**Report**

**Introduction:**

1. **Problem definition**

The main goal of conducting our research is to provide valuable insights about the lifestyle of people, affordability of houses, and housing market trends across different regions. Highlighting what kind of relationship different features of a house have with its rent would be a primary goal of our research. We have gathered several research-oriented questions which we intend to answer after we have done performing analysis on the provided dataset. These questions are as follows:

* How does the rent of a house change with its size?
* Which city is the most expensive to live in? (in terms of house rent affordability)
* Does the floor level of a house affect its rent?
* How do the houses in each city compare with each other in terms of the size of the house and the number of floors?

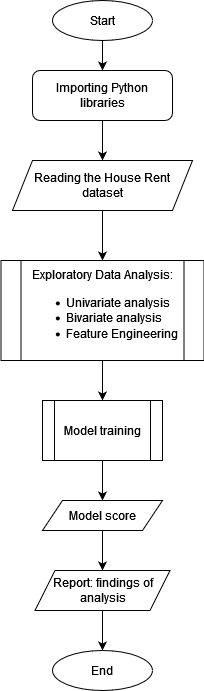
In this report, we will present the procedures involved in importing, analyzing, exploring, and transforming the given dataset about house rent prices to find valuable insights about the data and answer important business-oriented questions.

1. **Software/hardware specification**

* We have used Jupyter Notebook as the development environment as we have run all our code in it.
* For the programming language, we have performed all our tasks using Python and its libraries.
* All operations including model training were executed on a laptop with 4 GB RAM, integrated intel HD graphics, and Windows 10 as the operating system.

