Data Wrangling

Reading The Dataframe

In order to read the data into my notebook, I had to set up the environment in my jupyter notebook. I first imported the modules needed for my project, this includes:

- Pandas
- NumPy
- matplotlib.pyplot
- Seaborn

Once I had the packages I was able to read the file using a pandas method called **.read_csv()**, this allowed me to create a dataframe from the csv file.

Exploring and Cleaning the Dataframe

To explore the dataset I used a few methods that helped me understand the current raw data. These methods are .head(), and .info()

- **.head()**
  - Using .head() method I was able to visualize the dataset in a tabular format. I can see the name of the columns, along with the zero-indexed rows. I was also able to check if I read in the data correctly.

- **.info()**
  - This method provides a quick summary of our dataset. Which includes, the number of features (columns) and observations (rows), the number of non null observations in each column, and the datatype of each column.

From observing the dataset using .info() the dataset had 21613 observations and 21 features. I also doubled checked the dataset to see if there were Null values using **.isna()** and **.sum()**, The outcome was the sum of all null values which was 0 for each column, proving what we learned in using .info() was correct.

Next, I decided to drop the id column. The id column will not be beneficial to our model as it only indicates a house and provides no substantial properties. From using .info() we can see majority of the columns are an int or a float. However, some columns like waterfront (if a house has a waterfront) sounded more like a category (yes/no) than an numeric value. I did a quick check using the method .unique() to see if my intuition was correct. The features that I changed to categorical were waterfront, condition, grade, and view. I also changed the date column to a datetime datatype.

- **.unique()**
    - This method allowed me to find all the unique numbers of a certain column. In order to see if a column could be a category or not I had to check the results of all the observations. If a column had more than a closed set of number it is safe to say it is not a category, but if it has a limited amount of numbers we can turn that feature into a category.
- **.astype()**
    - This method allowed me to change data types. The column's data type I changed were waterfront, condition, grade, and view.

The step in exploring my dataset was to take a quick look at the descriptive statistics of all numerical features. This task was done by using the .describe() method.

- **.describe()**
    - This method generates descriptive statistics that summarize the central tendency, dispersion, and shape of the dataset. This was a great opportunity to see if any columns contain any erroneous data such as outliers.

From using .describe() I was able to conclude that bedrooms column indeed has an outlier, which was the value 33. Using a matplotlib's boxplot function along with the .value_counts() method I was able to confirm my initial thoughts. Since this outlier was only one row/observation I decided to not include this row in our analysis. Using boolean indexing the new dataframe contain all values that bedrooms were not 33.

After finding this one outlier, I decided to create boxplots on all other numeric columns, in order to see if there were more outliers. From the boxplots I concluded what there were an abundant of data that can be considered outliers. I decided not to delete the outliers as it may impact our results.