUARE	2.450000e+06	-41.273758	173.289393	1	Café	Bar	Pub	Indian Restaurant	Resta
391	ANHALT ROAD	2.435000e+06	51.480326	-0.166761	4	Pub	Grocery Store	Japanese Restaurant	Gym / Fitness Center	Diner
406	ANSDELL TERRACE	2.250000e+06	51.499890	-0.189103	2	Italian Restaurant	Clothing Store	Restaurant	Pub	Café
421	APPLEGARTH ROAD	2.400000e+06	53.749244	-0.326780	4	Pub	Nightclub	Casino	Food & Drink Shop	Farme Marke
699	AYLESTONE AVENUE	2.286667e+06	51.540916	-0.217874	3	Park	Café	Movie Theater	Yoga Studio	Food Court

	AVENUE							Theater	Studio	Court
853	BARONSMEAD ROAD	2.375000e+06	51.477315	-0.239457	0	Pub	Thai Restaurant	Park	Nature Preserve	Indie Movie Theat
979	BEAUCLERC ROAD	2.480000e+06	51.499577	-0.229033	1	Pub	Coffee Shop	Hotel	Chinese Restaurant	Groce Store
1100	BELVEDERE DRIVE	2.340000e+06	44.762842	-63.669231	0	Bowling Alley	Gas Station	Yoga Studio	Food Court	Farme Marke

Results and Discussion section

First of all, even though the London Housing Market may be in a rut, it is still an "ever-green" for business affairs. We may discuss our results under two main perspectives. First, we may examine them according to neighborhoods/London areas. It is interesting to note that, although West London and North-West London might be considered highly profitable venues to purchase a real estate according to amenities and essential facilities surrounding such venues i.e. elementary schools, high schools, hospitals & grocery stores, South-West London and North-West London are arising as next future elite venues with a wide range of amenities and facilities. Accordingly, one might target under-priced real estates in these areas of London in order to make a business affair. Second, we may analyze our results according to the five clusters we have produced. Even though, all clusters could praise an optimal range of facilities and amenities, we have found two main patterns. The first pattern we are referring to, i.e. Clusters 0, 2 and 4, may target home buyers prone to live in 'green' areas with parks, waterfronts. Instead, the second pattern we are referring to, i.e. Clusters 1 and 3, may target individuals who love pubs, theatres and soccer.

Conclusion

Finally, we drew the conclusion that even though the London Housing Market may be in a rut, it is still an "ever-green" for business affairs. We discussed our results under two main perspectives. First, we examined them according to neighborhoods/London areas although West London and North-West London might be considered highly profitable venues to purchase a real estate according to amenities and essential facilities surrounding such venues i.e. elementary schools, high schools, hospitals & grocery stores, South-West London and North-West London are arising as next future elite venues with a wide range of amenities and facilities. Accordingly, one might target under-priced real estates in these areas of London in order to make a business affair. Second, we analyzed our results according to the five clusters we produced. While Clusters 0, 2 and 4 may target home buyers prone to live in 'green' areas with parks, waterfronts, Clusters 1 and 3 may target individuals who love pubs, theatres and soccer.

```
In [ ]:
```