1. **Flowchart for activities performed in the analysis**

* **General flowchart:**

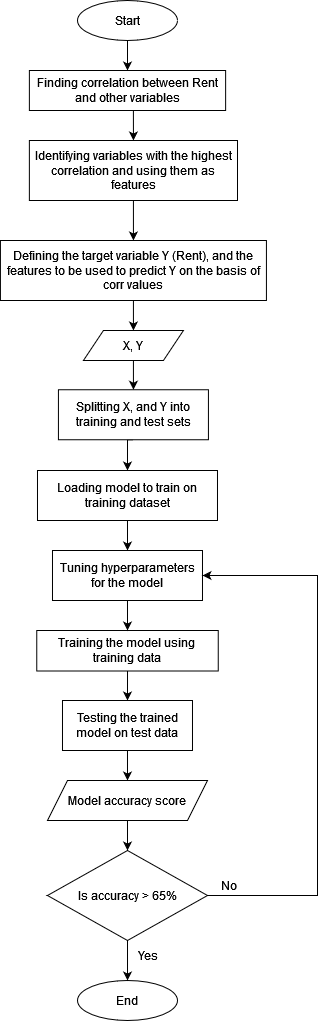
****

* **Sub-flows involved:**

1. **Activity flow diagram of Exploratory Data Analysis**

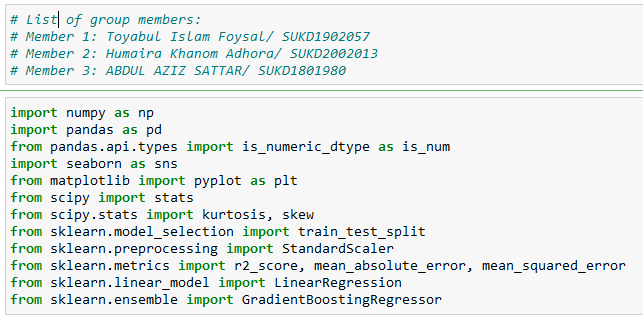
****

1. **Activity flow diagram for data classification:**

****

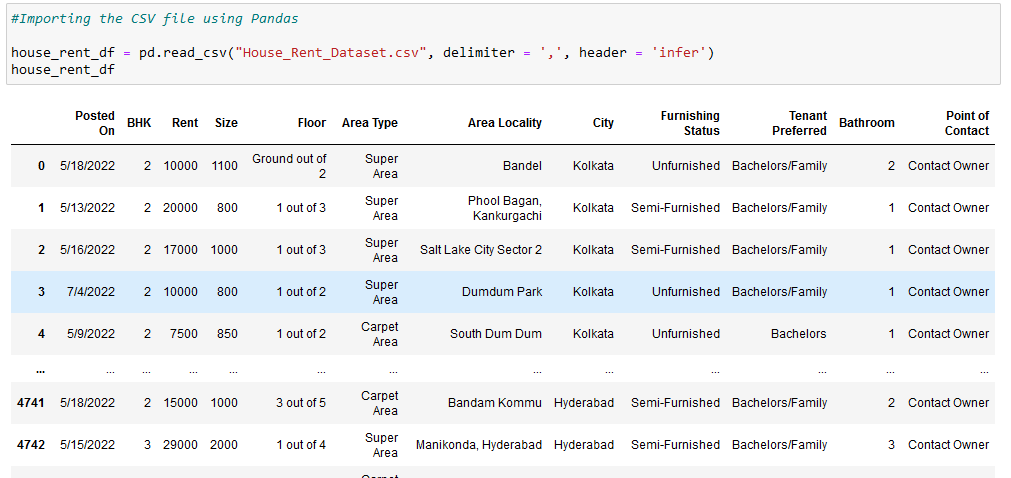
**Section-1: Importing the Dataset**

* 1. **Importing required libraries:**

****

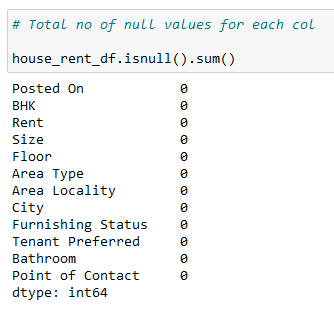
The libraries that we have imported for our analysis are as follows:

1. Numpy
2. Pandas
3. Seaborn
4. Scipy
5. Sklearn
   1. **Importing the dataset:**

****

We import the dataset CSV file stored in the Jupyter Notebook using the *read* command of Pandas. As can be seen, the original dataset has a total of 12 columns and 4742 rows with each column representing some feature about a particular house.

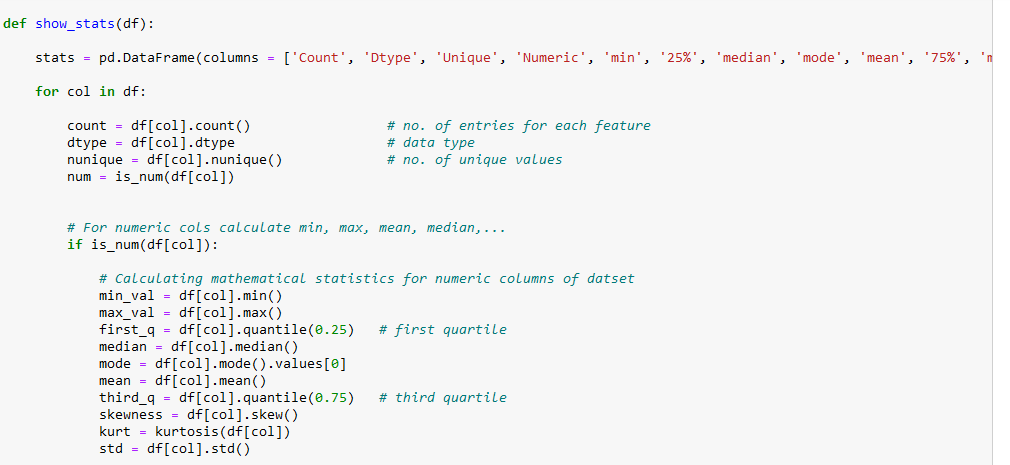
**Section-2: Exploratory Data Analysis (EDA)**

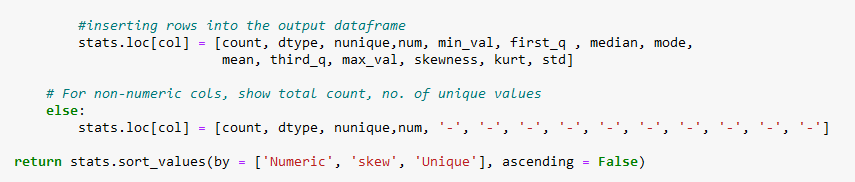


After importing the data, we check for any null values in the dataset if present. As can be seen in the code snippet above, null values are completely absent from the dataset. Therefore, we don’t have to worry about removing the null values.

We didn’t know the meaning of BHK in the columns that we had been given so after researching it on the internet, we found out that BHK refers to the abbreviation: Bathroom, Hall, and Kitchen. It can be thought of as a way to measure the number of rooms present in a house based on the proportion of rooms for bathrooms, halls, and kitchens. [1]

The number associated with BHK represents the number of bedrooms in that house with the assumption that each house contains 1 hall and 1 kitchen. For example, a house with 3 BHK would mean that it has 3 Bedrooms, a hall and a kitchen.





