Coursera Capstone Project

Applied Data Science

Project Title: Analysing Melbourne Housing Market

By: Zin Myint Naung

*****This report only includes "Data" section of the final report.***

2. Data

In order to analyse, we need to define our data set for Melbourne's housing price. The data set needs to have following columns:

- Suburb (List of neighbourhoods)
- Price (Housing prices, to calculate average price for each neighbourhoods)
- Latitude (To visualize suburb location)
- Longitude (To visualize suburb location)
- Nearby Venues (To fetch using Foursquare API)

2.1 Suburb Data

Melbourne's housing price list can be downloaded from Kaggle's open datasets. This dataset save our time from scraping suburban data from other third party website such as Wikipedia. The dataset also contains latitude and longitude values for further analysis. The URL for the data is as shown below:

https://www.kaggle.com/anthonypino/melbourne-housing-market#Melbourne_housing_FULL.csv

The csv file includes total of 21 columns with sold price for each address in a suburb. There are total of 34,857 records in the data set.

In [9]:
```
#CSV from https://www.kaggle.com/anthonypino/melbourne-housing-market#Melbourne_housing_FULL.csv
df = pd.read_csv('Melbourne_housing_FULL.csv')
df.head()
```

Out[9]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	Yea
0	Abbotsford	68 Studley St	2	h	nan	SS	Jellis	3/09/2016	2.50	3067.00	2.00	1.00	1.00	126.00	nan	
1	Abbotsford	85 Turner St	2	h	1480000.00	S	Biggin	3/12/2016	2.50	3067.00	2.00	1.00	1.00	202.00	nan	
2	Abbotsford	25 Bloomburg St	2	h	1035000.00	S	Biggin	4/02/2016	2.50	3067.00	2.00	1.00	0.00	156.00	79.00	19
3	Abbotsford	18/659 Victoria St	3	u	nan	VB	Rounds	4/02/2016	2.50	3067.00	3.00	2.00	1.00	0.00	nan	
4	Abbotsford	5 Charles St	3	h	1465000.00	SP	Biggin	4/03/2017	2.50	3067.00	3.00	2.00	0.00	134.00	150.00	19

In [10]:
```
df.shape
```

Out[10]: (34857, 21)

2.2 Data Wrangling

There are null values in our price columns. In the case of less data set, we should fill those values by taking assumption such as getting average values, etc. However, since we have large dataset, we would simply drop the null values records. Also drop the columns that are not required for our purpose. Therefore, our data frame should look something like below.

In [11]:
```
df = df[['Suburb', 'Price', 'Postcode', 'Lattitude', 'Longtitude']]
df['Neighbourhood'] = df[['Suburb']]
df.head()
```

Out[11]:

	Suburb	Price	Postcode	Lattitude	Longtitude	Neighbourhood
0	Abbotsford	nan	3067.00	-37.80	145.00	Abbotsford
1	Abbotsford	1480000.00	3067.00	-37.80	145.00	Abbotsford
2	Abbotsford	1035000.00	3067.00	-37.81	144.99	Abbotsford
3	Abbotsford	nan	3067.00	-37.81	145.01	Abbotsford
4	Abbotsford	1465000.00	3067.00	-37.81	144.99	Abbotsford

In [12]:
```
df = df.dropna()
df.shape
```

Out[12]: (20993, 6)

2.3 Calculating *mean (μ)* value

We will calculate the mean value for each suburb using *pandas groupby()* method. Then re-create our data frame using python list as shown below.

In [19]:
```
df.reset_index(drop=True)
df = pd.DataFrame(df.groupby(['Suburb'], sort = False).mean())
df.head()
```

Out[19]:

Suburb	Price	Postcode	Lattitude	Longtitude
Abbotsford	1096603.90	3067.00	-37.80	145.00
Airport West	780529.42	3042.00	-37.72	144.88
Albert Park	1983664.71	3206.00	-37.84	144.95
Alphington	1441155.56	3078.00	-37.78	145.03
Altona	872917.93	3018.00	-37.87	144.82

In [20]:
```
df.shape
```

Out[20]: (338, 4)

In [21]:
```
ls = []

for index, row in df.iterrows():
    ls.append((index, row['Postcode'], row['Price'], row['Lattitude'], row['Longtitude']))
data = pd.DataFrame(ls, columns = ('Suburb', 'Postcode', 'AvgPrice', 'Lattitude', 'Longtitude'))
data.head()
```

Out[21]:

	Suburb	Postcode	AvgPrice	Lattitude	Longtitude
0	Abbotsford	3067.00	1096603.90	-37.80	145.00
1	Airport West	3042.00	780529.42	-37.72	144.88
2	Albert Park	3206.00	1983664.71	-37.84	144.95
3	Alphington	3078.00	1441155.56	-37.78	145.03
4	Altona	3018.00	872917.93	-37.87	144.82