

Coursera
IBM Data Science Professional Course

Capstone project – The Battle of Neighborhoods
**Predicting Residential Units Rental Rates in
Riyadh City**

Alaa J. Jaradat

April 2019

Table of content:

I. Introduction:	2
II. Data description:	3
III. Methodology:	4
IV. Results:	19
V. Discussion:	22
VI. Conclusion:	22

I. Introduction:

Riyadh is the capital and largest city in the Kingdom of Saudi Arabia with a population of more than 5 million people with a total surface area of 1,798 kilometers squared and population density comes to approximately 4,300 people living per square kilometer.

Riyadh city is divided into districts and has approximately 1 million residential units.

The residential units rental prices started to drop starting from a couple of years ago due to number of reasons such as expatriates leaving the country, and the delivery of a number of residential construction projects done by the ministry of housing for Saudi nationals, and those reasons caused a low demand in residential units rental market.

This report is part of the final project in the IBM Data Science Professional course provided by Coursera with a requirement to leverage the Foursquare location data to execute this project.

The goal of this project is to predict residential units rental rates for the current year half (first 6 months 2019) to help the target audience of this report to understand the real estate market in the city of Riyadh and study the effect of surrounding venues of each district on the rental rates. Another goal is to share knowledge learned from this course with other data scientists who are looking for enrolling in this course.

Let us find out if conducting a data science approach for this problem allows us to predict the rental rates and eventually help the audience of this report in making decisions related to this market or at least extract some insights by doing this study.

The audience of this report are:

- Real estate investors.
- Riyadh residents or people planning to relocate to Riyadh.
- Individuals interested in the real estate trends in Riyadh.
- Data analysts and Data scientists that can contribute or get benefit from this project.

II. Data description:

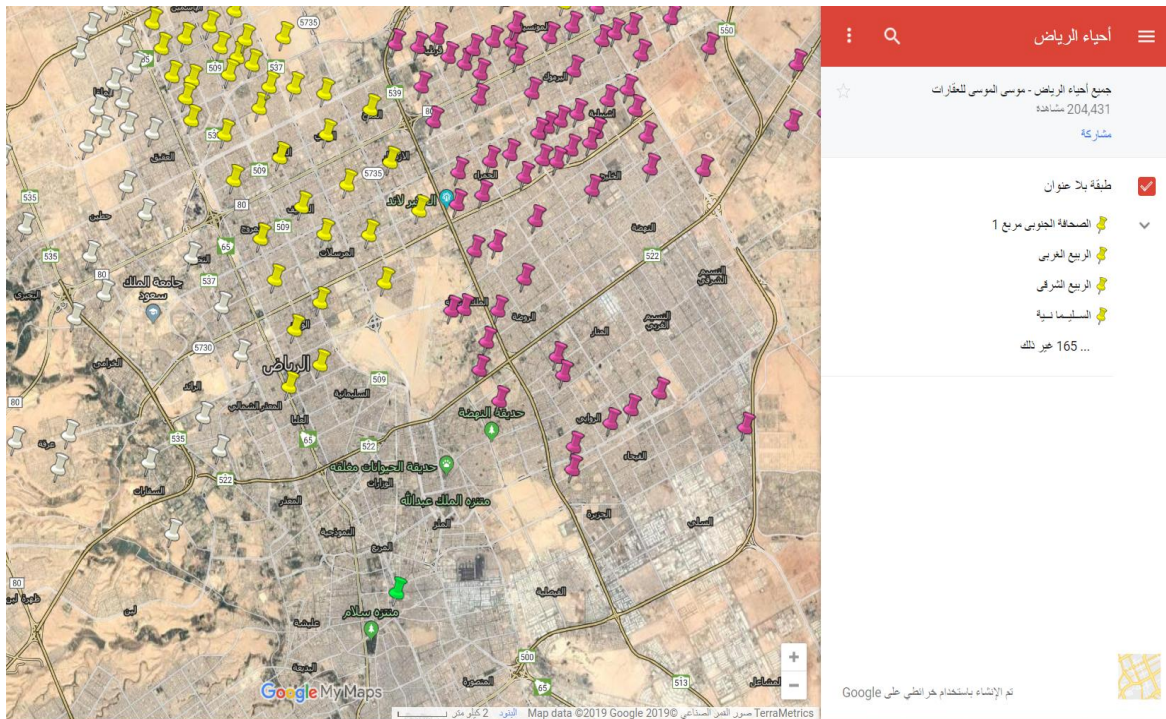
The data used in this project is collected from below data sources:

- **Foursquare.com:**
Foursquare is a web application that provides its users with features including search and discovery of current location, and a social networking tool enabling users share their location with friends via the check-in feature.
Foursquare API is used here to collect information about the surrounding venues to Riyadh districts such as venue name, venue category, venue coordinates.
- **Address.gov.sa:**
National address is a web application created by the Saudi post institute to unify and organize the locations in Saudi Arabia and approved by the Saudi government as the main addressing reference in Saudi Arabia.
National address website offers a web API that enables developers accessing its database to get geospatial information about regions, cities, districts, and other addressing components. This API is used in this project to get the location of Riyadh districts and its location boundaries.
- **Aqar.fm:**
Aqar is a popular web application in Saudi Arabia allowing its users that are looking for buying or selling real estates to search the app for the available real estates that are advertised by other users.
Aqar.fm has page that shows residential units rent prices statistics for each district in a year half time frames.
- Another data source is a web page I found while I was trying to collect data from the internet, and it has Riyadh districts locations, the webpage is part of Google MyMaps service.
Web scraping is used to extract the results from the Google MyMaps page but I did not include it in the analysis after I found a more trusted data source, which is the national address service of Saudi Arabia.

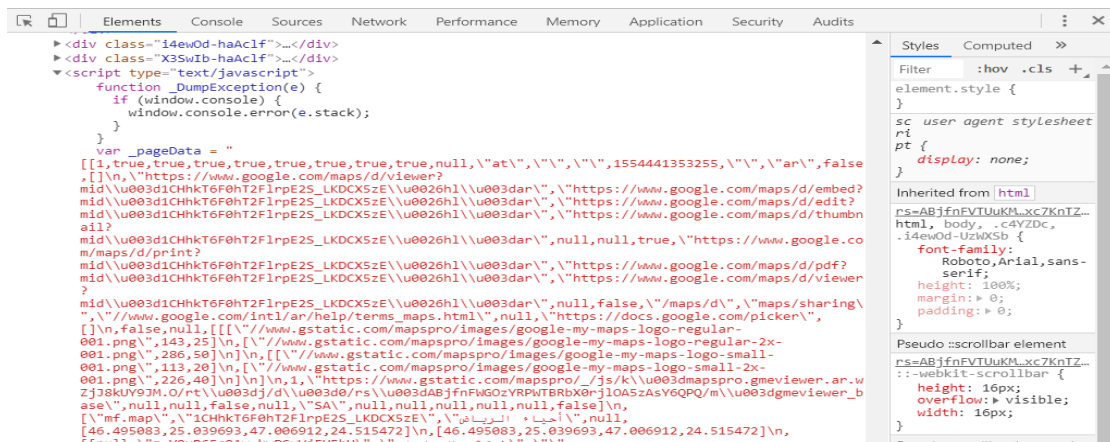
III. Methodology:

Collecting and pre-processing data is the most challenging section in this project since it is not easy to find public data for the selected city.

