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.

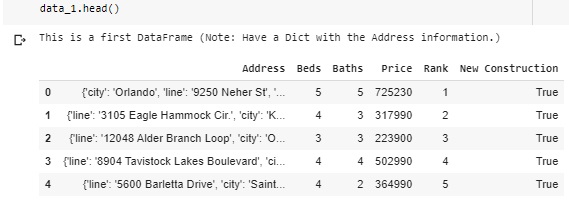
*Note: Have been used CSV files to save each step dataset results (in my Google Drive).*

## Results

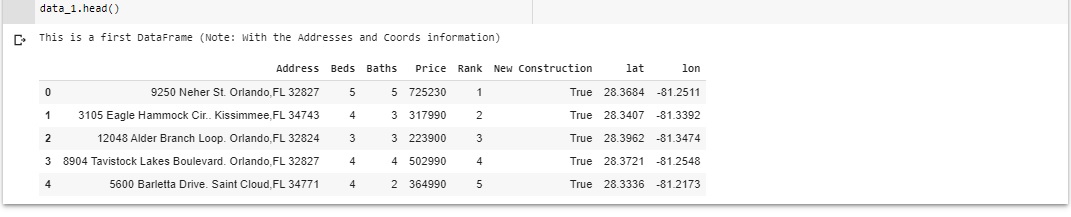
In this section can be found the outcomes step by step. Here data descriptive, statistics, and visualizations have been presented.

### Step 1

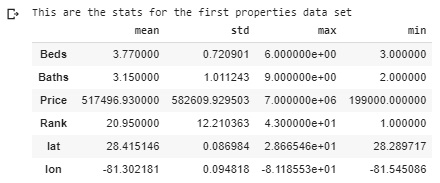
Ones set the parameters to make the request, formatted the Realty API response, and created the first DataFrame. Note the Address field has dictionaries with the info without the correct format.



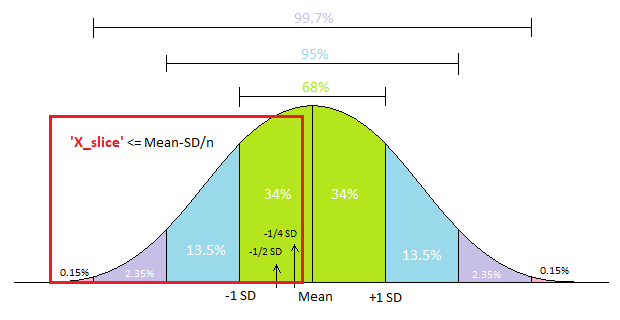
The Address field was fixed using Pandas by extracting each dictionary and concatenating the string values in a single for each property. Then, in the dictionary place, the results were saved.



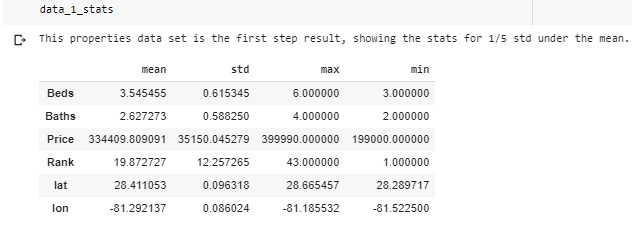
With the property list, then their stats can be found. The Price values present a huge dispersion because the mean is $517496.93 and less than its standard deviation of $582609.93. In consequence, this data set must be filtered by the Price value to have a data set under the goal of $400000.