We code a function *show\_stats* to find out the various statistical properties [2] for each column of our dataset. These include:

1. Count: The total number of entries
2. dtype: The data type (e.g. int, str, float etc.)
3. Unique: Total no. of unique values
4. Numeric: If the column is numeric or not
5. Min: The minimum value of the column
6. Max: The highest value in the column
7. 25% (first quarter): The value below which 25% of the data lie
8. Median: The value below which 50% of the data lie
9. Mode: The most commonly repeated value
10. Mean: The average
11. 75% (third quarter): The value below which 75% of the data lie
12. Skew (skewness): How much the graph is skewed and in which direction
13. Kurt (kurtosis): The vertical height of the graph
14. Std (standard deviation): The deviation from mean

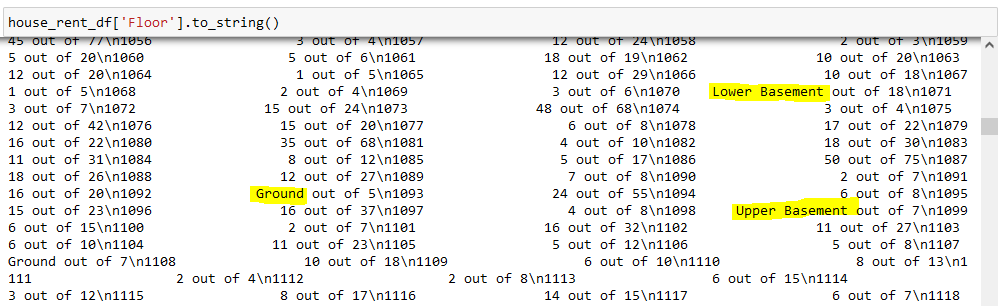
We can conclude from the output statistical information that only 4 of the 12 columns are numeric and the rest are categorical. The numeric columns include *Rent*, *Size*, *BHK*, and *Bathroom*.

**2.1 Data transformation**

**2.1.1 Feature Engineering: Adding 2 new extra features: *Floor Level* and *Max Floor***

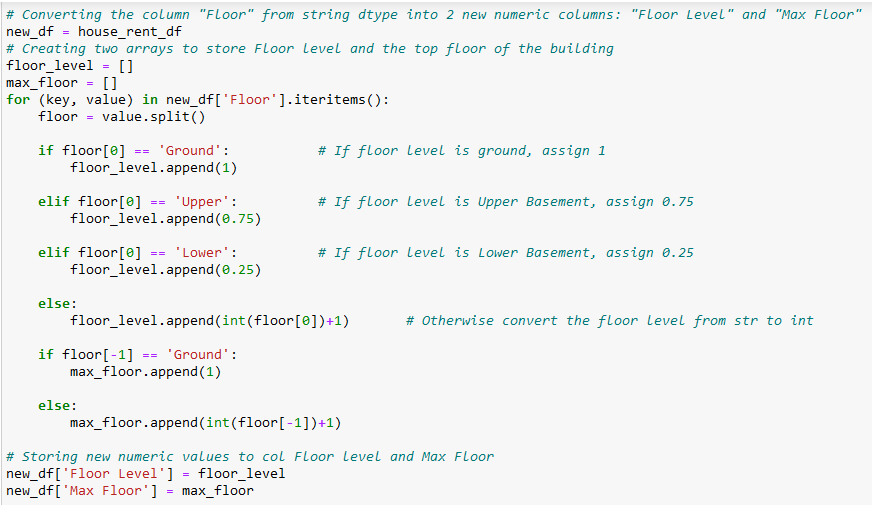
The *Floor* column was expected to be in numeric form but as we found out above, it is an object type. To find out in what form are floors represented, we run a single line of code:

It seems that the floor of a house is being stored as a string object where each entry represents the current floor level together with the top floor of the building. Therefore, we can extract 2 different kinds of information from this: (i) the floor level, and (ii) the top floor.



