Data Visualization

Pratishtha Abrol - 2020121002

Gunjan Gupta - 2019111035

The assignment requires us to analyse the given cleaned dataset to suit the requirements of certain investors and make it more informative at first glance. As we go through the specified requirements, we thoroughly analyse the dataset, its columns, and their respective unique values, their frequencies, and how they vary with other columns. Let us walk you through our approach.

We start with a basic overview of the dataset, its columns, their data types, number of columns, and the first few dataset entries. We notice that quite a few columns such as the `Price`, the `Price_per_unit_area`, the `No_of_BHK` and the `Size` columns have object data types. Analysing these would at first require us to convert them to integer. We convert the `Price`, and the `Price_per_unit_area` columns into integer as per our first requirement. Describing them using the `df.describe()` function we get the following perspective into the dataset, specifically, these two columns.

	Price_per_unit_area	Price
count	143708.000000	1.437080e+05
mean	10664.535600	1.365094e+07
std	8512.886153	1.940871e+07
min	104.000000	1.000000e+05
25%	5256.000000	5.259438e+06
50%	7938.000000	8.500000e+06
75%	13333.000000	1.549800e+07
max	171969.000000	8.000000e+08

1. Every investor has different budget ranges, so divide the overall opportunities into three ranges

We, at first, cosider a uniform division, with equal price ranges for the start, `udf=1` consists of houses in price range 1,00,000 to 26,00,00,000, `udf=2` in price range 26,00,00,000 to 52,00,00,000 and `udf=3` in range 52,00,00,000 to 80,00,00,000.

We see that the frequency of the houses in these ranges varies dramatically, with 143605 houses in `udf=1`, 93 in `udf=2` and only 10 in `udf=3`. Therefore, we utilise a much more practical, a real world price range distribution, with `rdf=1` for price range 1,00,000 to 80,00,000, `rdf=2` for range 80,00,000 to 2,50,00,000, and `rdf=3` for price range 2,50,00,000 to 80,00,00,000.

The reason for this price range can be logically explained by the 3 sets of income in the society. The person, belonging to lower middle class and lower class of economic background, with family income upto 12,00,000 is the one who will think of buying a house in the range of 1,00,000 to 80,00,000. Similary people with family income upto 50,00,000 will be the ones buying houses in middle price range of 80,00,000 to 2,50,00,000. So these price ranges comply with the real world situation. This can be reasoned by saying that they can take a loan upto 80% of the house price which will make them pay installments of 25,00,000-30,00,000 each year. One need 50% of the income generally to survive according to the societal standards. So this range make appropriate sense.

We still see that the frequency of the houses in these 3rd range is very less but not that badly as uniform distribution, with houses 67514 in `udf=1`, 60267 in `udf=2` and only 15927 in `udf=3`. So, we adjust our ranges a little, catering to the real world, but also providing a somewhat uniform distribution of properties across these ranges as`rdf=1` for price range 1,00,000 to 70,00,000, `rdf=2` for range 70,00,000 to 1,50,00,000, and `rdf=3` for price range 1,50,00,000 to 80,00,00,000.

This yeilds a much more uniform distribution of the frequency of the houses in these ranges, with 57451 houses in `rdf=1`, 49546 in `rdf=2` and 36711 in `rdf=3`. So, we will henceforth follow this division based on prices.

Our first visualisation is of these price range with respect to the number of properties available in the range. This would allow the investors to know how many properties lie within which range at first glance. They can easily pick a range depending on their preference.

Following this, we explore the dataset a little, we look at the variation of the property's `Price` with its location (`City_name`), number of rooms (`No_of_BHK`) and `Property_type`. Each of these have the `Price` value varying on the vertical axis, which provides the innvestors a bird's eye view of the properties of the available plots in their price range.

2. Some investors are interested only in tech-emerging cities like Hyderabad, Banglore, and Chennai with ready-to-move properties for leasing purposes.

Ofcourse apartments for lease should be ready-to-move-in, so our first task in this is to select the properties with the `is_ready_to_move` set to `True`. Once that is done, we can start exploring out subset of data for variations of the `Price` with the `City_name`. With a scatter plot, we not only get the range of prices per city, but also get a rough estimate of the number of such properties in each of these cities. Moreover, we decided to color the scatter plot based on the number of bedrooms(`No_of_BHK` in the properties. This added another element to the visualisation. Now the investors can easily look at the properties with their requirements of price range, city and number of bedrooms easily from one graph.