I started by searching the web to find a page that contains a list of districts of Riyadh city and other information for each district and I found a Google MyMaps page for one of its users and the this user inserted pins for many districts in Riyadh as shown in the snapshot below:



By inspecting the page source code, i was able to do web scraping for the page to get coordinates for the pinned locations in the map:



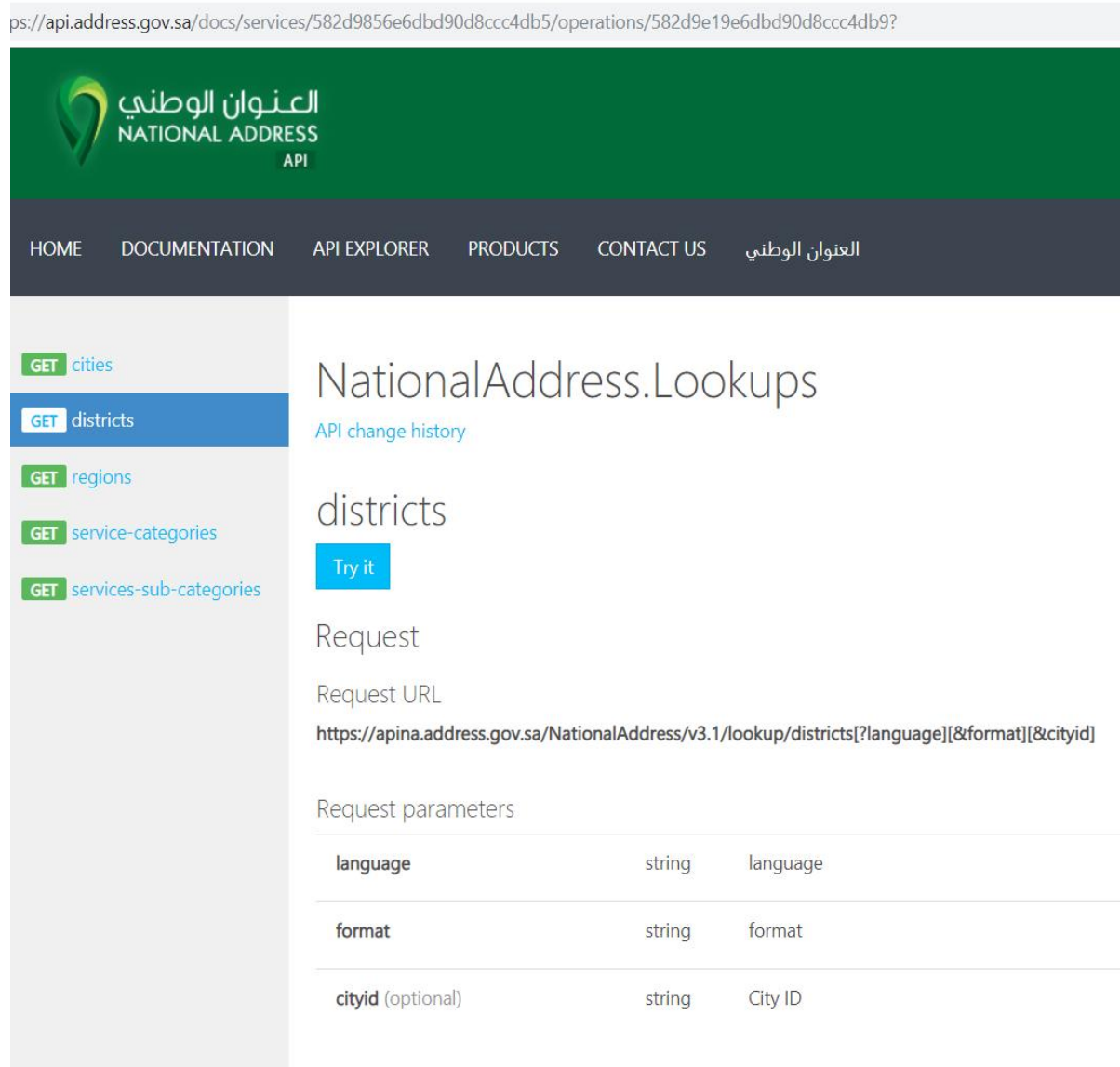
The other data source is api.address.gov.sa and it provides the registered users with access to geospatial information inside Saudi Arabia.

The registration requires entering the national ID number and as a result only parties with valid national ID number are allowed to register.

City, region, district, service categories, nearest POI's and extents of a feature are examples of attributes that can be extracted by this API.

I used it to get a list of all districts in Riyadh:

ps://api.address.gov.sa/docs/services/582d9856e6dbd90d8ccc4db5/operations/582d9e19e6dbd90d8ccc4db9?



The screenshot shows the National Address API documentation page. The header is green with the logo 'العنوان الوطني NATIONAL ADDRESS API'. The navigation bar includes links for HOME, DOCUMENTATION, API EXPLORER, PRODUCTS, CONTACT US, and the logo in Arabic. The left sidebar lists endpoints: GET cities, GET districts (selected), GET regions, GET service-categories, and GET services-sub-categories. The main content area is titled 'NationalAddress.Lookups' and 'districts'. It includes a 'Try it' button, a 'Request' section with the 'Request URL' `https://apina.address.gov.sa/NationalAddress/v3.1/lookup/districts[?language][&format][&cityid]`, and a 'Request parameters' table.

Request parameters		
language	string	language
format	string	format
cityid (optional)	string	City ID

The formal districts list extracted and stored in a pandas data frame:

```
try:
    conn = http.client.HTTPSConnection('apina.address.gov.sa')
    conn.request("GET", "/NationalAddress/v3.1/lookup/districts?%s" % params, "{body}", headers)
    response = conn.getresponse()
    address_data = response.read()
    #print(address_data)
    conn.close()
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))

#####

: import requests
  from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe
  import json

: address_data_decode = address_data.decode("windows-1256")

dict_obj = json.loads(address_data_decode)
dict_obj['Districts'][0]['Id']
Districts_df = pd.DataFrame( dict_obj['Districts'])
Districts_df.head(10)

:
      Id      Name
0  10100003028  المدينة الصناعية الثانية
1  10100003151  جامعة الامام محمد بن سعود الاسلامية
2  10100003166  جامعة الملك سعود
3  10100003059  حي احد
4  10100003137  حي اشبيلية
5  10100003086  حي الازدهار
6  10100003077  حي الاسكان
```

Extents of a feature interface to get location boundaries of each district:

```
minx  miny  maxx  maxy  success  result  statusdescription  fullexception  Id  Name
Geo_df.to_csv(r'C:\Users\Test\Downloads\Course\Data Science\Riyadh_Geo.csv')
```

```
Geo_df = pd.read_csv(r'C:\Users\Test\Downloads\Course\Data Science\Riyadh_Geo.csv', index_col = 0)
Geo_df.head(10)
```

C:\Users\Test\Anaconda3\lib\site-packages\IPython\core\displayhook.py:271: UserWarning: Output cache limit (current) hit.
Flushing oldest 200 entries.
'Flushing oldest {cull_count} entries.'.format(sz=sz, cull_count=cull_count))

