Capstone Project - The Battle of Neighbourhoods

Business Problem section

Background

According to Bloomberg News, the London Housing Market is in a rut. It is now facing a number of different headwinds, including the prospect of higher taxes and a warning from the Bank of England that U.K. home values could fall as much as 30 percent in the event of a disorderly exit from the European Union. More specifically, four overlooked cracks suggest that the London market may be in worse shape than many realize: hidden price falls, record-low sales, homebuilder exodus and tax hikes addressing overseas buyers of homes in England and Wales.

Business Problem

In this scenario, it is urgent to adopt machine learning tools in order to assist home-buyers clientele 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 clientele 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 neighbourhoods in order to recommend venues and the current average price of real estate where homebuyers 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/). The following fields comprise the address data included in Price Paid Data: Postcode; PAON Primary Addressable Object Name. Typically the house number or name; SAON Secondary Addressable Object Name. If there is a sub-building, for example, the building is divided into flats, there will be a SAON; Street; Locality; Town/City; District; County.

To explore and target recommended locations across different venues according to the presence of amenities and essential facilities, we will access data through Four-Square API interface and arrange them as a data frame 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 Four-Square 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

1. Collect Inspection Data

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

```
import os
import numpy as np
import pandas as pd
import datetime as dt # Datetime
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 folium #import folium # map rendering library
print ('Libraries imported.')



#Read the data for examination (Source: http://landregistry.data.gov.uk/)

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

We read the dataset that we collected from the HM Land Registry website into a pandas' data frame and display the first five rows of it as follows:

```
df_ppd.head(5)

df_ppd.shape
```

3. Data preparation and pre-processing

At this stage, we prepare our dataset for the modeling process, opting for the most suitable machine learning algorithm for our scope. Accordingly, we perform the following steps:

- Rename the column names
- Format the date column
- Sort data by date of sale
- Select data only for the city of London
- Make a list of street names in London
- Calculate the street-wise average price of the property
- Read the street-wise coordinates into a data frame, eliminating recurring word London from individual names
- Join the data to find the coordinates of locations which fit into client's budget
- Plot recommended locations on London map along with current market prices

```
# 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']
```

```
# 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)

df_ppd_london = df_ppd.query("Town_City == 'LONDON'")

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

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']

#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)")

# Display the data frame
df_affordable
```

	Street	Avg_Price
20	ABBOTSBURY CLOSE	2.367093e+06
178	ALBION SQUARE	2.450000e+06
355	ANHALT ROAD	2.435000e+06

	Street	Avg_Price
368	ANSDELL TERRACE	2.250000e+06
381	APPLEGARTH ROAD	2.400000e+06
617	AYLESTONE AVENUE	2.286667e+06
753	BARONSMEAD ROAD	2.375000e+06
867	BEAUCLERC ROAD	2.480000e+06
1079	BICKENHALL STREET	2.351667e+06
1094	BILLING ROAD	2.200000e+06
1108	BIRCHLANDS AVENUE	2.217000e+06
1310	BOWERDEAN STREET	2.300000e+06

131 rows × 2 columns

```
import pandas as pd
import numpy as np
import datetime as DT
import hmac
from geopy.geocoders import Nominatim
from geopy.distance import vincenty

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

for index, item in df_affordable.iterrows():
    print(f"index: {index}")
    print(f"item: {item}")
    print(f"item.Street only: {item.Street}")

geolocator = Nominatim()

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

	Street	Avg_Price	city_coord
20	ABBOTSBURY CLOSE	2.367093e+06	(51.5322588, -0.0061531)
178	ALBION SQUARE	2.450000e+06	(-41.27375755, 173.289393239104)
355	ANHALT ROAD	2.435000e+06	(51.4803265, -0.1667607)
368	ANSDELL TERRACE	2.250000e+06	(51.4998899, -0.1891027)
381	APPLEGARTH ROAD	2.400000e+06	(53.7486539, -0.3266704)
617	AYLESTONE AVENUE	2.286667e+06	(51.5409157, -0.2178742)
753	BARONSMEAD ROAD	2.375000e+06	(51.4773147, -0.239457)
867	BEAUCLERC ROAD	2.480000e+06	(51.4995771, -0.2290331)
1079	BICKENHALL STREET	2.351667e+06	(51.5211969, -0.1589341)
1094	BILLING ROAD	2.200000e+06	(51.4818833, -0.1878624)
1108	BIRCHLANDS AVENUE	2.217000e+06	(51.4483941, -0.1604676)

