FINAL CAPSTONE PROJECT:

"Finding the Best House"

Gabriel Villa

August 2020

Introduction

Background

Finding a house to buy may be an easy task if you know your needs exactly. But, sometimes you don't have the correct advice, you didn't use the best tools, or simply you don't know the path to explore the market before asking for a deal. It could be so complicated to find it, or even worse, you could buy the wrong option.

Problem

A family plans to buy a house, they are looking for a 'New Construction', and when the realtor asks them for their needs, these customers said:

> "We need a 'Unifamily' Property with sale 'Price' under $400000, 3 or more 'Beds', 2 or more 'Baths' in Orlando, FL. The 'Property' must be close to the largest number of universities and colleges in Orlando, FL. Plus in the best location for 'Orlando Science High School' and our 'Work Office.'

The objective of this project is to propose a Python code that allows selecting the best options to buy a property, based on the needs of the client, data sets of the real estate market, and geo-located databases.

Interest

Real estate agents would be very interested in accurately choosing a new property for their clients, because they could gain a competitive advantage and better deals. Others may be interested in real estate, such as investors and lenders.

Data Acquisition and Cleaning

Data Sources

The first data source is the customer's needs, with this I have the parameters to search the property data set and the venues data set, also the keys to filter and select the options to propose.
This parameters are:

Location: Orlando, FL
Type of Properties: “Unifamily”
Price of Sale: less than $400000
Beds: from 3
Baths: from 2
New Construction: “Yes”
Near to: Work=“DownTown”, High=“Orlando Science High School”
Venues Required: “College”, “University”

The second source is a Realty API. The credentials to use it were obtained under the license of rapidapi.com to use the API from realtor.com; this is an API that responds with a string of XML data. Each string must be converted to a JSON object to access, index, format, and filter the required information. Furthermore, based on the restrictions on this service, a limit of 200 properties on the application within a 5 miles radius has been set. The result is a Pandas DataFrame with six columns ["Address", "Beds", "Bathrooms", "Price", "Classification", "New Construction"], where:

"Address" values are dictionaries where each element has a 1st line, city, state, etc.
"Beds", **"Bathrooms"**, **"Price"**, and **"Rating"** are integer values.
"New construction" is a Boolean.

The third source is the Foursquare API. The credentials to use it were obtained from the Foursquare developer page. This is an API that responds with a JSON object that must be converted into a dictionary to access, index, format, and filter the required information. Following the restrictions of this service, a limit of 100 venues in a radius of 6000 meters has been set. The result is a large dictionary with two main tags ("College" and "University"). Those are the keys that are used in the API request for each property.

Data Cleaning

When the property data is requested, the "Address" value found is a dictionary with these fields: First Line, City, State Code, and Postal Code. All this information has been merged to have the correct format.

To cluster the property data, the "Mean Distance" from the "High School" and the "Work Office" to each property have been used. Once this is calculated, this data is normalized and a new column called "Mean Dist" is added. Besides, that column needs to be used to generate the five property clusters. Then each property needs to be labeled by adding a new column called "Cluster labels".

To complete the venue data process. First, the best cluster of properties must be selected based on the smallest mean distance. Furthermore, before requesting the venue's data, It must create a list of properties. Then, it created a new DataFrame from the Venues Dictionary using these fields: Name, Address, Latitude, Longitude, and Category. After that, the venues for each property are counted and create a properties list with the largest count.

Method

Step 1

First, a realty API needs to be used to find a 'Unifamily' for-sale property listing in the Orlando Florida area. They need to have three or more Beds and two or more Baths. When it gets the list, it must be filtered to choose properties with a price lower than $ 400000. For that, the Property DataFrame must be checked with *Pandas*, and the result lists display with *Folium* maps.

Step 2

In the second step, according to the 'Mean Distance' from 'High School' and 'Work' to each 'Property' the DataFrame needs to be clustered with "K-means". Also, I select the 'Best Cluster' and find a second properties list. I am going to review the Properties' DataFrame with "Pandas" and visualize the result lists with "Folium" maps.

Step 3

In third place, the 'Venues' around each 'Property' must be analyzed, by using the "Foursquare API" to find the Universities and Colleges. The end of this process is a third property list with the most count of 'Venues' selected to look for a deal. Properties DataFrame was reviewed with *Pandas*, and the result lists visualize with *Folium* maps.

Step 4

In this part, the outcomes have to be formatted, presented, visualized, and saved. For which, *Pandas* and *Folium* have to be used.

Note: Have been used CSV files to save each step dataset results (in my Google Drive).

Results

In this section can be found the outcomes step by step. Here data descriptive, statistics, and visualizations have been presented.

