Data Cleaning and EDA

Aldo N. Cao Romero

May 2022

We begin by importing the proper packages such as numpy, pandas and matplotlib. We observed the data by using the commands pd.read_csv and data_frame.head(5). The first five rows of the used data can be observed in figure (1), which is the info of house sales. Also we compute the shape of the data given as a result of 5000 rows and 16 columns. Then, by typing data_frame.dtypes we had access to the data types. Figure

index	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitchen_features	fireplaces	floor_covering	HOA
0	21530491	5300000.0	85637	-1.103.782	31.356.362	2154.0	5272.0	1941	13	10	10500	0	Dishwasher, Freezer, Refrigerator, Oven	6.0	Mexican Tile, Wood	0
1	21529082	4200000.0	85646	-111.045.371	31.594.213	1707.0	10422.36	1997	2	2	7300	0	Dishwasher, Garbage Disposal	5.0	Natural Stone, Other	0
2	3054672	4200000.0	85646	-111.040.707	31.594.844	1707.0	10482.0	1997	2	3	None	None	Dishwasher, Garbage Disposal, Refrigerator	5.0	Natural Stone, Other: Rock	None
3	21919321	4500000.0	85646	-111.035.925	31.645.878	636.67	8418.58	1930	7	5	9019	4	Dishwasher, Double Sink, Pantry: Butler, Refrigerator	4.0	Ceramic Tile, Laminate, Wood	None
4	21306357	3411450.0	85750	-110.813.768	32.285.162	3.21	15393.0	1995	4	6	6396	3	Dishwasher, Garbage Disposal, Refrigerator, Microwave, Oven	5.0	Carpet, Concrete	55

Figure 1: First five rows of the used data to be cleansed

```
MLS                int64
sold_price       float64
zipcode            int64
longitude         object
latitude          object
lot_acres        float64
taxes            float64
year_built         int64
bedrooms           int64
bathrooms         object
sqrt_ft           object
garage            object
kitchen_features  object
fireplaces       float64
floor_covering    object
HOA               object
dtype: object
```

Figure 2: Types of variable of every column in the data

From this point we can make some observations. First of all, by seeing the table (1) we note that some variables can have a None value. This value can only be assigned to categorical variables. So, we began by doing that and apply an algorithm to search for the None values in each categorical column, that is to say, in every string column. The output of this process can be seen in the table (3)

```
bathrooms:
6
sqrt_ft:
56
garage:
7
kitchen_features:
33
floor_covering:
1
HOA:
562
```

Figure 3: Categorical variables with their amount of None values

Additionally, by observing the table (2), we see that some variables are categorical (i.e. are object type) when they are supposed to be numerical, for example, the number of bathrooms, the surface of the house (sqrt_ft), the garage, and, most likely, since it has numbers, the HOA.

So, an algorithm was made in order to change the None values by zero,and then, by using the command .astype(float/int) we convert the whole column into a number.

Another thing that is important to be noted. For some variable it is ok having a zero value (like bathrooms or garage) but for other variables it does not make sense having zero values. For instance, in 'year built' we find 5 zeros. But since is a small amount of data, we can be able of neglecting those rows, as well as in bathrooms. Also, by doing zero the None values for the variable sqrt_ft we have to make sure there is not, indeed, a house there, so, by simply counting the number of bedrooms we were able to observe that all the houses had bedrooms. So, we conclude that we cannot use zero values for the sqrt_ft variable.

First, we work with the HOA variable. We compute the mode of this variable, and we observe that most of the values were zero, so, we keep as zero all the none values and continue with them. Then, for the sqrt_ft variable we compute the mean, median and mode, and we observe that these values where really closely, so this variable follows a normal distribution, so any of these could work for the none values. We chose to use the mean.

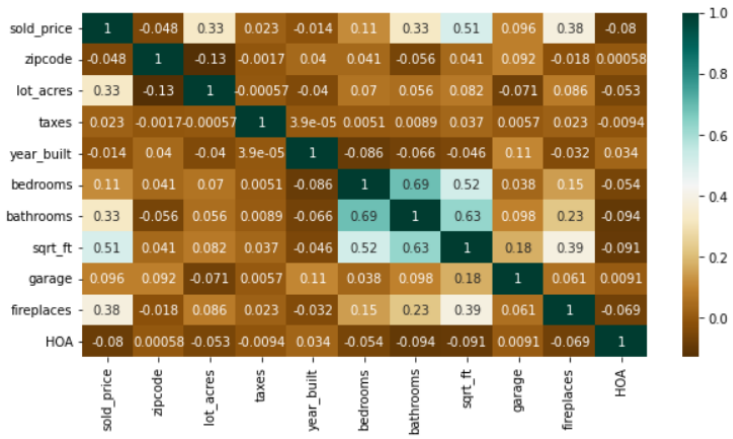

Figure 4: Heatmap of the correlation matrix showing the correlation among variables.

Now, we got rid off the null values. So, by using the .isnull().sum() method we compute the number of null values in our data frame.In our case were lot_acres and fireplaces with 10 and 25 null (or NaN) values respectively. We dropped all of them and the shape of the data was barely affected (4973, 16). We based our action in the 5% criteria.