Since the question only wants to focus on tech emerging cities, we simply single out the cities from the dataset, and plot the `Price` verses `City_name` scatter plot, colored according to the `No_of_BHK`.

3. Some prefer larger area properties with relatively lesser budgets.

To select the properties with a larger area but a relatively less budget, it makes sense to look at the `Price_per_unit_area` property. Here, we also convert `Size` property to integer, to better make sense of the values. A simple scatter plot visualisation of the `Price` to `Size` graph shows us that all elements bellow the average `Price_per_unit_area` diagonal should meet the requirements of a higer size to

lower proce ratio. To construct this diagonal, we do a `df.describe()` to find out the summary of the `Price_per_unit_area` property. We get the following output:

```
count    143708.000000
mean      10664.535600
std        8512.886153
min         104.000000
25%        5256.000000
50%        7938.000000
75%       13333.000000
max      171969.000000
```

The diagonal can easily be made with the mean (rounded off to 10664), and represented through points `[[0,0],[1,10664]]`. We add this diagonal to out scatter plot.

Plotting the scatter plot for the data below this diagonal, we get the properties matching our requirements.

Going a step further, we create bar graphs for the same, by dividing the size of the area into ranges of 100 square feet to 700 square feet, 700 square feet to 1000 square feet, 1000 square feet to 1600 square feet, and 1600 square feet to 90000 square feet. We will refer to these slices as `s_sl=1`, `s_sl=2`, `s_sl=3`, and `s_sl=4` respectively. We get the frequencies of the properties matching our requirements according to their size ranges. Quantitatively, we have 18864 low budget property opportunities in the size range of 100 square feet to 700 square feet, 19873 such in 700 square feet to 1000 square feet, 36405 in 1000 square feet to 1600 square feet, and 21655 in 1600 square feet to 90000 square feet.

Doing the same, but dividing it into price ranges this time, we get a frequency of required properties in price ranges. Specifically, 54267 larger but comparetively low budget property opportunities in the price range of 1,00,000 to 26,00,00,000, 32901 in 26,00,00,000 to 52,00,00,000 and 9629 in range 52,00,00,000 to 80,00,00,000.

4. Some investors prefer making all the investments in one locality, while some are interested in diversifying the portfolio by assets spread across one city.

We split the database based on cities. This would allow us to make bar graphs for the number of properties based on their localities in each city. Such a method allows investors to look at the data in a statistical way, apealing to both, those who want to invest by locality or across the city. In order to do this, our first step requires finding all the unique values in the `City_name` column. WE find that the dataset spans across 8 cities, namely Ahemdabad, Bangalore, Chennai, Delhi, Hyderabad, Kolkata, Lucknow, and Mumbai.

Accordingly, we take splits of dataset and plot frequencies per city. as bar graphs. It is evident from these that some localities per city have a huge number of available properties and a large number of localities have only a minimal number. Investors can now choose a city from the bar graph representing the number of properties per city, and a locality from that city through the locality frequency visualisation bar graphs for each city. Specifically, we have the counts per city as:

```
Mumbai       72152
Bangalore    21019
Chennai      13406
Kolkata      12224
Hyderabad     9129
Delhi         7476
Ahmedabad     6612
Lucknow       1690
```

Please refer to the frequencies per locality in the accompanying jupyter notebook, since each city has about 100 localities.

We also explore the cost variations per locality in each of these 8 cities. The next 8 graphs give a visualisation of the number of properties in each locality, their price and they are colored according to the number of bedrooms in each property, accurately catering to every possible requirement of the investor in such a case.

5. Some investors are interested in knowing the hotspot for their offices in Mumbai and Ahmedabad.

To look at the office hotspots specifically in Mumbai and Ahemdabad, we can divide this into 2 subsets. Investers who are interested in a single locality should find a locality with the most available properties, and those who are diversified should find the locality that, along with other neighbouring localities would form the max available properties for sale. The first subset is easily solvable by looking at the datasets and respective frequencies for each locality. We do this in the following part for both the cities.

The second subset, although solvable, involves dynamic programming algorithms on the dataset and higher computation power, which is beyond the scope of this assignment.

As per our results, we can suggest the localities of Bopal, Shela, and Gota in Ahemdabad, with 999, 984 and 825 properties for sale respectively, and Mira Road East, and Thane West in Mumbai with 9590, 9559 properties each.