On further analysis, we found out that the floor level can be of 4 different types:

1. Lower Basement
2. Upper Basement
3. Ground
4. Any floor that is above the ground floor (i.e 1,2,3,4…)



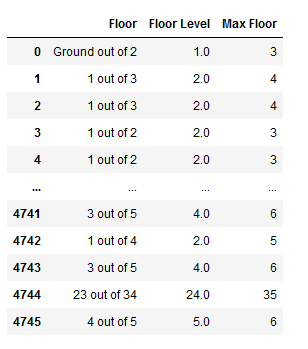
After the above analysis, we write a program to iterate all of the rows of the *Floor* column and store the information at each row in 2 new variables:

1. Floor Level: The floor level at which the family lives.
2. Max Floor: The highest floor level of the building.

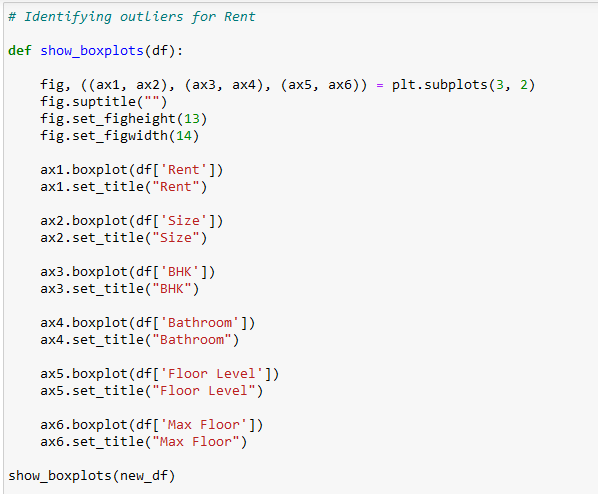
As we have found out that floors exist below the ground level (i.e. Upper and Lower Basement), we decided to represent floors with the following number values:

1. Lower Basement = 0.25
2. Upper Basement = 0.75
3. Ground = 1
4. Nth floor = n + 1

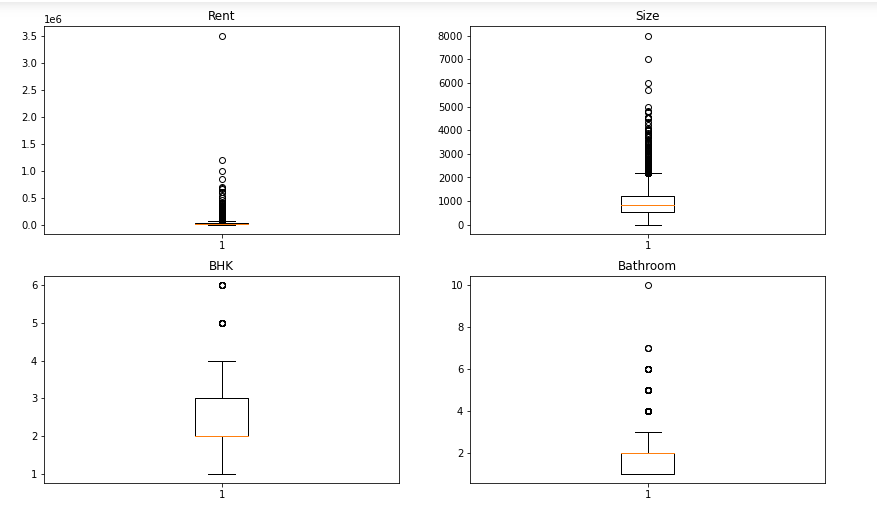
Floors are now represented as two new variables: *Floor Level* and *Max Floor*:

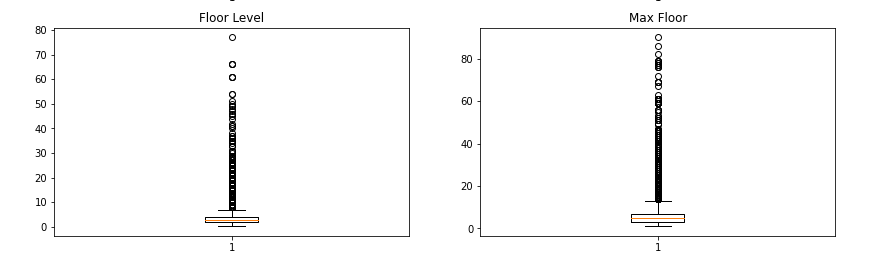


**2.1.2 Identifying and removing outliers (Data transformation)**



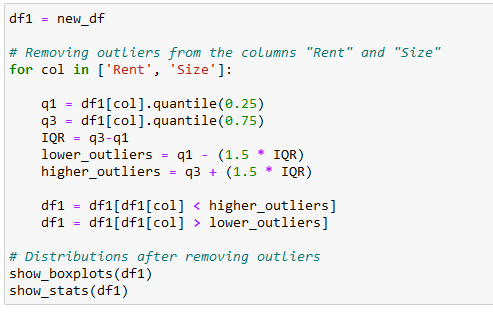
Boxplots are a great way to identify outliers [3] since they show us where the majority of the data lie. We plotted box plots of the numeric columns in our dataset using matplotlib’s *boxplot* function [4]. The result is shown below:





As can be seen from the boxplots, there are quite a few extreme values in every numeric column. For instance, in the graph of the variable *Rent*, it can be clearly seen that there is a value of rent (around 350k) that has a large difference from the other points plotted on the graph. Therefore, we can say that it is a big outlier in the dataset as it is not representative of the majority of the values.

Similarly we can see that there are outliers for every other numeric feature as well. This is expected as there might have been some error while recording those values. Our next task would be to remove these outliers in order to make the data normally distributed.

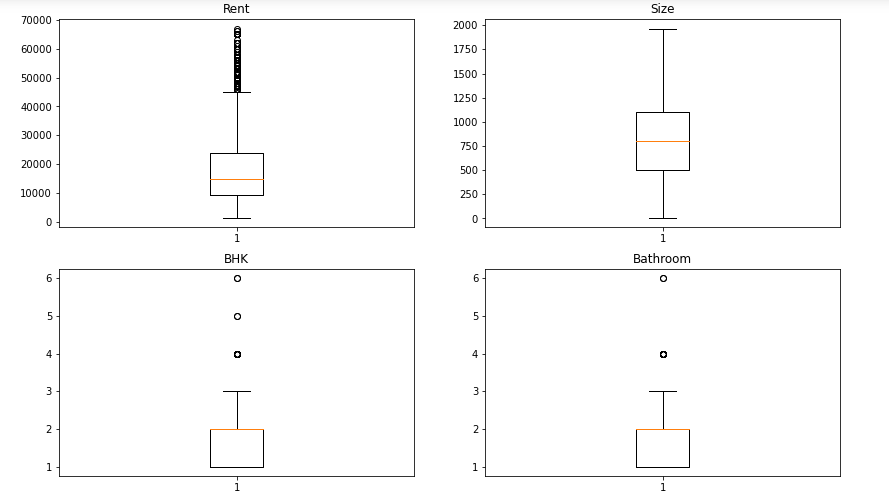


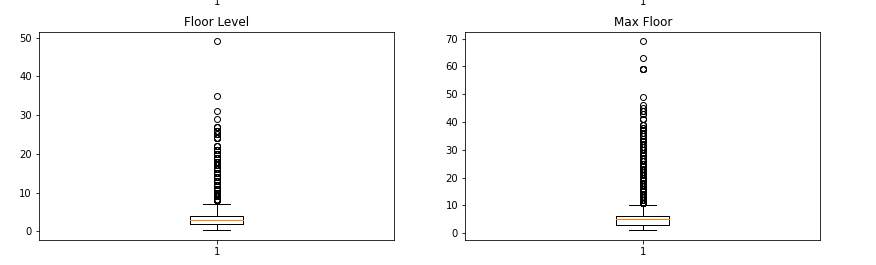
For removing outliers, we have only considered the *Rent* and *Size* fields as we identified these two variables to be the most significant. In the code screenshot above, we are removing outliers from the two main columns: *Rent* and *Size*. We have spared the rest of the numeric categories to maintain a healthy size of our dataset so that we have enough data for the rest of our analysis.

We have considered the standard criteria for classifying a value as an outlier:

1. Higher outliers: for the limit above which higher outliers lie, I have calculated 1.5 times the interquartile range and subtracted it from the value of 1st quartile. I have assumed all values above this limit to be an outlier.
2. Lower outliers: for the limit below which lower outliers lie, I have calculated 1.5 times the interquartile range and added it to the value of 3rd quartile. I have assumed all values below this limit to be an outlier.