Step 1

Ones set the parameters to make the request, formatted the Realty API response, and created the first DataFrame. Note the Address field has dictionaries with the info without the correct format.

```
data_1.head()
```

```
This is a first DataFrame (Note: Have a Dict with the Address information.)
```

	Address	Beds	Baths	Price	Rank	New Construction
0	{'city': 'Orlando', 'line': '9250 Neher St', '...	5	5	725230	1	True
1	{'line': '3105 Eagle Hammock Cir.', 'city': 'K...	4	3	317990	2	True
2	{'line': '12048 Alder Branch Loop', 'city': 'O...	3	3	223900	3	True
3	{'line': '8904 Tavistock Lakes Boulevard', 'ci...	4	4	502990	4	True
4	{'line': '5600 Barletta Drive', 'city': 'Saint...	4	2	364990	5	True

The Address field was fixed using Pandas by extracting each dictionary and concatenating the string values in a single for each property. Then, in the dictionary place, the results were saved.

```
data_1.head()
```

```
This is a first DataFrame (Note: With the Addresses and Coords information)
```

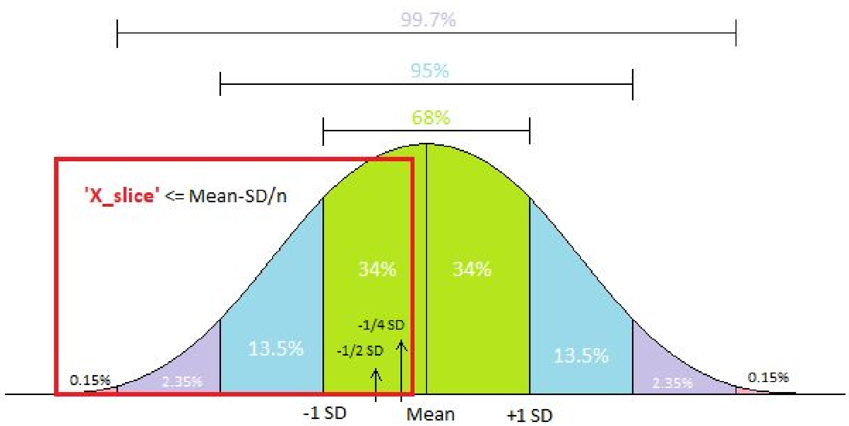
	Address	Beds	Baths	Price	Rank	New Construction	lat	lon
0	9250 Neher St. Orlando,FL 32827	5	5	725230	1	True	28.3684	-81.2511
1	3105 Eagle Hammock Cir.. Kissimmee,FL 34743	4	3	317990	2	True	28.3407	-81.3392
2	12048 Alder Branch Loop. Orlando,FL 32824	3	3	223900	3	True	28.3962	-81.3474
3	8904 Tavistock Lakes Boulevard. Orlando,FL 32827	4	4	502990	4	True	28.3721	-81.2548
4	5600 Barletta Drive. Saint Cloud,FL 34771	4	2	364990	5	True	28.3336	-81.2173

With the property list, then their stats can be found. The Price values present a huge dispersion because the mean is $517496.93 and less than its standard deviation of $582609.93. In consequence, this data set must be filtered by the Price value to have a data set under the goal of $400000.

```
This are the stats for the first properties data set
```

	mean	std	max	min
Beds	3.770000	0.720901	6.000000e+00	3.000000
Baths	3.150000	1.011243	9.000000e+00	2.000000
Price	517496.930000	582609.929503	7.000000e+06	199000.000000
Rank	20.950000	12.210363	4.300000e+01	1.000000
lat	28.415146	0.086984	2.866546e+01	28.289717
lon	-81.302181	0.094818	-8.118553e+01	-81.545086

To have a dataset filtered according to the stats, has been sliced under the mean value minus a standard deviation fraction, tested to produce the highest max value under the goal of $400000. Like can be shown in the next figure.