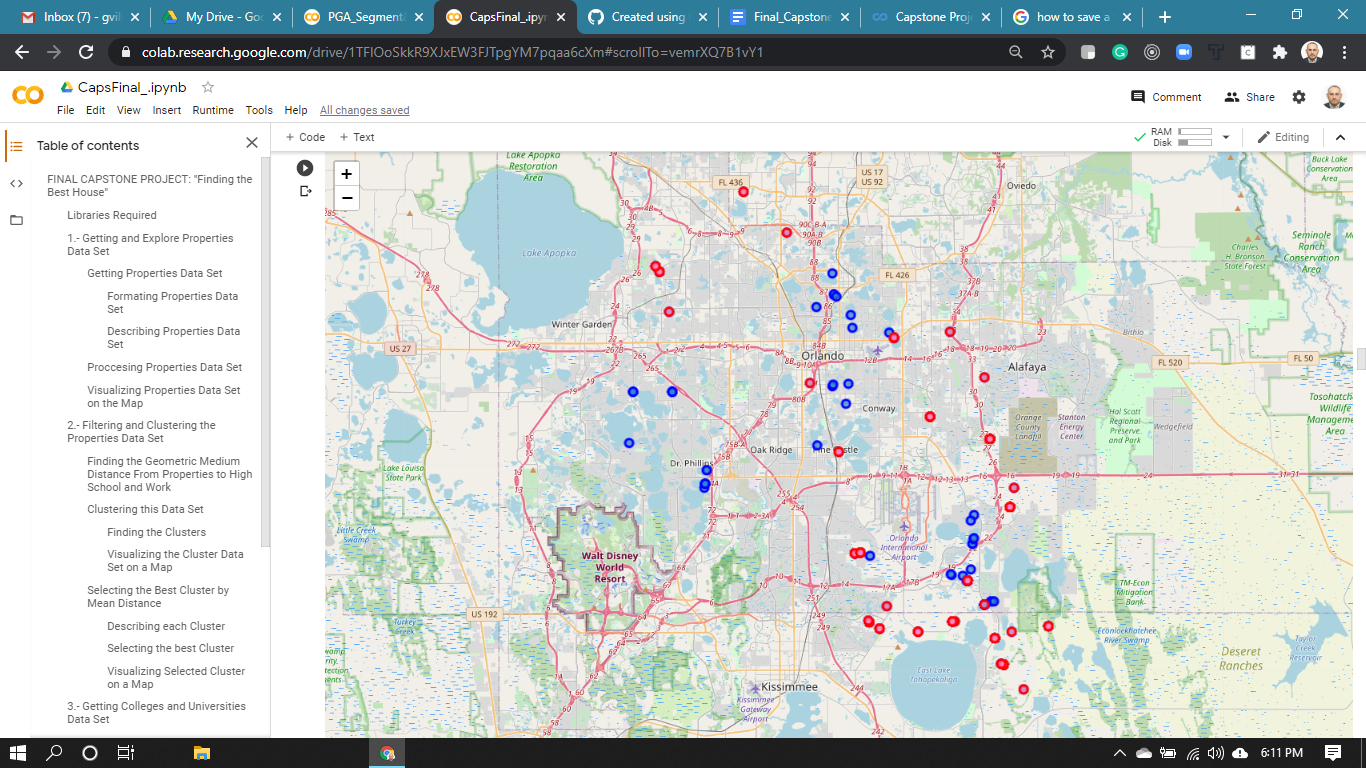
To have a dataset filtered according to the stats, has been sliced under the mean value minus a standard deviation fraction, tested to produce the highest max value under the goal of $400000. Like can be shown in the next figure.



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.

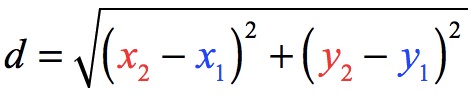


Finally, once the Property DataFrame is checked, the result lists are displayed with the Folium maps library. Where the Reds are the properties to be selected under the Price goal.



### 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".



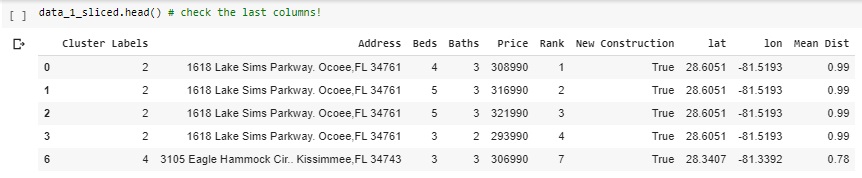
Then, the mean values for both have been found and store them in a new column called "Mean Dist".