After removing outliers, we plotted boxplots again and below are the reults:

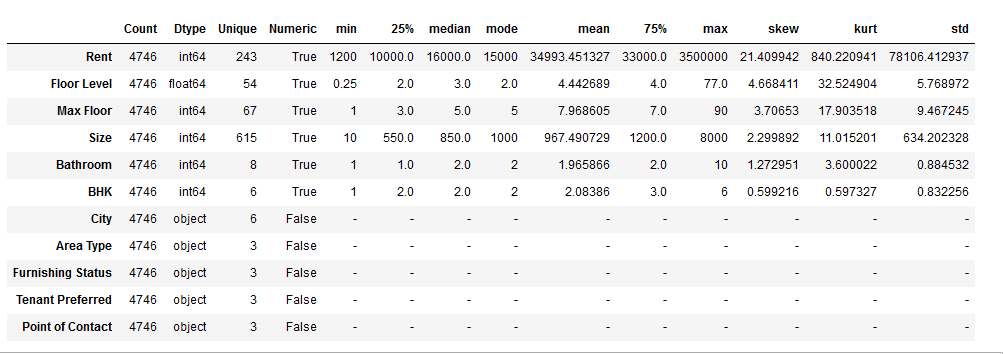




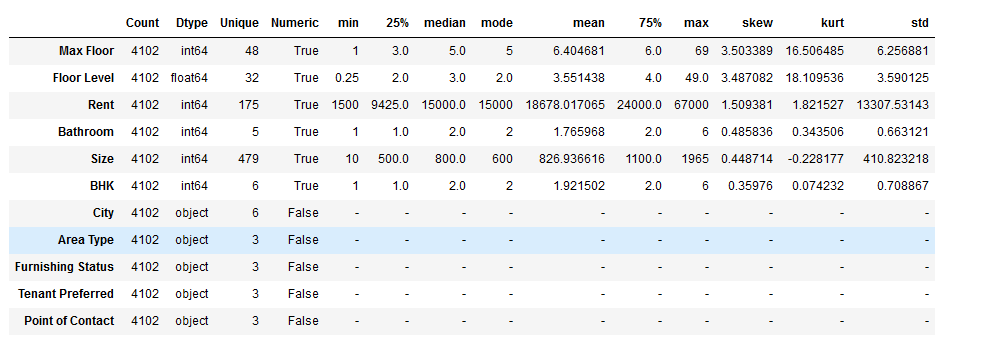
After outlier removal, it can be noticed that the extreme values from all the majority columns have reduced or been eliminated. This is especially the case for the property *Rent* and *Size* of a house as before the higher limit for *Rent*, and *Size* was 350,000 and 8000 but not it has been reduced to 70,000 and 2000 respectively.

We calculated the stats of our dataset before and after removing outliers. They are as follows:

* Stats of the dataset before removing outliers:

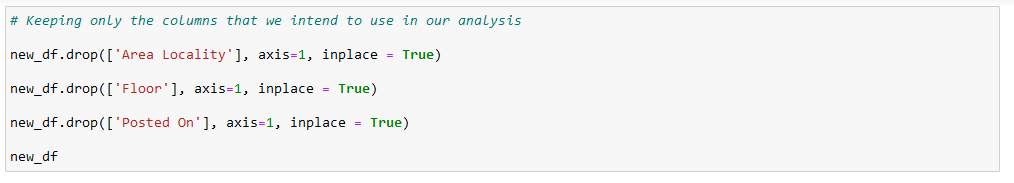


* Stats after removing outliers:



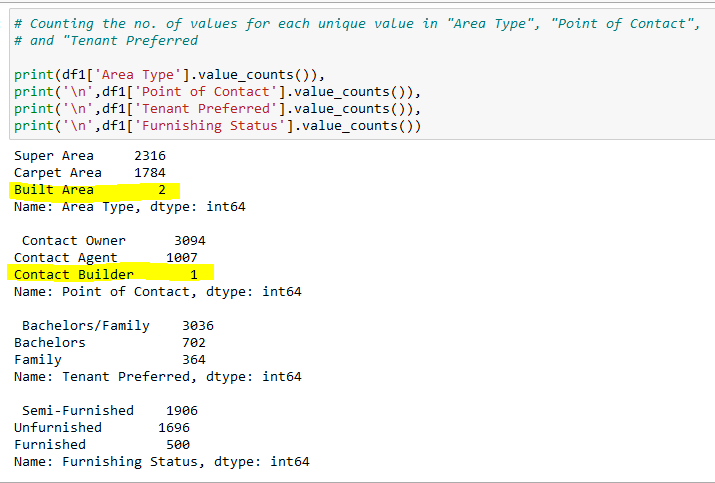
As can be analyzed from the above 2 tables, the stats for each column have changed after outlier removal. The overall Skewness and Kurtosis for the column *Rent* and *Size* have considerably decreased meaning that the distribution has become normalized. This is what we wanted as a normalized distribution (where the distribution graph is bell-shaped) would mean higher accuracy when training a model for rent prediction in the later sections.