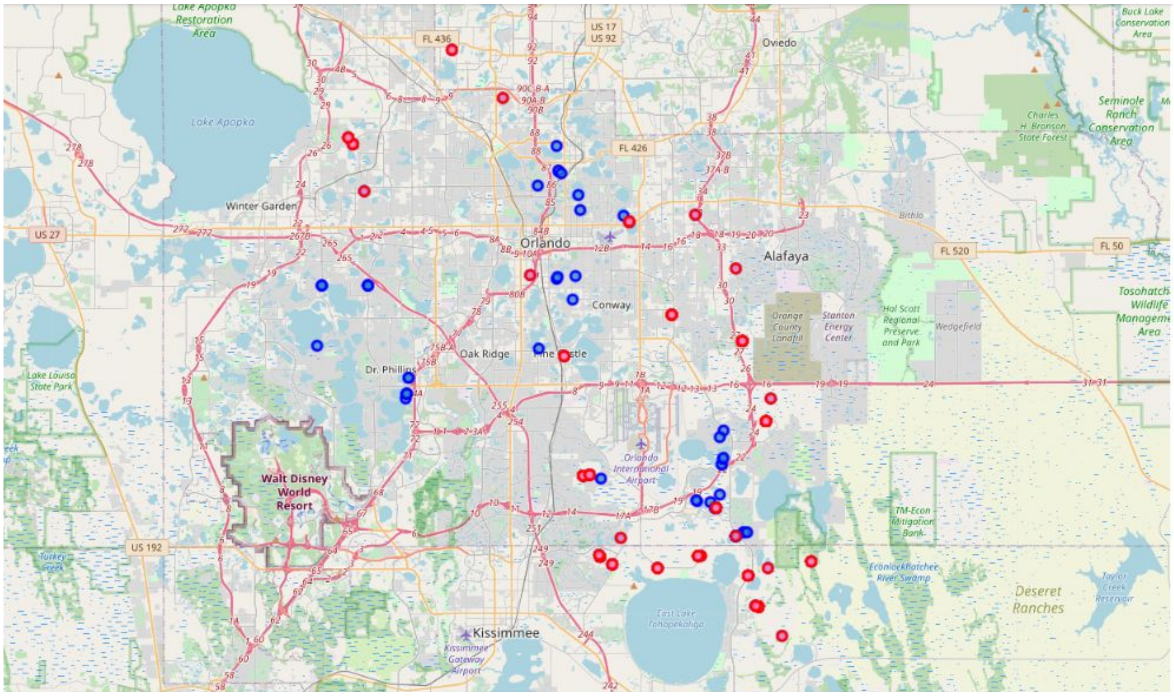
After that, there are the new stats for the property list sliced. The Price values present a little dispersion, because the mean is $334409.81 and its standard deviation of $35150.04, which is 10.5% approx. In consequence, this data set has the Price values spread under the goal.

```
data_1_stats
```

```
This properties data set is the first step result, showing the stats for 1/5 std under the mean.
```

	mean	std	max	min
Beds	3.545455	0.615345	6.000000	3.000000
Baths	2.627273	0.588250	4.000000	2.000000
Price	334409.809091	35150.045279	399990.000000	199000.000000
Rank	19.872727	12.257265	43.000000	1.000000
lat	28.411053	0.096318	28.665457	28.289717
lon	-81.292137	0.086024	-81.185532	-81.522500

Finally, once the Property DataFrame is checked, the result lists are displayed with the Folium maps library. Where the Reds are the properties to be selected under the Price goal.



Step 2

First, using the mean distance equation, I found the mean distance from each property to the high school and the work office. Where "x" is for "Latitude" and "y" is for "Longitude". The values in red are for "Property", the blue ones are for "High School" and "Work Office".

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Then, the mean values for both have been found and store them in a new column called "Mean Dist".

```
data_1_sliced.head()
```

	Address	Beds	Baths	Price	Rank	New Construction	lat	lon	Mean Dist
0	1618 Lake Sims Parkway. Ocoee,FL 34761	4	3	308990	1	True	28.6051	-81.5193	0.99
1	1618 Lake Sims Parkway. Ocoee,FL 34761	5	3	316990	2	True	28.6051	-81.5193	0.99
2	1618 Lake Sims Parkway. Ocoee,FL 34761	5	3	321990	3	True	28.6051	-81.5193	0.99
3	1618 Lake Sims Parkway. Ocoee,FL 34761	3	2	293990	4	True	28.6051	-81.5193	0.99
6	3105 Eagle Hammock Cir.. Kissimmee,FL 34743	3	3	306990	7	True	28.3407	-81.3392	0.78

The next is clustering the dataset, to find five clusters, the Mean Dist column only was used. Labels for each cluster were generated and a new column named Cluster Labels to tag each property.

```
[ ] # set number of clusters
    kclusters = 5

    c=['Address','Beds','Baths','Price','Rank','New Construction','lat','lon']
    data_1_sliced_clustering = data_1_sliced.drop(columns=c)

    # run k-means clustering
    kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(data_1_sliced_clustering)

    # check cluster labels generated for each row in the dataframe
    kmeans.labels_[0:10]

    array([2, 2, 2, 2, 4, 4, 4, 3, 0, 3], dtype=int32)
```