We analyzed the MLS column, by making some research we conclude there could be something weird with this variable. So, with the numpy package, we apply the unique method to see how was the variation of this variable, then we measure the length and we noticed that it has the same size of the whole data (4973 so far), so, there were no same values in this variable, therefore, it does not contribute anything to the analysis. We just dropped it.

In order to complement our work, we present a heat map that shows the correlation among variables. We observe in the figure (4) that the sales price is highly correlated to the lot_acres, the bathrooms and sqrt_ft. In the same manner, bathrooms and bedrooms are dependent between them and sqrt_ft.
In order to correct the outliers, we made an analysis based on the box plots. Here, we present some of them.

Figures (5) and (6)

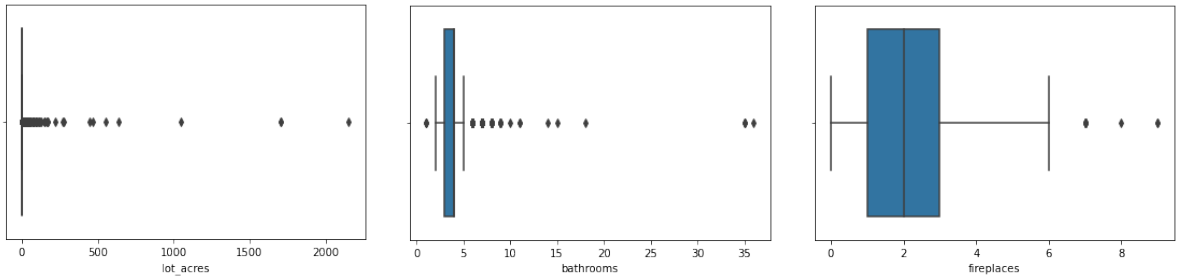

Figure 5: Box plots made to identify the outliers for the variables lot_acres, bathrooms and fireplaces.

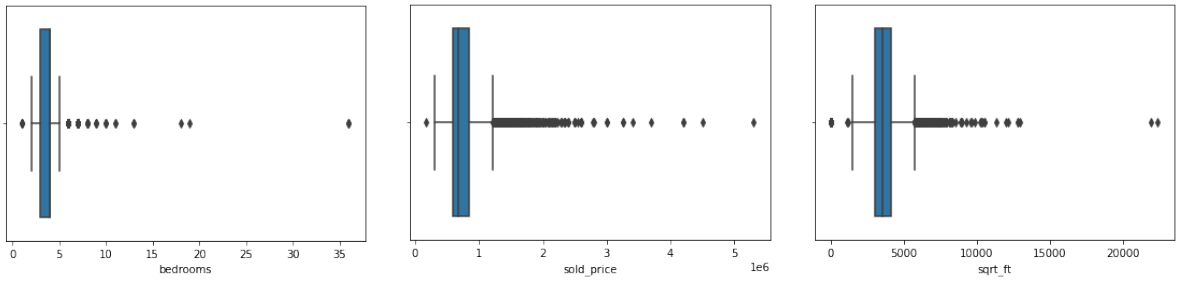

Figure 6: Box plots made to identify the outliers for the variables bedrooms sold_prices and sqrt_ft.

We observe in the lot_acres variable that it seems that I forgot to analyse the zero values there and, at least after 500, we have several outliers, so, we apply the command ˜ in the following manner:

kitchen_features	fireplaces	floor_covering	HOA	kitchen coder	floor coder
Dishwasher, Freezer, Refrigerator, Oven	6.0	Mexican Tile, Wood	0	[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, ...
Dishwasher, Garbage Disposal	5.0	Natural Stone, Other	0	[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, ...
					[0.0

Figure 7: Result of the encoding kitchen_features and floor_covering.

$$\text{data\_frame} = \text{data\_frame}[\tilde{}(\text{data\_frame}[\text{"lot\_acres"}] > 500)]$$

$$\text{data\_frame} = \text{data\_frame}[\tilde{}(\text{data\_frame}[\text{"lot\_acres"}] == 0)]$$

in this manner, and with the help of the box plots we were able to eliminate some rows that are identified as outliers. Once this was made we compute the shape of the data and it was (4916,15). So we didn't eliminate to many rows so, we still have a good amount of data without significant outliers. If we try to

apply the criteria

$$\mathrm{df} = \mathrm{df}[\ ((\mathrm{df} < (\mathrm{Q1} - 1.5 * \mathrm{IQR}))|(\mathrm{df} > (\mathrm{Q3} + 1.5 * \mathrm{IQR}))).\mathrm{any}(\mathrm{axis} = 1)]$$

where IQR = Q3-Q1, which is the distance between the first and the third quantile in the box plot, we got as a result the elimination of almost the 30% of the data, which was quite a lot.

Finally, we encoded every item or observations for the kitchen_features and floor_covering. We create an algorithm that searches for any unique value in the data base, we sort them into an array and then, following that order, we made a vector of lists and, depending on the row, we put zero or one depending on the case. If the house has an item of the 'unique' list, we assign one, and if it does not have it, we assign zero value. The result of this can be seen in the figure (7) There must be said, that the order of the encoding depends on how the program sorted every label, so the programmer has to have access to this information to know exactly which item the house has.