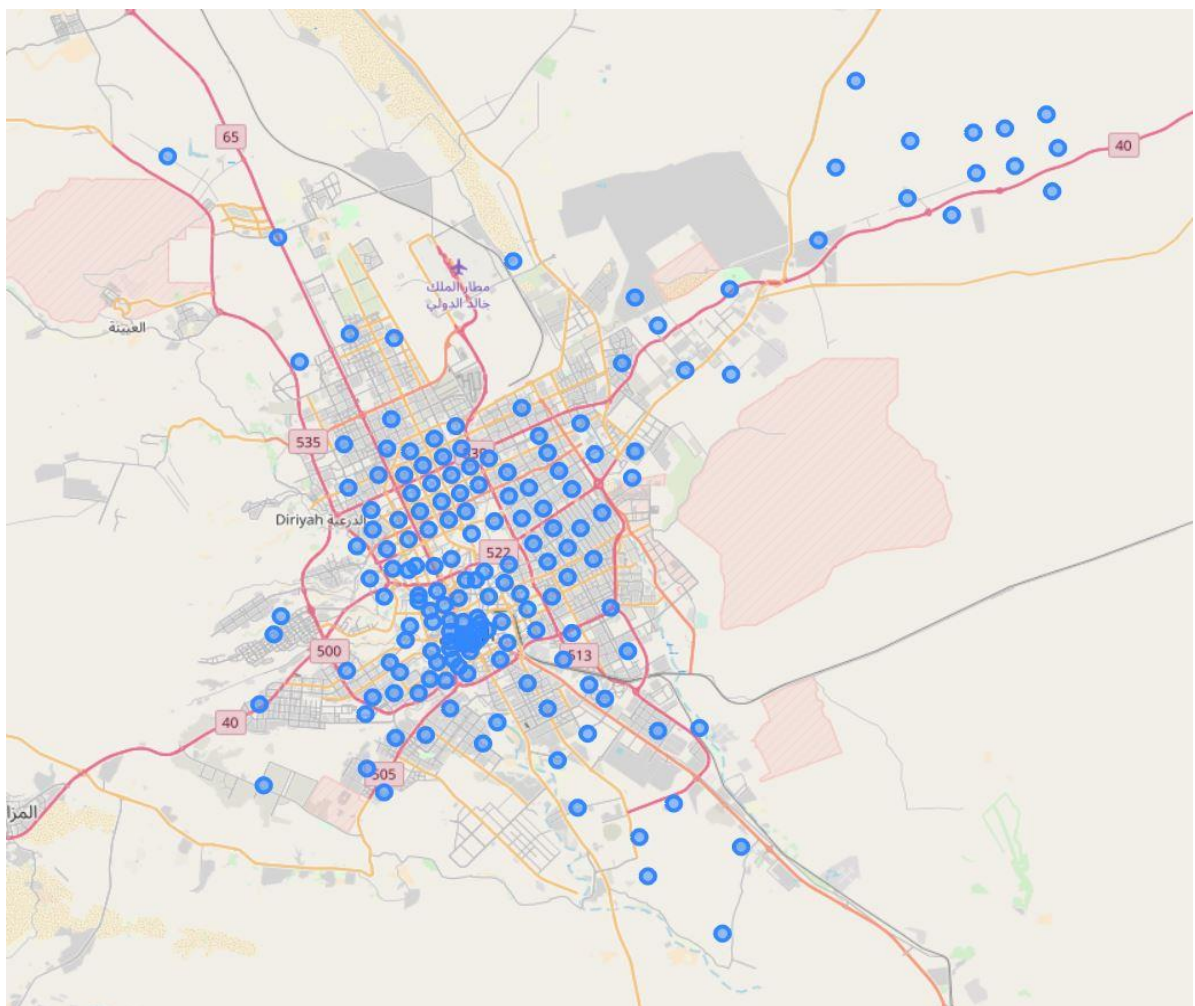
	minx	miny	maxx	maxy	success	result	statusdescription	fullexception	Id	Name
0	46.849667	24.502699	46.946597	24.586056	True	NaN	SUCCESS	NaN	10100003028	المدينة الصناعية الثانية
1	46.687785	24.801407	46.721923	24.830920	True	NaN	SUCCESS	NaN	10100003151	جامعة الامام محمد بن سعود الاسلامية
2	46.602110	24.703914	46.646853	24.744095	True	NaN	SUCCESS	NaN	10100003166	جامعة الملك سعود
3	46.614732	24.472169	46.655266	24.507673	True	NaN	SUCCESS	NaN	10100003059	حي احد
4	46.766300	24.776342	46.818068	24.809261	True	NaN	SUCCESS	NaN	10100003137	حي اشبيلية
5	46.704352	24.768613	46.730595	24.792597	True	NaN	SUCCESS	NaN	10100003086	حي الازدهار
6	46.826500	24.558753	46.869002	24.589107	True	NaN	SUCCESS	NaN	10100003077	حي الاسكان
7	46.774519	24.730206	46.802697	24.756757	True	NaN	SUCCESS	NaN	10100003087	حي الاندلس
8	46.668959	24.605898	46.692829	24.626462	True	NaN	SUCCESS	NaN	10100003071	حي البديعة
9	46.869214	24.498098	47.006928	24.596494	True	NaN	SUCCESS	NaN	10100003094	حي البرية

Since the API retrieves only two coordinate points that is the minimum and maximum points for the extents of the feature, the center of each district is calculated by the center of maximum and minimum points:

```
: Geo_df['longitude'] = ( Geo_df['minx'] + Geo_df['maxx'] ) / 2
Geo_df['latitude'] = ( Geo_df['miny'] + Geo_df['maxy'] ) / 2
Geo_df.rename(columns={'Name': 'District'}, inplace=True)
#Geo_df.reset_index(inplace=True)
Geo_df.head(10)
```

	minx	miny	maxx	maxy	Id	District	longitude	latitude
0	46.849667	24.502699	46.946597	24.586056	10100003028	المدينة الصناعية الثانية	46.898132	24.544378
1	46.687785	24.801407	46.721923	24.830920	10100003151	جامعة الامام محمد بن سعود الاسلامية	46.704854	24.816164
2	46.602110	24.703914	46.646853	24.744095	10100003166	جامعة الملك سعود	46.624481	24.724005
3	46.614732	24.472169	46.655266	24.507673	10100003059	احد	46.634999	24.489921
4	46.766300	24.776342	46.818068	24.809261	10100003137	اشييلية	46.792184	24.792802

Representing the districts centers as circle markers in a folium map:



The residential units rent rates are found in aqar.fm website, the statistics are based on the advertised available residential units for each site grouped by prices mean of each year half time frames.

The statistics for renting three bedroom apartments is selected for analysis and web-scraping technique is applied here to get the data.

أَسْعَارُ شَقَقٍ لِلْإِجَارِ

الإحصائيات المقارنة: أسعار شقق للإيجار

الرياض

متوسط سعر إيجار شقة 3 غرف	النصف الثاني 2015	النصف الأول 2016	النصف الثاني 2016	النصف الأول 2017	النصف الثاني 2017	النصف الأول 2018	النصف الثاني 2018
ريال 21,800							
ريال 27,666							
ريال 26,378							
ريال 24,754							
ريال 23,771							
ريال 21,873							
ريال 21,250							

Data scraped from this page is processed, cleaned and stored in a data frame as follows:

```
Stats_df['Neighborhood'] = Stats_df['Neighborhood'].str.replace('حي', '')
Stats_df['AveragePrice'] = Stats_df['AveragePrice'].str.replace(',', '')
Stats_df['Year_Half'] = Stats_df['Year_Half'].str.replace('الأول', 'First')
Stats_df['Year_Half'] = Stats_df['Year_Half'].str.replace('الثاني', 'Second')
Stats_df['Year_Half'] = Stats_df['Year_Half'].str.replace('First', '0')
Stats_df['Year_Half'] = Stats_df['Year_Half'].str.replace('Second', '0.5')

Stats_df['AveragePrice'] = Stats_df['AveragePrice'].astype(int)
Stats_df['Year'] = Stats_df['Year'].astype(int)
Stats_df['Year_Half'] = Stats_df['Year_Half'].astype(float)
Stats_df['PeriodVal'] = Stats_df['Year'] + Stats_df['Year_Half']
Stats_df.head(10)
```