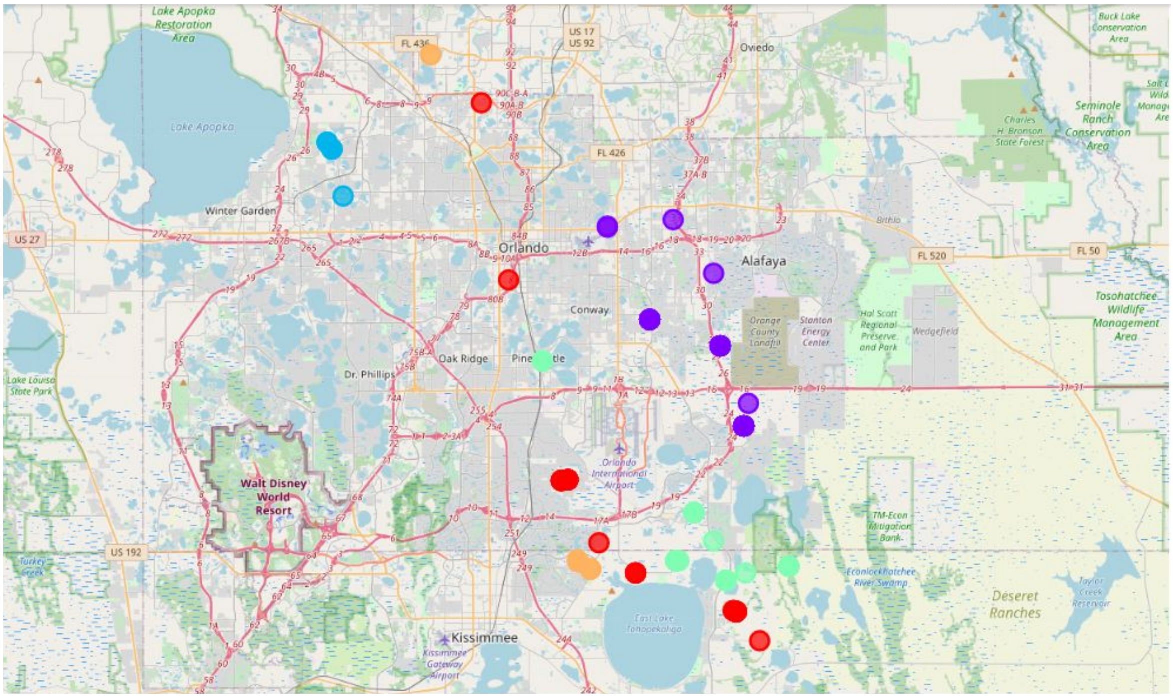
```
[ ] # add clustering labels to each property
    data_1_sliced['Cluster Labels']=kmeans.labels_
    col_list=data_1_sliced.columns.tolist()
    new_list=list(range(len(col_list)))
    for n in list(range(len(col_list))):
      new_list[n]=col_list[n-1]
    new_list
    data_1_sliced=data_1_sliced[new_list]
```

The DataFrame below this line shows the result for this process:

```
[ ] data_1_sliced.head() # check the last columns!
```

	Cluster Labels	Address	Beds	Baths	Price	Rank	New Construction	lat	lon	Mean Dist
0	2	1618 Lake Sims Parkway. Ocoee,FL 34761	4	3	308990	1	True	28.6051	-81.5193	0.99
1	2	1618 Lake Sims Parkway. Ocoee,FL 34761	5	3	316990	2	True	28.6051	-81.5193	0.99
2	2	1618 Lake Sims Parkway. Ocoee,FL 34761	5	3	321990	3	True	28.6051	-81.5193	0.99
3	2	1618 Lake Sims Parkway. Ocoee,FL 34761	3	2	293990	4	True	28.6051	-81.5193	0.99
6	4	3105 Eagle Hammock Cir.. Kissimmee,FL 34743	3	3	306990	7	True	28.3407	-81.3392	0.78

Finally, once the Property DataFrame is checked, the result lists are displayed below these lines with the Folium maps library. Where the properties are shown according to their cluster, the Salmons in Cluster 4, Greens in cluster 3, Light Blues in cluster 2, Purples in cluster 1, and Reds in cluster 0.



To have a better idea about each cluster now each cluster is described showing their stats for all their fields, where the focus is the mean distance value.

How can be checked, the cluster 0 presents a count of 29 properties, with a mean distance value of 0.70 and a standard deviation of 0.01, with a max of 0.73 and a min of 0.67.

```
[ ] # Showing Cluster 0 Estructure
    print('Cluster 0 Stats\n')
    cluster0 = data_1_sliced[data_1_sliced['Cluster Labels']==0]
    cluster0.describe()
```

```
Cluster 0 Stats
```

	Cluster Labels	Beds	Baths	Price	Rank	lat	lon	Mean Dist
count	29.0	29.000000	29.000000	29.000000	29.00000	29.000000	29.000000	29.000000
mean	0.0	3.689655	2.517241	328953.448276	24.00000	28.359721	-81.289798	0.701034
std	0.0	0.541390	0.687682	35708.988442	12.75035	0.073460	0.065202	0.014478
min	0.0	3.000000	2.000000	199000.000000	3.00000	28.289717	-81.409783	0.670000
25%	0.0	3.000000	2.000000	314990.000000	14.00000	28.308548	-81.346810	0.690000
50%	0.0	4.000000	2.000000	333995.000000	24.00000	28.333585	-81.297520	0.700000
75%	0.0	4.000000	3.000000	349990.000000	36.00000	28.393053	-81.223995	0.710000
max	0.0	5.000000	4.000000	386995.000000	43.00000	28.634829	-81.206569	0.730000