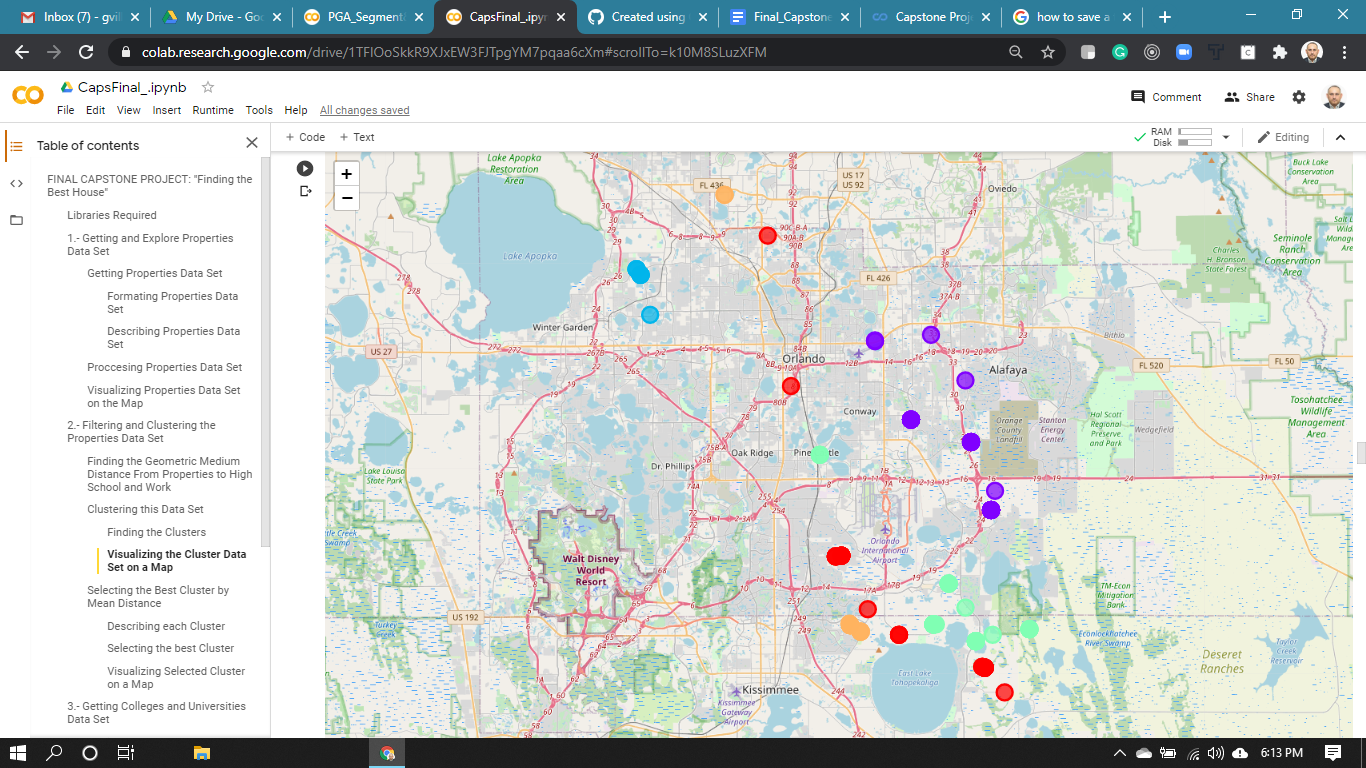
The next is clustering the dataset, to find five clusters, the Mean Dist column only was used. Labels for each cluster were generated and a new column named Cluster Labels to tag each property.