* + 1. **Dropping unnecessary columns**



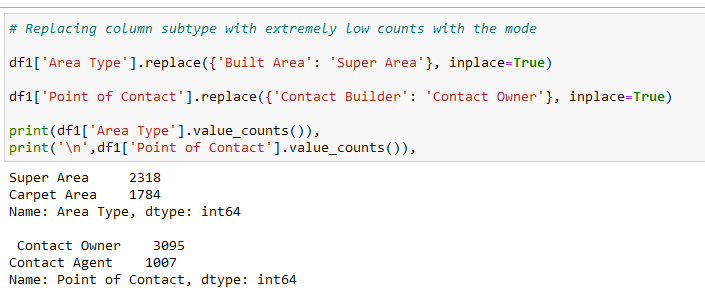
In the code screenshot above, we are dropping the columns that we will not incorporate into our analysis. We identified 3 columns namely *Area Locality*, *Floor*, and *Posted On* which we no longer require. The reasons for removing them are mentioned as follows:

1. Area Locality: This categorical column has too many unique values to be broken down individually.
2. Floor: We have already extracted the data about floors into 2 new additional columns therefore we can remove it.
3. Posted On: We didn’t consider it for our analysis as we think we already have enough useful features for analysis.



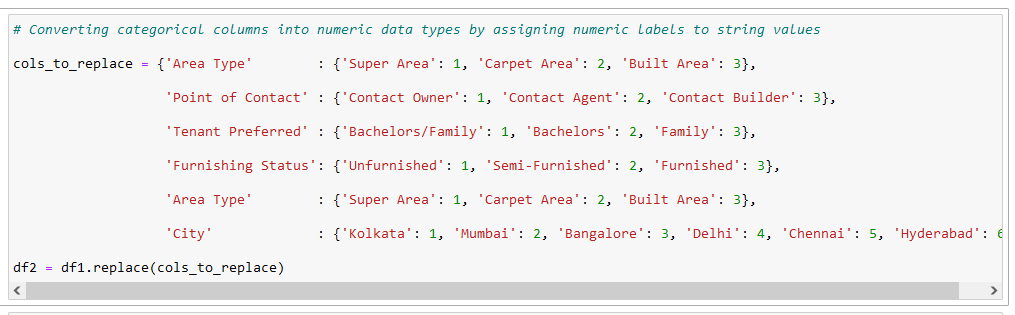
To further analyze the distribution of data in every categorical column, we count the total occurrences for each subcategory of the four categorical columns present in our dataset: *Area Type*, *Point of Contact*, *Tenant Preferred*, and *Furnishing Status* as shown in the above screenshot.

We can see from the output that in both the *Contact Owner*, and *Super Area* there is one subcategory that has unusually low value counts i.e. *Built Area* in the column *Area Type* with a count of only 2 values, and *Contact Builder* in *Point of Contact* with just a single value.



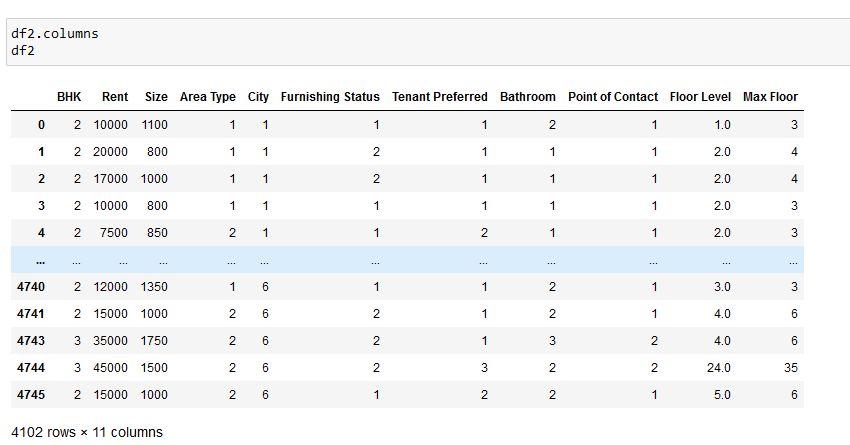
With such low value counts, dropping these subcategories would have no effect on our dataset therefore we would be replacing these subcategories with the mode of their respective category.