For the cluster 1 presents a count of 31 properties, with a mean distance value of 0.40 and a standard deviation of 0.04, with a max of 0.48 and a min of 0.30.

```
[ ] # Showing Cluster 1 Estructure
    print('Cluster 1 Stats\n')
    data_1_sliced[data_1_sliced['Cluster Labels']==1].describe()
```

Cluster 1 Stats

	Cluster Labels	Beds	Baths	Price	Rank	lat	lon	Mean Dist
count	31.0	31.000000	31.000000	31.000000	31.000000	31.000000	31.000000	31.000000
mean	1.0	3.516129	2.677419	336901.290323	18.161290	28.470762	-81.246496	0.397097
std	0.0	0.724383	0.599283	31866.892301	11.602002	0.042081	0.033393	0.043680
min	1.0	3.000000	2.000000	256000.000000	4.000000	28.427700	-81.318177	0.300000
25%	1.0	3.000000	2.000000	318990.000000	8.500000	28.427700	-81.286881	0.350000
50%	1.0	3.000000	3.000000	339990.000000	15.000000	28.479141	-81.235528	0.410000
75%	1.0	4.000000	3.000000	358990.000000	26.500000	28.495640	-81.218600	0.430000
max	1.0	6.000000	4.000000	395990.000000	43.000000	28.559794	-81.214789	0.480000

For the cluster 2 presents a count of 8 properties, with a mean distance value of 0.99 and a standard deviation of 0.01, with a max of 1.00 and a min of 0.96.

```
[ ] # Showing Cluster 2 Estructure
    print('Cluster 2 Stats\n')
    data_1_sliced[data_1_sliced['Cluster Labels']==2].describe()
```

Cluster 2 Stats

	Cluster Labels	Beds	Baths	Price	Rank	lat	lon	Mean Dist
count	8.0	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000
mean	2.0	3.875000	3.000000	349865.000000	14.000000	28.602855	-81.519447	0.990000
std	0.0	0.834523	0.534522	44309.584581	14.392458	0.011372	0.003808	0.013093
min	2.0	3.000000	2.000000	293990.000000	1.000000	28.575140	-81.522500	0.960000
25%	2.0	3.000000	3.000000	314990.000000	2.750000	28.605100	-81.522500	0.990000
50%	2.0	4.000000	3.000000	342740.000000	9.000000	28.605100	-81.519300	0.990000
75%	2.0	4.250000	3.000000	396615.000000	21.500000	28.609100	-81.519300	1.000000
max	2.0	5.000000	4.000000	399990.000000	38.000000	28.609100	-81.510876	1.000000

For the cluster 3 presents a count of 29 properties, with a mean distance value of 0.63 and a standard deviation of 0.03, with a max of 0.66 and a min of 0.58.

```
[ ] # Showing Cluster 3 Estructure
    print('Cluster 3 Stats\n')
    data_1_sliced[data_1_sliced['Cluster Labels']==3].describe()
```

Cluster 3 Stats

	Cluster Labels	Beds	Baths	Price	Rank	lat	lon	Mean Dist
count	29.0	29.000000	29.000000	29.000000	29.000000	29.000000	29.000000	29.000000
mean	3.0	3.379310	2.620690	339546.206897	19.896552	28.355460	-81.253233	0.631379
std	0.0	0.493804	0.493804	34008.485644	11.693542	0.041610	0.047288	0.031929
min	3.0	3.000000	2.000000	278990.000000	2.000000	28.328749	-81.365438	0.580000
25%	3.0	3.000000	2.000000	309000.000000	11.000000	28.333583	-81.267612	0.590000
50%	3.0	3.000000	3.000000	339990.000000	16.000000	28.341523	-81.254840	0.650000
75%	3.0	4.000000	3.000000	372990.000000	30.000000	28.354238	-81.231623	0.660000
max	3.0	4.000000	3.000000	396990.000000	42.000000	28.469219	-81.185532	0.660000

For the cluster 4 presents a count of 13 properties, with a mean distance value of 0.78 and a standard deviation of 0.02, with a max of 0.84 and a min of 0.77.

```
[ ] # Showing Cluster 4 Estructure
    print('Cluster 4 Stats\n')
    data_1_sliced[data_1_sliced['Cluster Labels']==4].describe()
```

Cluster 4 Stats

	Cluster Labels	Beds	Baths	Price	Rank	lat	lon	Mean Dist
count	13.0	13.000000	13.000000	13.000000	13.000000	13.000000	13.000000	13.000000
mean	4.0	3.461538	2.538462	319671.461538	18.307692	28.389166	-81.353098	0.785385
std	0.0	0.518875	0.518875	36309.375572	11.600177	0.122645	0.041951	0.024703
min	4.0	3.000000	2.000000	267990.000000	4.000000	28.335548	-81.447361	0.770000
25%	4.0	3.000000	2.000000	291990.000000	8.000000	28.335548	-81.339196	0.770000
50%	4.0	3.000000	3.000000	308829.000000	15.000000	28.340735	-81.339196	0.780000
75%	4.0	4.000000	3.000000	348990.000000	25.000000	28.340735	-81.330225	0.780000
max	4.0	4.000000	3.000000	375000.000000	38.000000	28.665457	-81.330225	0.840000