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



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.

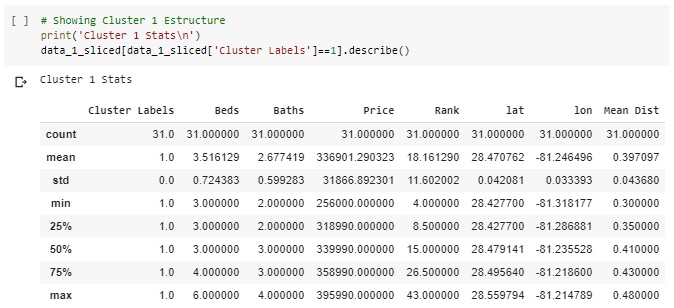


To have a better idea about each cluster now each cluster is described showing their stats for all their fields, where the focus is the mean distance value.

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.



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.



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.



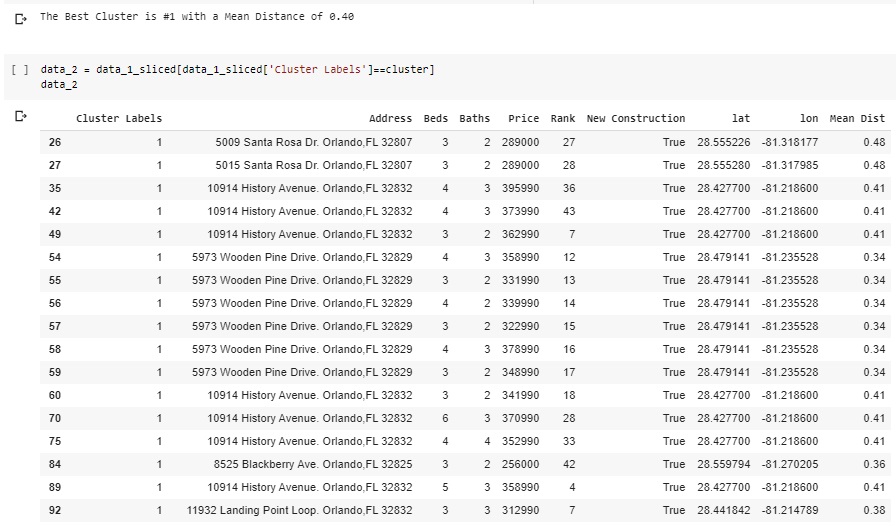
For the cluster 3 presents a count of 29 properties, with a mean distance value of 0.63 and a standard deviation of 0.03, with a max of 0.66 and a min of 0.58.



