**Capstone Project - The Battle of Neighborhoods (Week 1-2)**

**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 homebuyers 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 homebuyers 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 neighborhoods 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 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

**1. Collect Inspection Data**

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

*#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")

Before using data, we will have to explore and understand it.

**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:

Our dataset consists of over 700000 rows and 16 columns. We will now prepare and preprocess data accordingly.

**3. Data preparation and preprocessing**

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

In [5]:

*# 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']

|  | **Street** | **Avg\_Price** |
| --- | --- | --- |
| **20** | ABBOTSBURY CLOSE | 2.367093e+06 |
| **178** | ALBION SQUARE | 2.450000e+06 |
| **355** | ANHALT ROAD | 2.435000e+06 |
| **368** | ANSDELL TERRACE | 2.250000e+06 |

131 rows × 2 columns

In [11]:

item.Street only: WYCOMBE SQUARE

In [13]:

geolocator = Nominatim()

/home/jupyterlab/conda/lib/python3.6/site-packages/geopy/geocoders/osm.py:143: UserWarning: Using Nominatim with the default "geopy/1.17.0" `user\_agent` is strongly discouraged, as it violates Nominatim's ToS https://operations.osmfoundation.org/policies/nominatim/ and may possibly cause 403 and 429 HTTP errors. Please specify a custom `user\_agent` with `Nominatim(user\_agent="my-application")` or by overriding the default `user\_agent`: `geopy.geocoders.options.default\_user\_agent = "my-application"`. In geopy 2.0 this will become an exception.

UserWarning

In [14]:

df\_affordable['city\_coord'] = df\_affordable['Street'].apply(geolocator.geocode).apply(**lambda** x: (x.latitude, x.longitude))

/home/jupyterlab/conda/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/indexing.html#indexing-view-versus-copy>

"""Entry point for launching an IPython kernel.

|  | **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) |

131 rows × 3 columns

In [16]:

df\_affordable[['Latitude', 'Longitude']] = df\_affordable['city\_coord'].apply(pd.Series)

/home/jupyterlab/conda/lib/python3.6/site-packages/pandas/core/frame.py:3140: 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/indexing.html#indexing-view-versus-copy

self[k1] = value[k2]

df\_affordable

131 rows × 5 columns

df = df\_affordable.drop(columns=['city\_coord'])

df

131 rows × 4 columns

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))

The geograpical coordinate of London City are 51.5073219, -0.1276474.

*# 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)

Your credentails:

CLIENT\_ID: KI3TR0QO4JOKMFELOMF3WSOOI3HFNBF5YLW354MYWBKDHEX3

CLIENT\_SECRET:QF4ZBLJRBV4BQX52DVWUPEHJ14A2UJABPCZARZQZYTKIISUD

We can now proceed to the Modeling phase. We will analyze neighborhoods to recommend real estates 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. Modeling**

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 dataframe called location\_venues and display it.*

location\_venues = getNearbyVenues(names=df['Street'],

latitudes=df['Latitude'],

longitudes=df['Longitude']

)

ABBOTSBURY CLOSE

ALBION SQUARE

ANHALT ROAD

ANSDELL TERRACE

APPLEGARTH ROAD

location\_venues

4869 rows × 7 columns

location\_venues.groupby('Street').count()

125 rows × 6 columns

*# get the List of Unique Categories*

print('There are **{}** uniques categories.'.format(len(location\_venues['Venue Category'].unique())))

There are 313 uniques categories.

location\_venues.shape

(4869, 7)

*# 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()

5 rows × 314 columns

london\_grouped = venues\_onehot.groupby('Street').mean().reset\_index()

london\_grouped

125 rows × 314 columns

london\_grouped.shape

(125, 314)

*# 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**')

----ABBOTSBURY CLOSE----

venue freq

0 Thai Restaurant 0.2

1 Park 0.2

2 Waterfront 0.2

3 Grocery Store 0.2

4 Hotel 0.2

----ALBION SQUARE----

venue freq

0 Café 0.22

1 Bar 0.07

2 Pub 0.07

3 Indian Restaurant 0.07

4 Coffee Shop 0.07

----ANHALT ROAD----

venue freq

0 Pub 0.29

1 Japanese Restaurant 0.07

2 Gym / Fitness Center 0.07

3 Pizza Place 0.07

4 Diner 0.07

----ANSDELL TERRACE----

venue freq

0 Clothing Store 0.07

1 Italian Restaurant 0.07

2 Juice Bar 0.05

3 Hotel 0.05

4 Indian Restaurant 0.05

*# 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()

venues\_sorted.shape

(125, 11)

london\_grouped.shape

(125, 314)

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]

array([2, 0, 4, 1, 4, 3, 2, 0, 2, 1, 1, 3, 0, 4, 3, 2, 2, 3, 4, 1, 3, 3,

4, 0, 0, 1, 3, 1, 4, 4, 2, 1, 2, 2, 4, 4, 4, 3, 1, 4, 3, 2, 3, 1,

2, 4, 1, 1, 1, 1], dtype=int32)

*#Dataframe to include Clusters*

london\_grouped\_clustering=df

london\_grouped\_clustering.head()

london\_grouped\_clustering.shape

(131, 4)

df.shape

(131, 4)

london\_grouped\_clustering.dtypes

Street object

Avg\_Price float64

Latitude float64

Longitude float64

dtype: object

df.dtypes

Street object

Avg\_Price float64

Latitude float64

Longitude float64

dtype: object

*# 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 neighborhoods/London areas. It is interesting to note that, although West London (Notting Hill, Kensington, Chelsea, Marylebone) and North-West London (Hampsted) 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 (Wandsworth, Balham) and North-West London (Isliington) 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**

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 homebuyers 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 homebuyers 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 neighborhoods 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 FourSquare 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 FourSquare 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 preprocessing; 4. Modeling. In particular, in the modeling 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 neighborhoods/London areas. although West London (Notting Hill, Kensington, Chelsea, Marylebone) and North-West London (Hampsted) 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 (Wandsworth, Balham) and North-West London (Isliington) 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.