	Period	Neighborhood	Year_Half	Year	AveragePrice	PeriodVal
0	النصف الأول 2015	اشبيلية	0.0	2015	24138	2015.0
1	النصف الثاني 2015	اشبيلية	0.5	2015	24626	2015.5
2	النصف الأول 2016	اشبيلية	0.0	2016	23944	2016.0
3	النصف الثاني 2016	اشبيلية	0.5	2016	22871	2016.5
4	النصف الأول 2017	اشبيلية	0.0	2017	22625	2017.0
5	النصف الثاني 2017	اشبيلية	0.5	2017	20616	2017.5
6	النصف الأول 2018	اشبيلية	0.0	2018	19205	2018.0
7	النصف الثاني 2018	اشبيلية	0.5	2018	18076	2018.5
8	النصف الثاني 2015	الازدهار	0.5	2015	34181	2015.5
9	النصف الأول 2016	الازدهار	0.0	2016	33600	2016.0

After gathering sufficient amount of data, we can start analyzing the datasets.

To describe the data retrieved from Aqar i issued the following command:

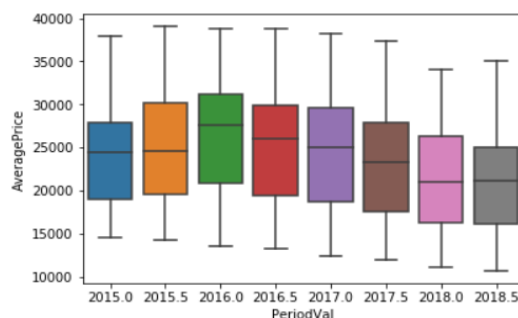
```
pd.DataFrame(Stats_df['AveragePrice']).describe()
```

AveragePrice	
count	666.000000
mean	23791.319820
std	6487.063118
min	10666.000000
25%	18025.250000
50%	23590.000000
75%	28795.500000
max	39050.000000

We have 666 observations with a mean of 23791 SAR and 6487 std.

A box plot for each year half shows nothing unusual and is displayed as:

```
ax = sns.boxplot(x="PeriodVal", y="AveragePrice", data=Stats_df)
```



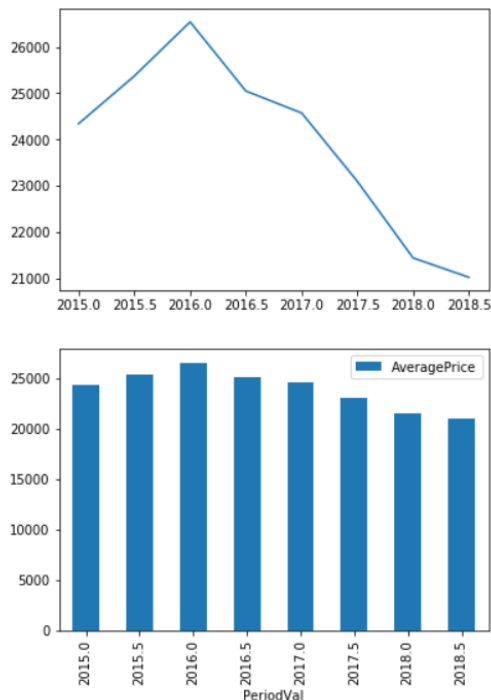
Let's get the means of average prices for each year half and plot it:

```
: Stats_YearHalf_Means = Stats_df.groupby('Period').mean()
Stats_YearHalf_Means.reset_index(inplace=True)
Stats_YearHalf_Means.sort_values(by=['PeriodVal'], inplace=True)
```

```
: Stats_YearHalf_Means
```

	Period	Year_Half	Year	AveragePrice	PeriodVal	
0	2015	النصف الأول	0.0	2015.0	24340.628571	2015.0
4	2015	النصف الثاني	0.5	2015.0	25369.472222	2015.5
1	2016	النصف الأول	0.0	2016.0	26537.556962	2016.0
5	2016	النصف الثاني	0.5	2016.0	25044.525773	2016.5
2	2017	النصف الأول	0.0	2017.0	24571.412371	2017.0
6	2017	النصف الثاني	0.5	2017.0	23093.408163	2017.5
3	2018	النصف الأول	0.0	2018.0	21440.770000	2018.0
7	2018	النصف الثاني	0.5	2018.0	21023.306818	2018.5

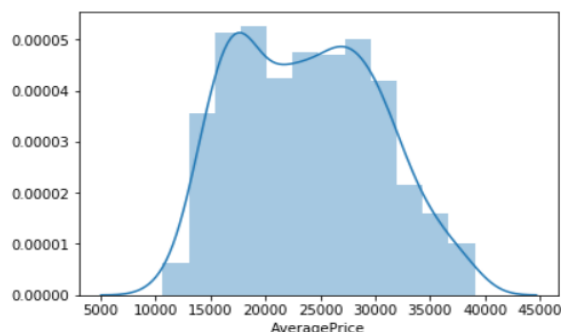
```
: plt.plot(Stats_YearHalf_Means['PeriodVal'], Stats_YearHalf_Means['AveragePrice'])  
ax = Stats_YearHalf_Means.plot.bar(x='PeriodVal', y = 'AveragePrice')
```



The graph clearly shows that the average prices reached its maximum level in the first half of year 2016 and then started to drop deeply after that. This result connects us with the main goal of this project to know whether the prices will continue dropping or not.

Now, do we have a normally distributed data set ?

```
sns.distplot(Stats_df.AveragePrice)  
<matplotlib.axes._subplots.AxesSubplot at 0x1729cb9efd0>
```



The answer is close to yes and therefore we can rely on it and continue our analysis.

To better understand our data, let's build a choropleth map with districts and its average residential units rent rates.