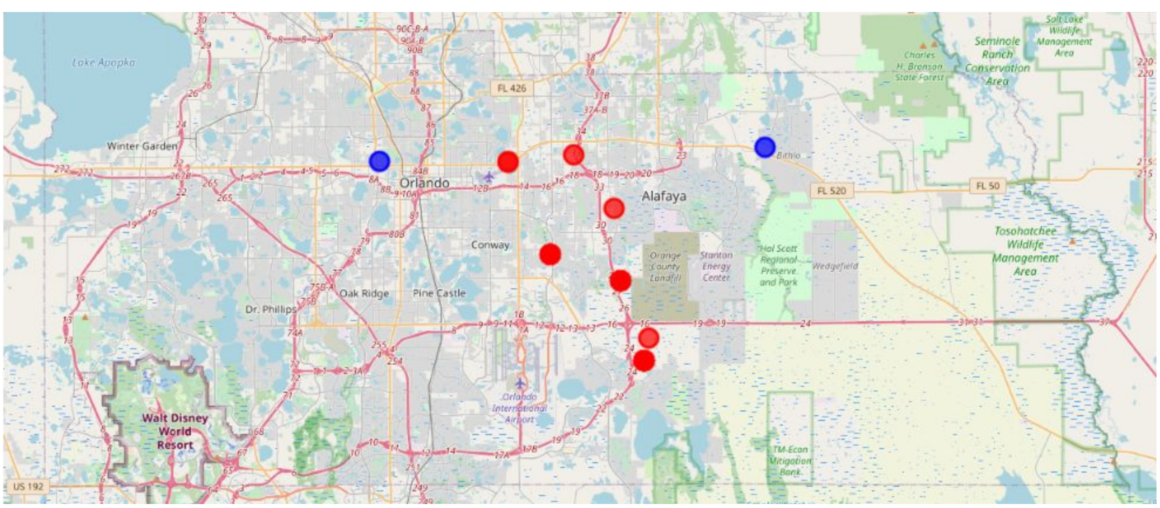
After comparing the cluster mean distance, cluster 1 was selected, a new DataFrame named data_2 was created, and part of it is shown below these lines.

```
The Best Cluster is #1 with a Mean Distance of 0.40
```

```
[ ] data_2 = data_1_sliced[data_1_sliced['Cluster Labels']==cluster]
    data_2
```

	Cluster Labels	Address	Beds	Baths	Price	Rank	New Construction	lat	lon	Mean Dist
26	1	5009 Santa Rosa Dr. Orlando,FL 32807	3	2	289000	27	True	28.555226	-81.318177	0.48
27	1	5015 Santa Rosa Dr. Orlando,FL 32807	3	2	289000	28	True	28.555280	-81.317985	0.48
35	1	10914 History Avenue. Orlando,FL 32832	4	3	395990	36	True	28.427700	-81.218600	0.41
42	1	10914 History Avenue. Orlando,FL 32832	4	3	373990	43	True	28.427700	-81.218600	0.41
49	1	10914 History Avenue. Orlando,FL 32832	3	2	362990	7	True	28.427700	-81.218600	0.41
54	1	5973 Wooden Pine Drive. Orlando,FL 32829	4	3	358990	12	True	28.479141	-81.235528	0.34
55	1	5973 Wooden Pine Drive. Orlando,FL 32829	3	2	331990	13	True	28.479141	-81.235528	0.34
56	1	5973 Wooden Pine Drive. Orlando,FL 32829	4	2	339990	14	True	28.479141	-81.235528	0.34
57	1	5973 Wooden Pine Drive. Orlando,FL 32829	3	2	322990	15	True	28.479141	-81.235528	0.34
58	1	5973 Wooden Pine Drive. Orlando,FL 32829	4	3	378990	16	True	28.479141	-81.235528	0.34
59	1	5973 Wooden Pine Drive. Orlando,FL 32829	3	2	348990	17	True	28.479141	-81.235528	0.34
60	1	10914 History Avenue. Orlando,FL 32832	3	2	341990	18	True	28.427700	-81.218600	0.41
70	1	10914 History Avenue. Orlando,FL 32832	6	3	370990	28	True	28.427700	-81.218600	0.41
75	1	10914 History Avenue. Orlando,FL 32832	4	4	352990	33	True	28.427700	-81.218600	0.41
84	1	8525 Blackberry Ave. Orlando,FL 32825	3	2	256000	42	True	28.559794	-81.270205	0.36
89	1	10914 History Avenue. Orlando,FL 32832	5	3	358990	4	True	28.427700	-81.218600	0.41
92	1	11932 Landing Point Loop. Orlando,FL 32832	3	3	312990	7	True	28.441842	-81.214789	0.38

Finally, once the new Property DataFrame was created, the result lists are displayed below these lines with the Folium maps library. Where the properties are shown in Red, and the Blue are the High School and the Work Office.