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

df_affordable
```

	Street	Avg_Price	city_coord	Latitude	Longitude
20	ABBOTSBURY CLOSE	2.367093e+06	(51.5322588, -0.0061531)	51.532259	-0.006153
178	ALBION SQUARE	2.450000e+06	(-41.27375755, 173.289393239104)	-41.273758	173.289393
355	ANHALT ROAD	2.435000e+06	(51.4803265, -0.1667607)	51.480326	-0.166761
368	ANSDELL TERRACE	2.250000e+06	(51.4998899, -0.1891027)	51.499890	-0.189103
381	APPLEGARTH ROAD	2.400000e+06	(53.7486539, -0.3266704)	53.748654	-0.326670
617	AYLESTONE AVENUE	2.286667e+06	(51.5409157, -0.2178742)	51.540916	-0.217874
753	BARONSMEAD ROAD	2.375000e+06	(51.4773147, -0.239457)	51.477315	-0.239457
867	BEAUCLERC ROAD	2.480000e+06	(51.4995771, -0.2290331)	51.499577	-0.229033
1079	BICKENHALL STREET	2.351667e+06	(51.5211969, -0.1589341)	51.521197	-0.158934
1094	BILLING ROAD	2.200000e+06	(51.4818833, -0.1878624)	51.481883	-0.187862

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

	Street	Avg_Price	Latitude	Longitude
20	ABBOTSBURY CLOSE	2.367093e+06	51.532259	-0.006153
178	ALBION SQUARE	2.450000e+06	-41.273758	173.289393
355	ANHALT ROAD	2.435000e+06	51.480326	-0.166761
368	ANSDELL TERRACE	2.250000e+06	51.499890	-0.189103
381	APPLEGARTH ROAD	2.400000e+06	53.748654	-0.326670
617	AYLESTONE AVENUE	2.286667e+06	51.540916	-0.217874
753	BARONSMEAD ROAD	2.375000e+06	51.477315	-0.239457
867	BEAUCLERC ROAD	2.480000e+06	51.499577	-0.229033

```
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))
```

```
# 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


#Define Foursquare Credentials and Version

CLIENT_ID = 'KI3TR0QO4JOKMFELOMF3WSOOI3HFNBF5YLW354MYWBKDHEX3' # Foursquare ID
CLIENT_SECRET = 'QF4ZBLJRBV4BQX52DVWUPEHJ14A2UJABPCZARZQZYTKIISUD' # Foursquare Secret
VERSION = '20181206' # Foursquare API version

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

We can now proceed to the Modeling phase. We will analyze neighborhoods to recommend real estate's where home buyers can make a real estate investment. We will then recommend profitable venues according to amenities and essential facilities surrounding such venues i.e. elementary schools, high schools, hospitals & grocery stores.

4. *Modelling*

After exploring the dataset and gaining insights into it, we are ready to use the clustering methodology to analyze real estates. We will use the k-means clustering technique as it is fast and efficient in terms of computational cost, is highly flexible to account for mutations in real estate market in London and is accurate.

```
def getNearbyVenues(names, latitudes, longitudes, radius=500, LIMIT=100):

    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={}&ll={},{}&radius={}&limit={}'.format(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)
```

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

location_venues
```

	Street	Street Latitude	Street Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	ABBOTSBURY CLOSE	51.532259	-0.006153	Pie Crust Cafe	51.536489	-0.004051	Thai Restaurant
1	ABBOTSBURY CLOSE	51.532259	-0.006153	Three Mills Green	51.528840	-0.006834	Park
2	ABBOTSBURY CLOSE	51.532259	-0.006153	Tesco Express	51.535118	-0.005973	Grocery Store
3	ABBOTSBURY CLOSE	51.532259	-0.006153	Holiday Inn Express	51.536332	-0.004623	Hotel
4	ABBOTSBURY CLOSE	51.532259	-0.006153	Bow Riviera	51.528684	-0.010352	Waterfront
5	ALBION SQUARE	-41.273758	173.289393	The Free House	-41.273340	173.287364	Bar

4869 rows × 7 columns

```
location_venues.groupby('Street').count()
```

	Street Latitude	Street Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Street						
ABBOTSBURY CLOSE	5	5	5	5	5	5
ALBION SQUARE	27	27	27	27	27	27
ANHALT ROAD	14	14	14	14	14	14

125 rows × 6 columns

```
# get the List of Unique Categories
print('There are {} uniques categories.'.format(len(location_venues['Venue
Category'].unique())))

location_venues.shape

#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()


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


london_grouped.shape


# What are the top 5 venues/facilities nearby profitable real estate investments?#

num_top_venues = 5

for hood in london_grouped['Street']:
    print("----"+hood+"----")
    temp = london_grouped[london_grouped['Street'] == hood].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_venues))
    print('\n')