The geospatial data set retrieved from api.address.gov.sa provides only minimum and maximum coordinates and therefore we will assume that each district has a rectangular shape and build our map based on that:

```
GeoList = []
for i in range(0, len(Geo_df)):
    x1 = Geo_df.iloc[i]['minx']
    y1 = Geo_df.iloc[i]['miny']
    x2 = Geo_df.iloc[i]['maxx']
    y2 = Geo_df.iloc[i]['maxy']
    GeoList.append(Polygon( [(x1,y1), (x2,y1), (x2, y2), (x1, y2), (x1, y1)] ) )

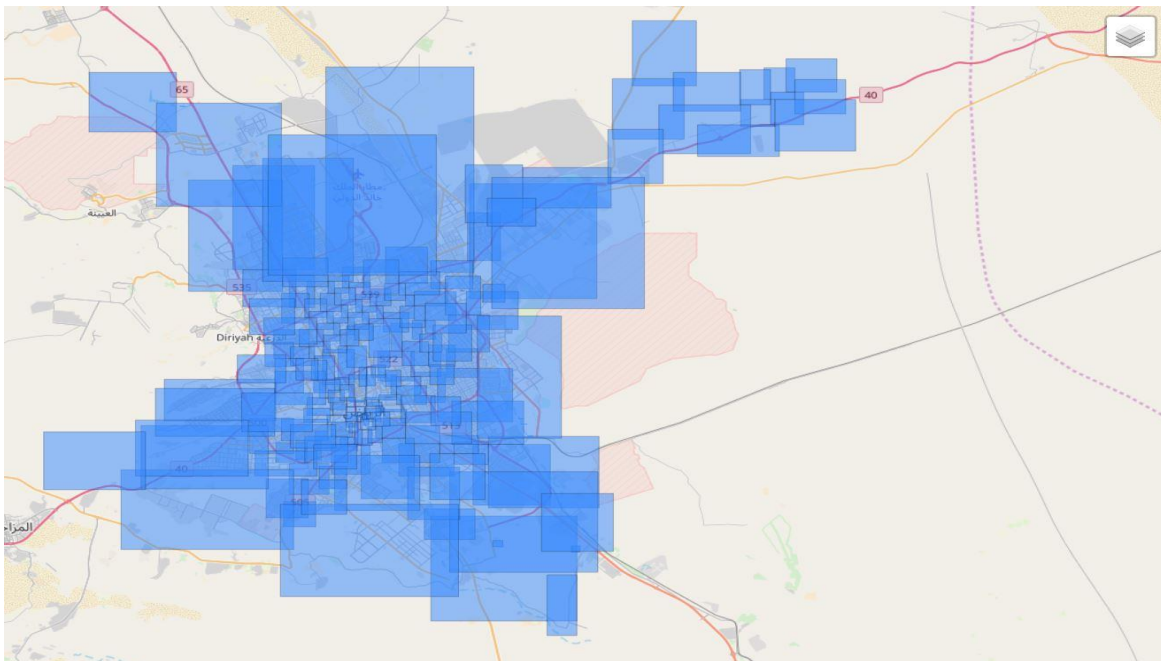
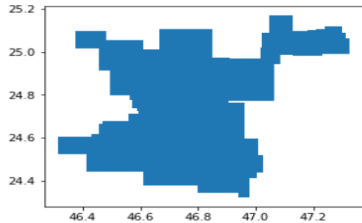
GeoList[:3]

[<shapely.geometry.polygon.Polygon at 0x172999e77b8>,
 <shapely.geometry.polygon.Polygon at 0x172999e7828>,
 <shapely.geometry.polygon.Polygon at 0x172999e7860>]

GeoJson = gpd.GeoSeries(GeoList).to_json()
with open(r"C:\Users\Test\Downloads\Course\Data Science\Riyadh_01.GeoJson", "w") as text_file:
    text_file.write(GeoJson)

gpd.GeoSeries(GeoList).plot()
```

<matplotlib.axes._subplots.AxesSubplot at 0x17299a55550>



The Map is crowded with overlapped rectangles 😞. And the shape looks like a sitting camel. Is there a solution to it?, Let's find out next.

We will first decrease rectangles boundaries by 15%:

	minx	miny	maxx	maxy	Id	District	longitude	latitude
0	46.849667	24.502699	46.946597	24.586056	10100003028	المدينة الصناعية الثانية	46.898132	24.544378
1	46.687785	24.801407	46.721923	24.830920	10100003151	جامعة الإمام محمد بن سعود الإسلامية	46.704854	24.816164
2	46.602110	24.703914	46.646853	24.744095	10100003166	جامعة الملك سعود	46.624481	24.724005
3	46.614732	24.472169	46.655266	24.507673	10100003059	احد	46.634999	24.489921
4	46.766300	24.776342	46.818068	24.809261	10100003137	اشبيلية	46.792184	24.792802

```
scale_ratio = 0.15
```

```
Geo_df['minx'] = Geo_df['minx'] + (Geo_df['maxx'] - Geo_df['minx']) * scale_ratio
Geo_df['miny'] = Geo_df['miny'] + (Geo_df['maxy'] - Geo_df['miny']) * scale_ratio
Geo_df['maxx'] = Geo_df['maxx'] - (Geo_df['maxx'] - Geo_df['minx']) * scale_ratio
Geo_df['maxy'] = Geo_df['maxy'] - (Geo_df['maxy'] - Geo_df['miny']) * scale_ratio
```

And then rotate each rectangle around its centroid by -25 degrees.

```
GeoList[0]
```



```
print(GeoList[0])
```

```
POLYGON (((46.86420629300218 24.51520233729032, 46.93423827110918 24.51520233729032, 46.93423827110918 24.57542830583388, 46.86420629300218 24.57542830583388, 46.86420629300218 24.51520233729032)))
```

```
from shapely import affinity
```

```
for i in range(0, len(GeoList)):
    GeoList[i] = affinity.rotate(GeoList[i], 25, origin='centroid')
```

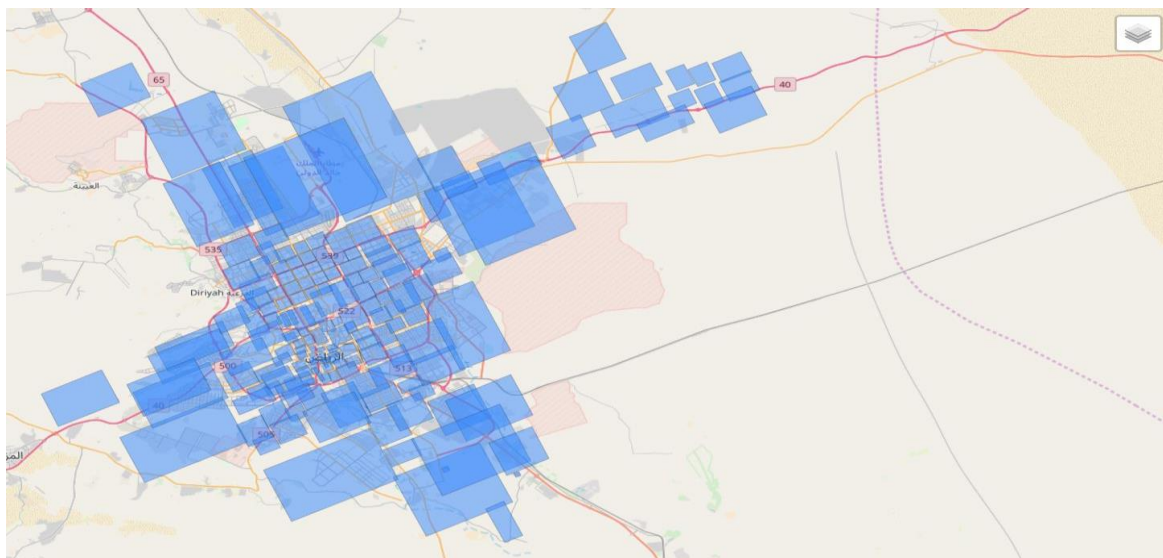
```
len(GeoList)
```

```
187
```

```
GeoList[0]
```



Now our map is looks better, even it looks like a space satellite 😊.