Step 3

Once the cluster was selected, a list of properties can be known. In this step, the first process found the venues per each property in the list using the keywords “College” and “University”.

To look for the venues the Foursquare API was used, creating a dictionary with the features available for all of them. The code below these lines was used for this job.

```
[ ] # creating the venues dictionary
    venues={}
    properties = data_2.index.values.tolist()
    i=0
    for n in ['University', 'College']:
      # getting the group DataFrames
      x = getNearbyVenues(names=n,latitudes=data_2['lat'],longitudes=data_2['lon'])
      #
      print('For {} have {} Venues'.format(n,x.shape[0]))
      venues[n]=x # saving the vanues in a dictionary
      i=i+1
    #
```

```
For University have 371 Venues
For College have 284 Venues
```

A total of 655 venues near the properties were found using the Foursquare API. By using the keyword University 371 venues, and 284 using the keyword College. But, some of them only use these words in their names or description, which is required to filter the venues by their category ID to be sure that it is a College or University.

Then, per each property in the list, the Colleges and Universities near them can be counted. Following that, a list of properties with the max count of venues can be made.

```
[ ] # Filtering and Counting the Venues by CategoryID (Colleges and Universities)
    venues_college = venues['College'].loc[venues['College']['Venue Category'].isin(['University',
                                                          'College & University'])].groupby('propertyID').count()
    venues_university = venues['University'].loc[venues['University']['Venue Category'].isin(['University',
                                                              'College & University'])].groupby('propertyID').count()

    # Finding the max venues count by categoryID in Venues Data set
    venues_total = round((venues_college + venues_university)/2)
    max_venues = venues_total.max()['Venue']

    # Getting the list of propertiesID with the max Venues count
    properties_max = venues_total[venues_total['Venue']==max_venues].index.tolist()

    # Filtering from Poperties Data Set by PropertyID
    data_3 = data_2.loc[properties_max]

    data_3
```

	Cluster Labels	Address	Beds	Baths	Price	Rank	New Construction	lat	lon	Mean Dist
26	1	5009 Santa Rosa Dr. Orlando,FL 32807	3	2	289000	27	True	28.555226	-81.318177	0.48
27	1	5015 Santa Rosa Dr. Orlando,FL 32807	3	2	289000	28	True	28.555280	-81.317985	0.48

Step 4

With the final list of properties known, now the results need to be formatted and visualized. In the same order, a list of the nearest Colleges and Universities can be found. This process can be checked in the code below these lines.

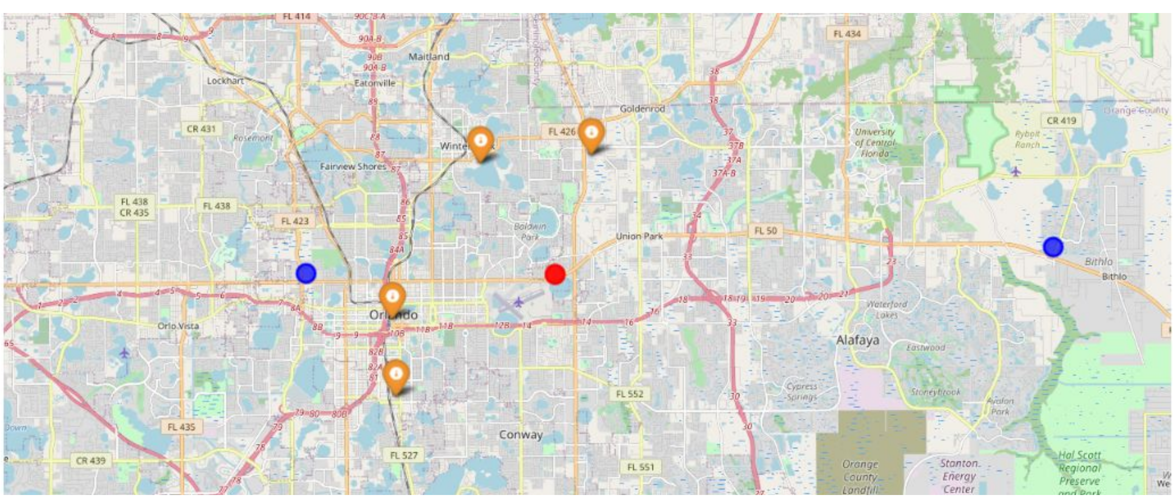
```
# Getting the Colleges & Universities List Near to the Best Properties
venues_final = venues['University'].loc[venues['University']['Venue Category'].isin(['University','College & University'])]
venues_final_list = venues_final[venues_final['propertyID']==data_3.index.values[0]]['Venue'].tolist()

# Filtering Properties Data set
data_top = data_3.reset_index()
data_top.drop(columns=['Cluster Labels','lat','lon','Rank','New Construction','Mean Dist'], inplace=True)

# Getting the Final Result Properties List
results_final = pd.DataFrame()
results_final[['Address','Beds','Baths','Price']]=data_top[['Address','Beds','Baths','Price']]

print('Properties List from the Best Cluster\n Max number of Universities and Colleges = {}'.format(max_venues))
print(' {}\n'.format(venues_final_list))
results_final
```

```
Properties List from the Best Cluster
 Max number of Universities and Colleges = 4.0
 ['Rollins College', 'Full Sail University', 'UCF Downtown Campus', 'National University - Orlando, Florida']
```