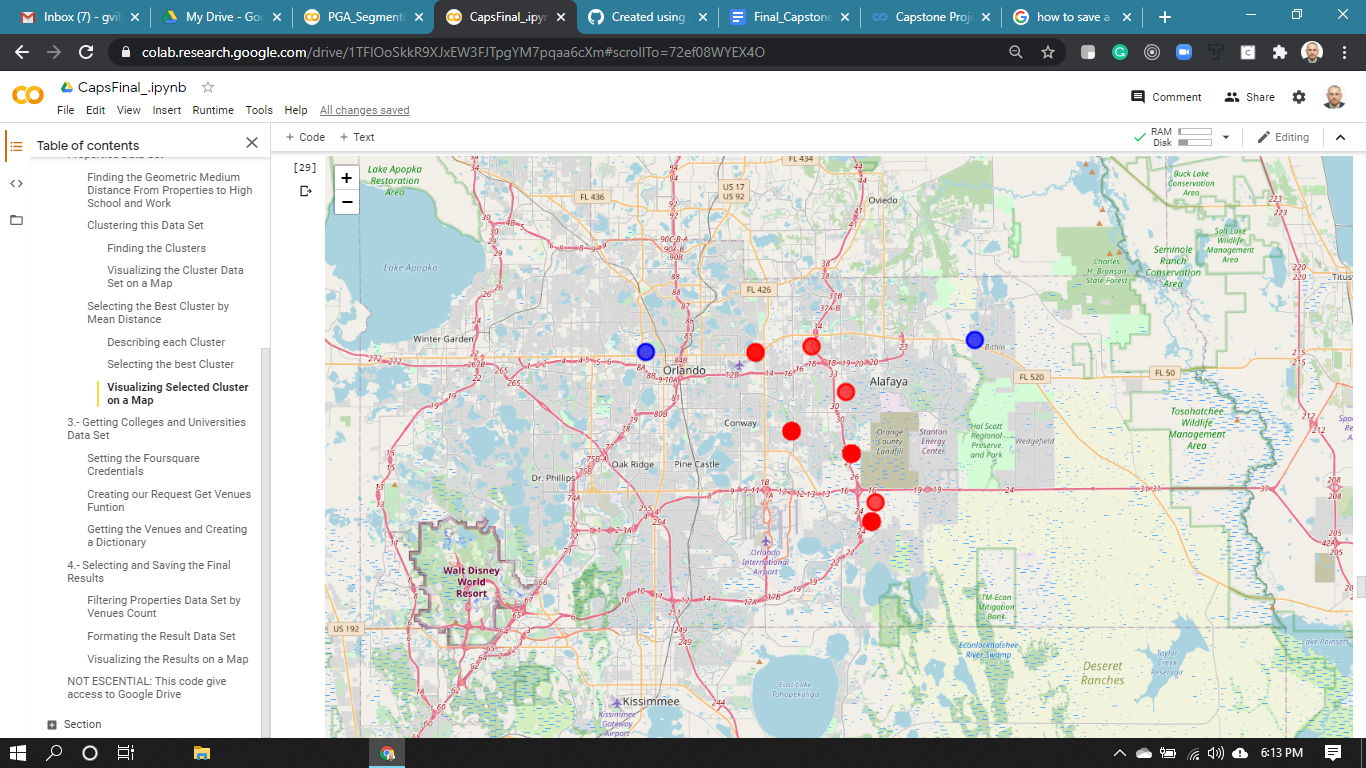
For the cluster 4 presents a count of 13 properties, with a mean distance value of 0.78 and a standard deviation of 0.02, with a max of 0.84 and a min of 0.77.



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.



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

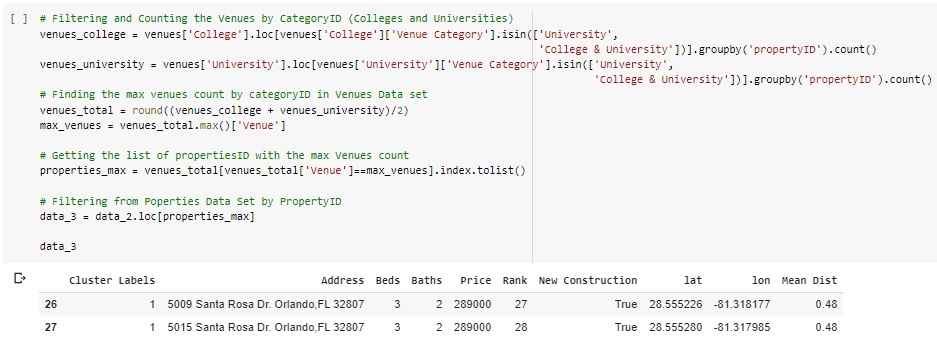
Once the cluster was selected, a list of properties can be known. In this step, the first process found the venues per each property in the list using the keywords “College” and “University”.

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.