# 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]


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))
```

```
# 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)
```

```
venues_sorted.head()
```

	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	10th Most Common Venue
0	ABBOTSBURY CLOSE	Grocery Store	Park	Waterfront	Hotel	Thai Restaurant	Farm	Eastern European Restaurant	Electronics Store	English Restaurant	Event Space
1	ALBION SQUARE	Café	Restaurant	Indian Restaurant	Bar	Coffee Shop	Pub	New American Restaurant	Seafood Restaurant	Fish & Chips Shop	Brewery
2	ANHALT ROAD	Pub	Plaza	Pizza Place	Grocery Store	Japanese Restaurant	French Restaurant	English Restaurant	Gym / Fitness Center	Diner	Garden
3	ANSDELL TERRACE	Clothing Store	Italian Restaurant	Café	English Restaurant	Pub	Juice Bar	Hotel	Indian Restaurant	Bakery	Garden
4	APPLEGARTH ROAD	Bar	Pub	Casino	Nightclub	Fast Food Restaurant	English Restaurant	Event Space	Exhibit	Falafel Restaurant	Farm

```
venues_sorted.shape
```

```
london_grouped.shape
```

```
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.

```
#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]



#Dataframe to include Clusters

london_grouped_clustering=df
london_grouped_clustering.head()
```

	Street	Avg_Price	Latitude	Longitude
20	ABBOTSBURY CLOSE	2.367093e+06	51.532259	-0.006153
178	ALBION SQUARE	2.450000e+06	-41.273758	173.289393
355	ANHALT ROAD	2.435000e+06	51.480326	-0.166761
368	ANSDELL TERRACE	2.250000e+06	51.499890	-0.189103
381	APPLEGARTH ROAD	2.400000e+06	53.748654	-0.326670

```
london_grouped_clustering.shape
df.shape

london_grouped_clustering.dtypes

df.dtypes


# add clustering labels
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(30) # check the last columns!



# 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(london_grouped_clustering['Latitude'],
london_grouped_clustering['Longitude'], london_grouped_clustering['Street'],
london_grouped_clustering['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_clusters)

map_clusters

london_grouped_clustering.loc[london_grouped_clustering['Cluster Labels'] == 0,
london_grouped_clustering.columns[[1] + list(range(5,
london_grouped_clustering.shape[1]))]].head()

london_grouped_clustering.loc[london_grouped_clustering['Cluster Labels'] == 1,
london_grouped_clustering.columns[[1] + list(range(5,
london_grouped_clustering.shape[1]))]].head()

london_grouped_clustering.loc[london_grouped_clustering['Cluster Labels'] == 2,
london_grouped_clustering.columns[[1] + list(range(5,
london_grouped_clustering.shape[1]))]].head()

london_grouped_clustering.loc[london_grouped_clustering['Cluster Labels'] == 3,
london_grouped_clustering.columns[[1] + list(range(5,
london_grouped_clustering.shape[1]))]].head()

london_grouped_clustering.loc[london_grouped_clustering['Cluster Labels'] == 4,
london_grouped_clustering.columns[[1] + list(range(5,
london_grouped_clustering.shape[1]))]].head()
```

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 neighbourhoods/London areas. It is interesting to note that, although West London (Notting Hill, Kensington, Chelsea, Marylebone) and North-West London (Hampstead) 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 (Wands worth, Balham) and North-West London (Islington) 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 analyse 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

To sum up, according to Bloomberg News, the London Housing Market is in a rut. It is now facing a number of different headwinds, including the prospect of higher taxes and a warning from the Bank of England that U.K. home values could fall as much as 30 percent in the event of a disorderly exit from the European Union. In this scenario, it is urgent to adopt machine learning tools in order to assist home-buyers clientele in London to make wise and effective decisions. As a result, the business problem we were posing was: how could we provide support to home-buyers clientele in to purchase a suitable real estate in London in this uncertain economic and financial scenario?

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

First, we gathered data on London properties and the relative price paid data were extracted from the HM Land Registry (http://landregistry.data.gov.uk/). Moreover, to explore and target recommended locations across different venues according to the presence of amenities and essential facilities, we accessed data through Four-Square API interface and arranged them as a data frame 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 Four-Square API interface, we were able to recommend profitable real estate investments.

Second, The Methodology section comprised four stages: 1. Collect Inspection Data; 2. Explore and Understand Data; 3. Data preparation and pre-processing; 4. Modelling. In particular, in the modelling section, we used the k-means clustering technique as it is fast and efficient in terms of computational cost, is highly flexible to account for mutations in real estate market in London and is accurate.

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 neighbourhoods/London areas. although West London (Notting Hill, Kensington, Chelsea, Marylebone) and North-West London (Hampstead) 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 (Wands worth, Balham) and North-West London (Islington) 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 analysed 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.