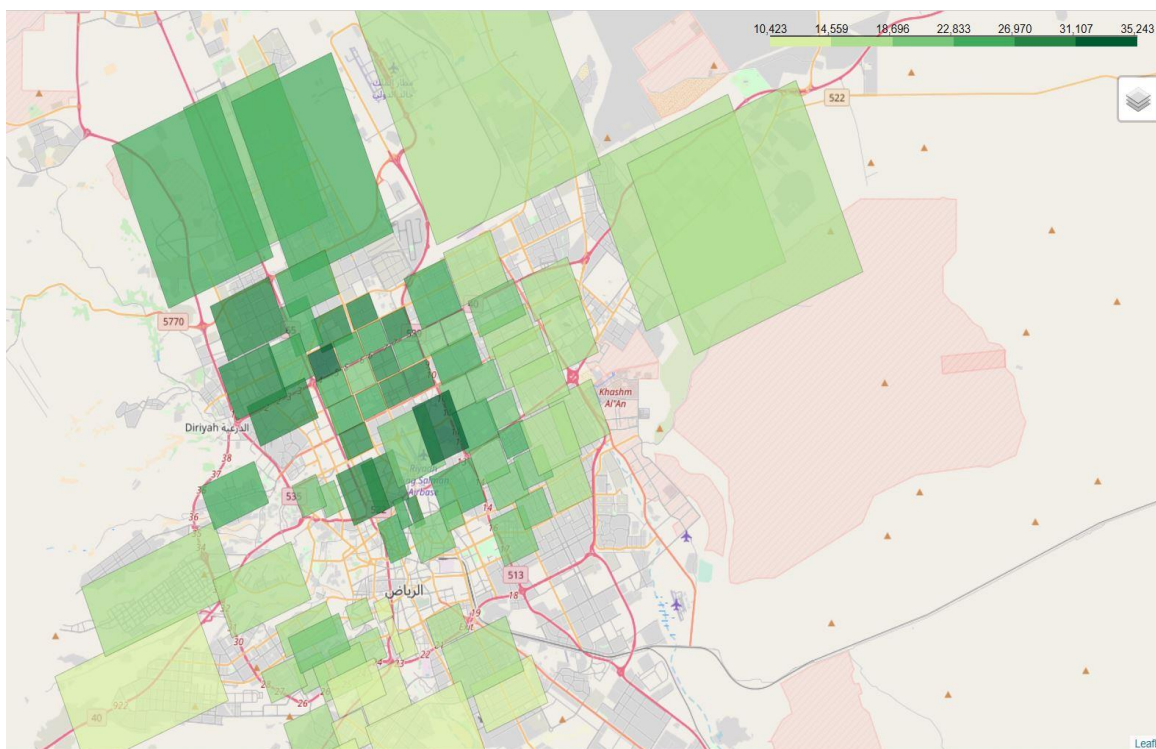
In this point, we can build a choropleth map by joining the Aqar statistics and Address API data frames and this join yields a total of 97 records based on the districts names (the not matched districts are discarded):

	minx	miny	maxx	maxy	Id	District	longitude	latitude
0	46.864206	24.515202	46.934238	24.575428	10100003028	المدينة الصناعية الثانية	46.898132	24.544378
1	46.692906	24.805834	46.717570	24.827157	10100003151	جامعة الإمام محمد بن سعود الإسلامية	46.704854	24.816164
2	46.608821	24.709941	46.641148	24.738972	10100003166	جامعة الملك سعود	46.624481	24.724005
3	46.620812	24.477494	46.650098	24.503146	10100003059	احد	46.634999	24.489921
4	46.774065	24.781280	46.811468	24.805064	10100003137	اشبيلية	46.792184	24.792802

```
Geo_Stats_df = pd.merge(Stats_pivot_df, Geo_df, how='inner', on = 'District')
print( Geo_Stats_df.shape)
Geo_Stats_df.head()
```

(97, 17)

And the average prices of residential units choropleth map for Riyadh city in the 2018 H2 appears as follows:



The choropleth maps shows that districts located in the center and North West of Riyadh city have high average rent rates. Moreover, areas in the South West have the lowest average rent prices.

The question is, Can we dig deeper to know if districts with High/Medium/Low rent prices have venues similarities? Let's see.

FourSquare as a data requirement in this project is our data source for venues located in Riyadh districts.

The FourSquare API python script is built to get most popular venues in Riyadh districts by passing the districts coordinates and a radius of 2000 meters as part of the URL which retrieves data in a JSON data format:

```
: Geo_Stats_df.loc[19, 'District']
: 'الروضة'

: i = 19
  district_latitude = Geo_Stats_df.loc[i, 'latitude']
  district_longitude = Geo_Stats_df.loc[i, 'longitude']

  #neighborhood_latitude = 24.7136
  #neighborhood_longitude = 46.6753

  district_name = Geo_Stats_df.loc[i, 'District'] #

  print('Latitude and longitude values of {} are {}, {}'.format(district_name,
                                                                district_latitude,
                                                                district_longitude))

Latitude and longitude values of الروضة are 24.733827669289504, 46.767914404087406.

: latitude = district_latitude
  longitude = district_longitude
  radius = 2000
  LIMIT = 100

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

: 'https://api.foursquare.com/v2/venues/explore?client_id=5JE0E3FOXGVACK0SU13CMEUDDVCGMXAESVUKYVD0LPFIJDNV&client_secret=SAAYJPKHY0KJYENTC1VPL40TOZVKTG2MMC60GBEEWIDCUA5R&ll=24.733827669289504,46.767914404087406&v=20180605&radius=2000&limit=100'
```

After that, the JSON result is normalized and venue categories and names along with coordinates are extracted and stored in a data frame:

```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
nearby_venues = nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)

# clean columns
nearby_venues.columns = [col.split(".")[1] for col in nearby_venues.columns]

nearby_venues.head(10)
```

	name	categories	lat	lng
0	Al Hatab Bakery (أفان الحطب)	Bakery	24.732104	46.765266
1	Juice Time	Juice Bar	24.732160	46.765153
2	يو إف فيد كيدة غنم وحاشي	Breakfast Spot	24.731759	46.765830
3	ركن قراميشو	Candy Store	24.734315	46.763464
4	Broast Line (بروست لاين)	Fried Chicken Joint	24.732988	46.764510
5	Shawarama Majed (شاورما ماجد)	Fast Food Restaurant	24.732121	46.764773

By the way, 3rd row venue is a buffet that have a good camel lever dishes if you want to try different taste ;) .

At this level we have districts with average rent prices and nearby venues based on the districts coordinates, and in order to distribute the districts into groups according to the most common venues is to some extent a complex problem and can be a time consuming problem if one decided to conduct regular statistics on the data set due to the diversity of venue categories in different areas. And here the power of machine learning methodologies comes into play.

KMeans clustering is an unsupervised machine learning algorithm and selected for this problem to help us in extracting some results and gain some insights about Riyadh city districts based on the nearby venues, but first we need to pre-process the data set to make it ready as an input for the KMeans clustering algorithm which expects a data frame where rows represent the observations that we want to cluster, and the columns represent attributes of the observations.