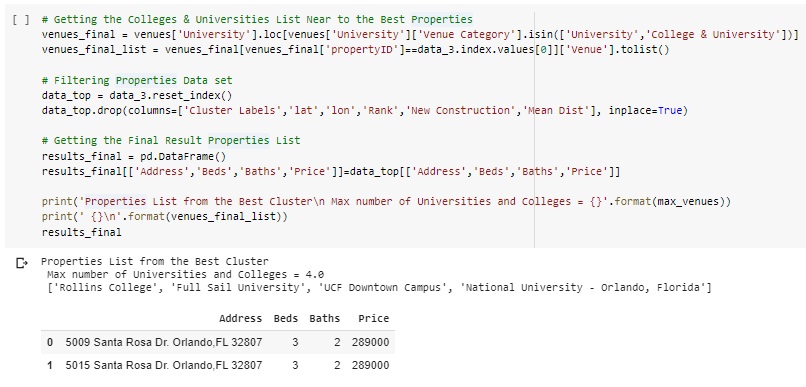
A total of 655 venues near the properties were found using the Foursquare API. By using the keyword University 371 venues, and 284 using the keyword College. But, some of them only use these words in their names or description, which is required to filter the venues by their category ID to be sure that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



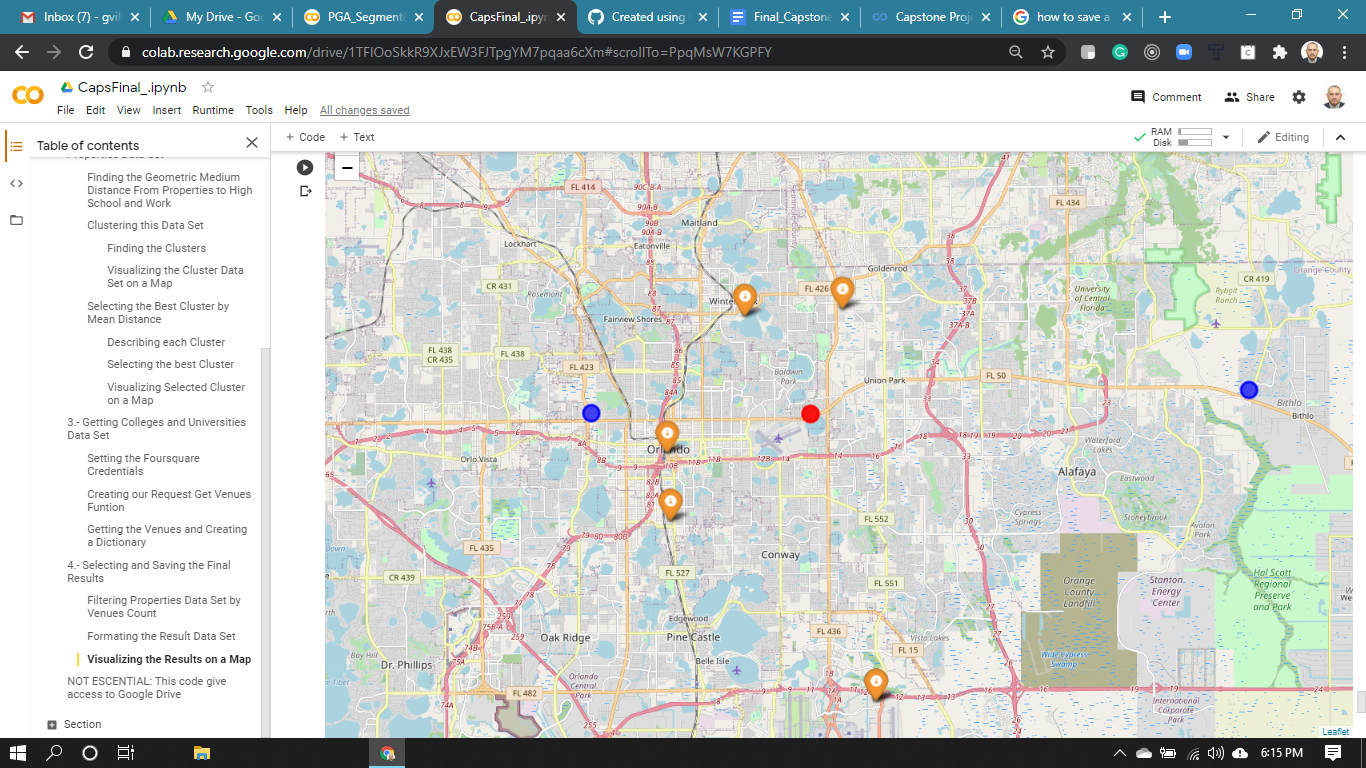
### Step 4

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.



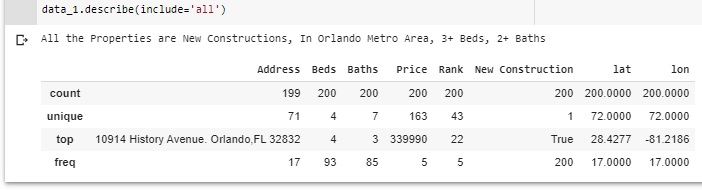
How was presented in the last results four venues are in the list: “Rollins College”, “Full Sail University”, “UCF Downtown Campus”, and “National University - Orlando, Florida”.

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.

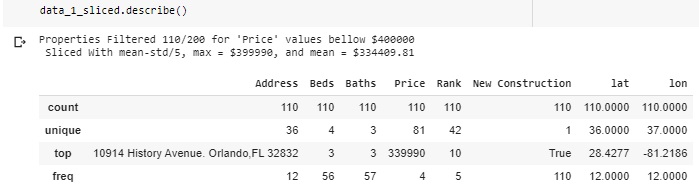


