FINAL CAPSTONE PROJECT: "Finding the Best House"

Gabriel Villa

August 2020

# Introduction

## Background

Find a house to buy may be an easy task if you know your needs exactly. But some times, you don’t have the correct advice, didn’t use the correct tools, or simply don’t know the correct path to explore the market before asking for a deal. Wich could be so complicated to find it, or even worst, buy the wrong option.

## Problem

A family plans to buy a house and ask for a 'New Construction'. When the realtor asks them for their needs, the customers said:

“We need to find a 'Unifamily' Property with sale 'Price' under $400000, 3 or more 'Beds', 2 or more 'Baths' in Orlando, FL. The 'Property' needs to be near the most count of Universities and Colleges in Orlando, FL. Also, the best location for the 'Orlando Science High School' and our 'Office in Down Town'.”

This project aims to propose a code to select the best options to buy a property, based on the customer needs, data sets from the realty market, and geolocated databases.

## Interest

Realtors would be very interested in accurately chosen for a new property to their customers, because they could gain competitive advantage and better business. Others who are interested in realty such investors and lenders may also be.

## Data Acquisition and Cleaning

### Data Sources

The first source of data is the customer needs, with this I have the parameters to look for the properties data set and the venues data set, also the keys to filter and to select the options to be proposed.

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=“Down Town”, High=“Orlando Science High School”

**Venues Required:** “College”, “University”

The second source is the Realty API, under license of ***rapidapi.com,*** I got the credentials to use the ***realtor.com*** API. This is a *request GET* API, wich response with an XML data string. The string needs to be converted in a JSON object to accessing, indexing, formatting, and filtering the info required. According to restrictions in this service, I put a limit of 200 properties in the request in a radius of 5 miles. The result is a Pandas ***DataFrame*** with six columns [‘Address’,’Beds’,’Baths’,’Price’,’Rank’,’New Construction’], where:

**‘Address’** values are dictionaries with each element like 1st Line, City, State, etc.

**’Beds’,’Baths’,’Price’, and ’Rank’** are integer values.

**’New Construction’** is a boolean.

The third source is the Foursquare API, I got the credentials to used it from the Foursquare Developer Page. This is a *request GET* API, wich response with a JSON object, The JSON needs to be converted to a dictionary to accessing, indexing, formatting, and filtering the info required. According to restrictions in this service, I put a limit of 100 venues in the request in a radius of 6000 meters. The result is a big ***Dictionary*** with 2 Tags in the first level ‘College’ and ‘University’ the two keys used to request the API per each property in the best cluster selected.

### Data Cleaning

When I requested the property data, the "Address" value found was a dictionary with these fields "First line", "City", "State Code" and "Postal Code". All this information was merged to have the correct format.

To cluster the property data, I found the "Mean Distance" from the "High School" and the "Work" to each property. Then this data was normalized and I added a new column called "Mean Dist". Furthermore, use that column to generate 5 groups of properties and label them, adding a new column called "Cluster Labels".

For the venues data, I first had to select the best cluster of properties based on the smallest "Mean Custer Distance". With this, I was able to know the list of properties to search the Venues data. Then, I located in the Venues Dictionary only the fields: "Name", Address", "Latitude", "Longitude" and "Category". To count the Venues for each property and create a list of properties with the highest count.

## Method

### Step 1

In the first place, I need to find a 'Unifamily' properties list, available for sale with 3 or more 'Beds', 2 or more 'Baths' in Orlando, FL using a "Realty API". When I get it, the list needs to be filtered to choose properties with a price under $400000. For that, I going to review the Properties' DataFrame with "Pandas" and visualize the result lists 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 clustering with "K-means". Also, I select the 'Best Cluster' and find a second properties list. I going to review the Properties' DataFrame with "Pandas" and visualize the result lists with "Folium" maps.

### Step 3

In third place, I analyze the 'Venues' around each 'Property', finding the Universities and Colleges using the "Foursquare API". The end of this process is a third property list with most count of 'Venues', which they will be selected to look for a deal. I going to review the Properties' DataFrame with "Pandas" and visualize the result lists with "Folium" maps.

### Note:

I use the data sets in CSV files to save each step results in my "Google Drive".