The data is manipulated by grouping the most common venue categories in each district by the frequent venue categories and therefore now it can be input to the KMeans machine learning algorithm:

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

# create columns according to number of top venues
columns = ['Neighborhood']
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
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = grouped['Neighborhood']

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

neighborhoods_venues_sorted.head()
```

	Neighborhood	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	المنطقة	Donut Shop	Dessert Shop	Fast Food Restaurant	Coffee Shop	Middle Eastern Restaurant	Gym / Fitness Center	Supermarket	Sandwich Place	Ice Cream Shop	Pizza Place
1	الازدهار	Coffee Shop	Cosmetics Shop	Juice Bar	Pharmacy	Breakfast Spot	Dessert Shop	Restaurant	Burger Joint	Health Food Store	Ice Cream Shop
2	الاندلس	Coffee Shop	Ice Cream Shop	Middle Eastern Restaurant	Donut Shop	Fast Food Restaurant	Pharmacy	Breakfast Spot	Pizza Place	Bakery	Café
3	البيعة	Coffee Shop	Fast Food Restaurant	Park	Grocery Store	Café	Donut Shop	Pizza Place	Farm	Ice Cream Shop	Convenience Store
4	التحارون	Coffee Shop	Ice Cream Shop	Pharmacy	Fast Food Restaurant	Burger Joint	Restaurant	Donut Shop	Gift Shop	Dessert Shop	Café

Post a number of trials, K=10 is selected and produced 3 main clusters and the remaining 7 clusters is discarded from further analysis:

```
from sklearn.cluster import KMeans

# set number of clusters
kclusters = 10

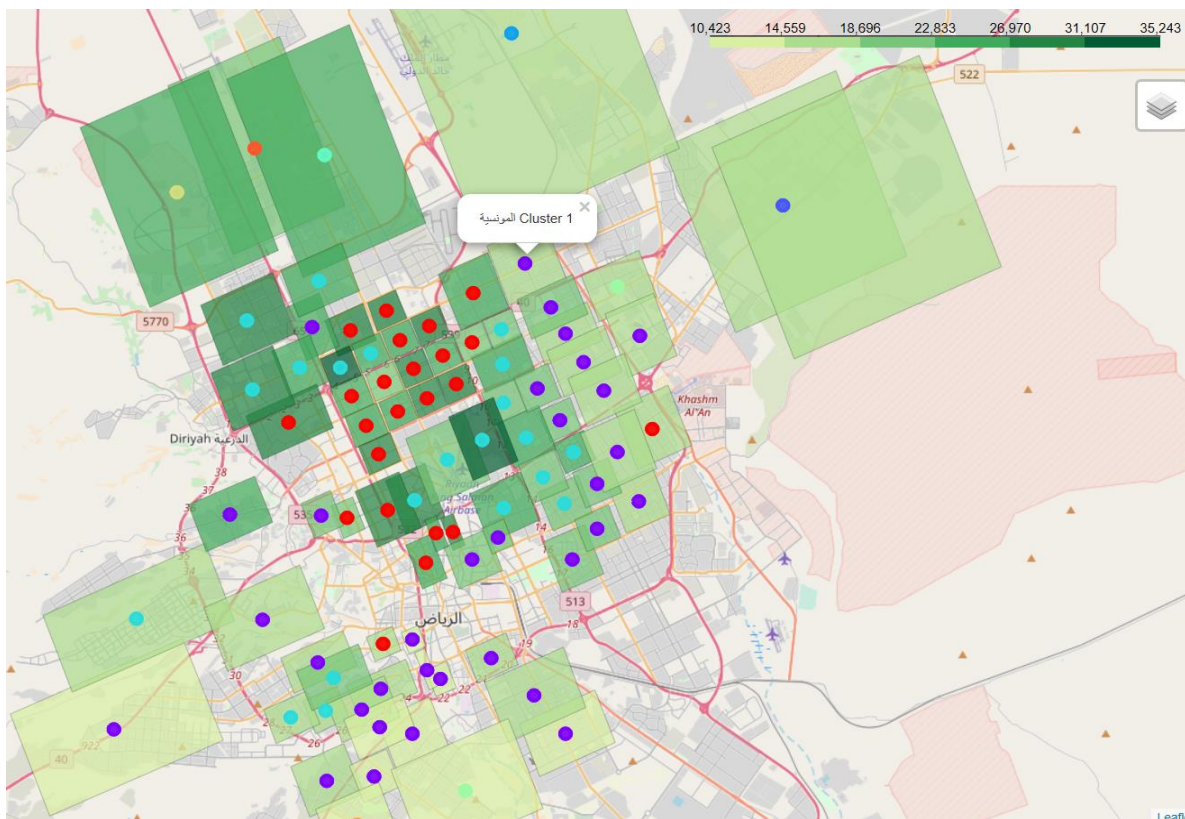
grouped_clustering = grouped.drop('Neighborhood', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:85]

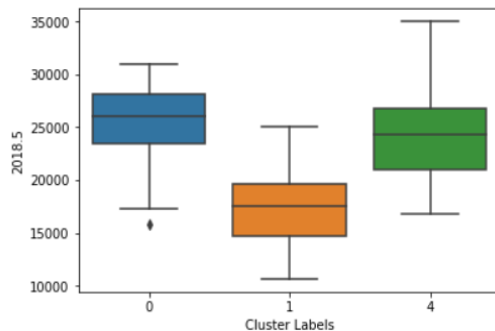
array([1, 0, 1, 0, 0, 1, 1, 2, 1, 4, 1, 1, 4, 0, 3, 4, 4, 4, 4, 1, 1, 4,
       1, 4, 1, 0, 1, 0, 9, 1, 1, 4, 0, 4, 0, 1, 6, 4, 7, 0, 0, 1, 0, 0,
       1, 0, 1, 4, 4, 0, 1, 4, 1, 1, 0, 0, 5, 0, 0, 1, 4, 1, 0, 0, 0,
       4, 1, 1, 0, 1, 6, 1, 4, 1, 1, 4, 4, 1, 1, 8, 4, 0, 1, 1])
```

By representing the clusters on the choropleth Riyadh map using folium library again, we can visually find that Cluster 1 (Purple circle markers) have low average rent prices and Cluster 0 (Red circle markers) have medium to high average rent prices and Cluster 4 (Blue circle markers) comes usually with high average rent prices, and therefore we can conclude that there is a correlation between districts nearby venues and the average rent prices of residential units:



Using statistical approach to understand the data, we find that cluster 0 clearly has different average prices when compared with cluster 1 & cluster 4 together:

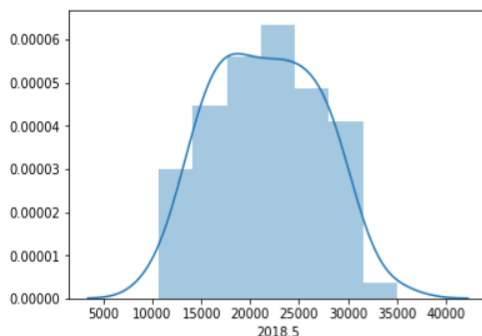
```
ax = sns.boxplot(x="Cluster Labels", y="2018.5", data=clustered_df)
```



And the distribution of average prices for the remaining districts after discarding some of them has a better normal shape when compared to the one when we included all districts in the original data set:

```
sns.distplot(clustered_df['2018.5'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x17295adc198>
```



Based on the results that we achieved so far, we will start now building our model that will represent the average prices of residential units over time and forecast the upcoming time frame average prices.

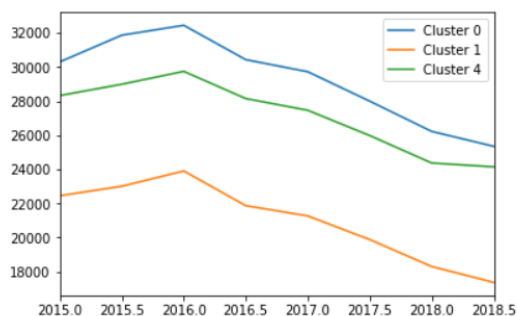
The output is a continuous value and that means it is a regression problem. Linear regression and Polynomial regression are built and evaluated on the data set.

IV. Results:

By drawing the mean of average prices of the three clusters in a run chart it describes how these district clusters average rent price are changing, starting from year 2015 till the end of year 2018:

```
fig, ax = plt.subplots()
clustered_df_c0.mean().plot(kind='line', ax=ax)
clustered_df_c1.mean().plot(kind='line', ax=ax)
clustered_df_c4.mean().plot(kind='line', ax=ax)
ax.legend(['Cluster 0', 'Cluster 1', 'Cluster 4'])
```

<matplotlib.legend.Legend at 0x1729955bf60>



The above figure shows that starting from year 2016 average rent prices started to drop and Cluster 1 average prices are dropping in deeper level than the other two clusters, on the other hand average prices for cluster 4 in year 2018 is becoming more stable than the period before.

To forecast the prices in the first half of year 2018 we need to build a regression model to describe the data set.

Let's try it with Linear regression and see if it serves the purpose:

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

```
model = LinearRegression()
model.fit(x_c0_train, y_c0_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                 normalize=False)
```

```
print( "R-squared: ", model.score(x_c0_train,y_c0_train) )
print( "Coefficient: " , model.coef_)
print( 'Intercept: ',model.intercept_)
```

```
R-squared: 0.6582018812031676
Coefficient: [[-1621.29402699]]
Intercept: [3299193.33558416]
```

```
yhat = model.predict(x_c0_test)
print( 'predicted average price (Cluster 0, 2018.5): ', yhat )
print( 'Actual average price: (Cluster 0, 2018.5)', y_test.values )
```

```
predicted average price (Cluster 0, 2018.5): [[26611.34209889]]
Actual average price: (Cluster 0, 2018.5) [[25347.17391304]]
```

Using linear regression, the predicted price for cluster 0 for 2018 H2 is 26611 SAR while the actual is 25347.

The evaluation of the model is calculated as in the below:

```
from sklearn.metrics import r2_score

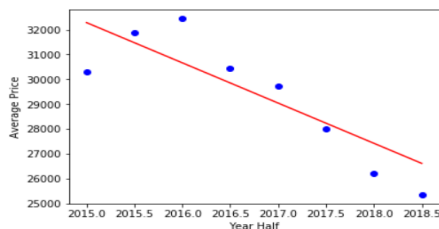
test_x = np.asarray(x_c0_test)
test_y = np.asarray(y_c0_test)
test_y_hat = model.predict(test_x)

print(test_x)
print(test_y)
print(test_y_hat)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_hat - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_hat - test_y) ** 2))
#print("R2-score: %.2f" % r2_score(test_y_hat , test_y))

[[2018.5]]
[[25347.17391304]]
[[26611.34209889]]
Mean absolute error: 1264.17
Residual sum of squares (MSE): 1598121.20
```

Drawing this small data set against the linear regression model equation tells us that our data is non-linear and building a polynomial model may produce results that are more accurate:

```
: plt.scatter(clustered_df_c0_T[['YearHalf']], clustered_df_c0_T[['AveragePrice']], color='blue')
plt.plot(clustered_df_c0_T[['YearHalf']], model.coef_[0][0]*clustered_df_c0_T[['YearHalf']] + model.intercept_[0], '-r')
plt.xlabel("Year Half")
plt.ylabel("Average Price")
: Text(0, 0.5, 'Average Price')
```



Polynomial regression model is built and produced a very good result:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

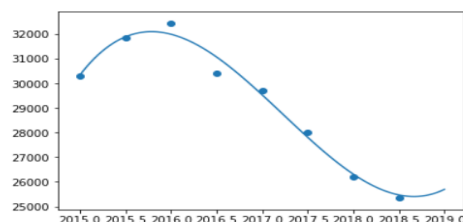
model = make_pipeline(PolynomialFeatures(degree=3), LinearRegression())

x = np.asarray(x_c0_train)
y = np.asarray(y_c0_train)

model.fit(x, y);

# Plotting
xAll = np.asarray(clustered_df_c0_T[['YearHalf']])
yAll = np.asarray(clustered_df_c0_T[['AveragePrice']])
xSpace = np.linspace(min(xAll), max(xAll)+0.5, 100)
plt.plot(xSpace, model.predict(xSpace))
plt.scatter(xAll, yAll)
```

<matplotlib.collections.PathCollection at 0x1729b471f60>



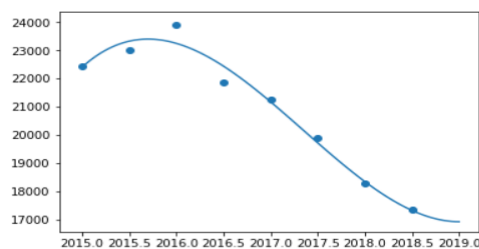
Cluster 0 model evaluation results:

```
test_x = np.asarray(x_c0_test)
test_y = np.asarray(y_c0_test)
test_y_hat = model.predict(test_x)

print(test_x)
print(test_y)
print(test_y_hat)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_hat - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_hat - test_y) ** 2))
#print("R2-score: %.2f" % r2_score(test_y_hat, test_y))

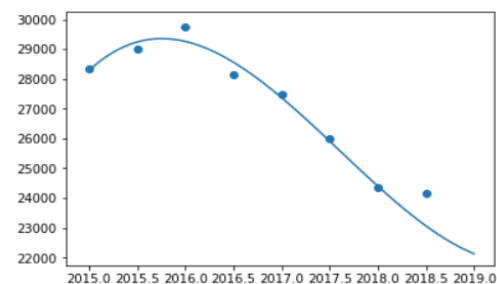
[[2018.5]]
[[25347.17391304]]
[[25471.77148438]]
Mean absolute error: 124.60
Residual sum of squares (MSE): 15524.55
```

The same modeling method is applied to cluster 1 and produced an excellent result:



```
[[2018.5]]
[[17366.60606061]]
[[17329.88818359]]
Mean absolute error: 36.72
Residual sum of squares (MSE): 1348.20
```

Polynomial regression model is built again for cluster 4 but this time the model didn't accurately forecast second half of 2018 (which is our test data point).



```
[[2018.5]]
[[24143.28571429]]
[[23064.66967773]]
Mean absolute error: 1078.62
Residual sum of squares (MSE): 1163412.55
```

As a final result after building our models the average prices of residential units in cluster 1 and cluster 4 are expected continue dropping, but for cluster 0 it is expected start raising during 2019.

V. Discussion:

From graphs in the results section of this report we can see that average rent prices for the residential units in Riyadh city is following a 3rd order polynomial regression model and we relied on it to forecast the average rent prices for the first half in 2019 but that doesn't mean these prices will continue following the same model forever.

In this project i tried to simplify the problem by grouping the city districts into clusters and then calculate the mean of the average rent prices for the clusters before applying machine learning algorithms on the data set, however this approach means that more accurate models can be built by applying machine learning algorithms directly with other methodologies to be applied on the whole data set.

As an advantage to simplifying the data set we were able to understand the data much better and get more insights from it.

An example of insights from the data set is that cluster 0 have Cafés and Hotels as most common venues and let's remember that cluster 0 have the highest average rent rates, while cluster 1 which has the lowest average rent rates have fast food restaurants and middle eastern restaurants as most common near-by venues, and cluster 4 has Cafés and dessert shops as most common venues. And based on this insight we can use it to know places where a certain venue category can be recommended to an investor.

Adding other features such as yearly population in Riyadh city can enhance the models accuracy, but data wasn't publicly available.

VI. Conclusion:

The average residential unit rent prices are forecasted to continue dropping in many districts in Riyadh city during 2019 H1 but with a slower decrease until it reaches a stability point.

Audience of this report can benefit from it to make related decisions such as buying or selling a residential unit in Riyadh city, or an occupant planning to move to a new location inside Riyadh city and so on.