## Analysis

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.

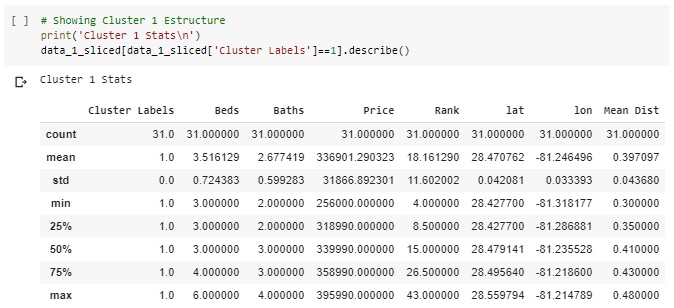


Following the same logic order, there is a Price ratio of 74% approx (163/200 in first and 81/110 in second). Which explains, a 26% of properties have a list price or the same conditions for sale.



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.

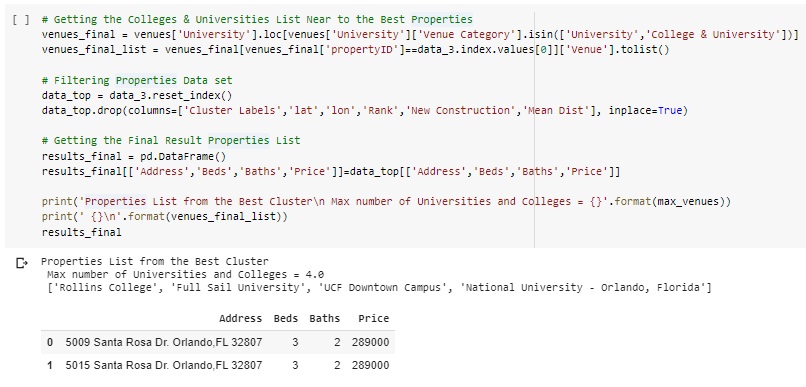
Regarding the selected cluster, this presents less than 1% standard deviation from the mean location coordinates (0.042/28.470 for latitude or 0.033/81.246 for longitude), and 11% for the mean distance (0.044/0.397). These can understand that all properties are relatively near and equidistant to the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



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.





## 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.