	Address	Beds	Baths	Price
0	5009 Santa Rosa Dr. Orlando,FL 32807	3	2	289000
1	5015 Santa Rosa Dr. Orlando,FL 32807	3	2	289000

How was presented in the last results four venues are in the list: “Rollins College”, “Full Sail University”, “UCF Downtown Campus”, and “National University - Orlando, Florida”.

Finally, using the last Property DataFrame, the result lists are displayed below on a Folium map. The venues are shown in the map using an Orange pin, in Blue are the High School and the Work Office, and in Red are the properties proposed like the final result.



Analysis

For both property location coordinate values datasets, between the count and the unique values, there is a ratio of 35% approx (72/200 in first and 37/110 in second). On that, 65% of properties are in big deploy projects where the address is the same, can be inferred.

```
All the Properties are New Constructions, In Orlando Metro Area, 3+ Beds, 2+ Baths
```

	Address	Beds	Baths	Price	Rank	New Construction	lat	lon
count	199	200	200	200	200	200	200.0000	200.0000
unique	71	4	7	163	43	1	72.0000	72.0000

Following the same logic order, there is a Price ratio of 74% approx (163/200 in first and 81/110 in second). Which explains, a 26% of properties have a list price or the same conditions for sale.

```
Properties Filtered 110/200 for 'Price' values bellow $400000
 Sliced With mean-std/5, max = $399990, and mean = $334409.81
```

	Address	Beds	Baths	Price	Rank	New Construction	lat	lon
count	110	110	110	110	110	110	110.0000	110.0000
unique	36	4	3	81	42	1	36.0000	37.0000

Another interesting point is when the price falls below the target ($ 400,000). In this case, the unique values for the baths decrease from 7 to 3. Therefore, it can be inferred more bathrooms increase the value of the property.

Regarding the selected cluster, this presents less than 1% standard deviation from the mean location coordinates (0.042/28.470 for latitude or 0.033/81.246 for longitude), and 11% for the mean distance (0.044/0.397). These can understand that all properties are relatively near and equidistant to the school and work.

```
Cluster 1 Stats
```

	Cluster Labels	Beds	Baths	Price	Rank	lat	lon	Mean Dist
count	31.0	31.000000	31.000000	31.000000	31.000000	31.000000	31.000000	31.000000
mean	1.0	3.516129	2.677419	336901.290323	18.161290	28.470762	-81.246496	0.397097
std	0.0	0.724383	0.599283	31866.892301	11.602002	0.042081	0.033393	0.043680

Finally, using the Foursquare API, 655 total venues had been found. There are 371 with University keyword, and 284 using College keyword. That results in 12 venues average per property (655/31 properties in cluster).

However, In the last property list proposed there are 4 venues only. That means, for the properties in the list, only 33% of the venues are Colleges or Universities.

```
For University have 371 Venues
For College have 284 Venues
```

```
Properties List from the Best Cluster
Max number of Universities and Colleges = 4.0
['Rollins College', 'Full Sail University', 'UCF Downtown Campus', 'National University - Orlando, Florida']
```

	Address	Beds	Baths	Price
0	5009 Santa Rosa Dr. Orlando,FL 32807	3	2	289000
1	5015 Santa Rosa Dr. Orlando,FL 32807	3	2	289000

Conclusions

Once the method was applied and the final properties list found we can arrive at the following affirmations:

- A code using Python can explore the real estate market looking for properties by a Realtor API.
- To clustering a property dataset, "mean distance" can be used.
- A code using Python can use the Foursquare API looking for venues near to addresses in a dataset.
- To filter and visualize property datasets, "Pandas and Folium libraries" can be used.

Recommendations

Because it can reduce their response time, Real estate agents can be more efficient in advising their clients using this method. This process could be generalized, so this is a line to explore in future projects.

The nature of the realty market and the geo-located databases like Foursquare change dynamically. The same case of study can be followed in the time to generate performance models to the method proposed in this paper.

In the customer needs and the data nature only, this study was focused. But, a lot of different variables can affect the value of a property. New variables like the age of construction, type of construction, or the property tax history, for example, could be part of the analysis in futures studies.