* + 1. **Encoding variables**



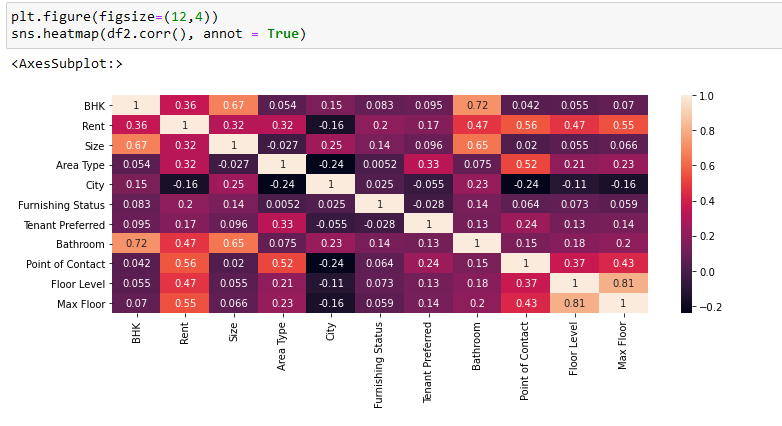
In order to use categorical columns as potential features and calculate their correlation with Rent, we have encoded all the categorical columns with numeric labels using the replace() function as can be seen above. (e.g. for the column “Area Type”, I have assigned the number 1 for value *Super Area*, 2 for *Carpet Area*, and 3 for *Built Area*.)

After encoding the subcategories of categorical features with numbers, our dataset now looks like:



Now that all of the columns in our dataset are in numerical form, we can calculate correlation of each column with Rent so that we can decide which columns will be significant deciding factors for determining the rent of a house.

**2.2 Finding correlation between *Rent* and other variables**



We have used the heatmap() function [4] of Seaborn to plot a heatmap to show the correlation of every column with every other column. Each cell stores a correlation value of 2 features in the dataset, and the lighter a cell in colour, the higher the correlation value is. The higher a correlation value for 2 features is, the more they are influenced by one another.

As we are only interested in finding out the correlation of features with respect to Rent, we will only look at the cells for the Rent feature.

To choose important features, we have considered the columns which have a correlation value of greater than 0.3. With that criteria, the following columns make it into our list of important features: *BHK, Size, Area Type, Bathroom, Point of Contact, Floor Level and Max Floor*. Among these features, the *Point of Contact* has the highest correlation value (i.e. 0.56).

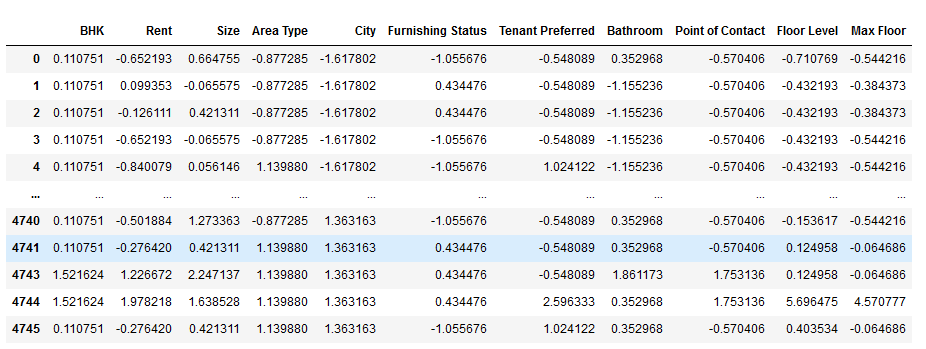
**Section-3: Model training**

**3.1 Scaling variables**

**

After identifying the features that have a significant correlation with Rent, we list down these features in a list and then scale these features using the StandardScaler() of Scikit Learn [5]. Scaling involves transforming our data into a specific range of values so that they all come on a common scale. This minimizes the difference between values of different features so that they don’t create bias when training a model on our data.

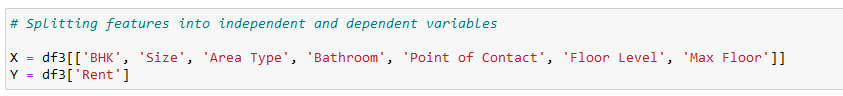
After scaling the columns, our new dataset becomes:



As can be noticed, now all the values in our dataset are more close to each other and appear to be on a similar scale than they previously were.

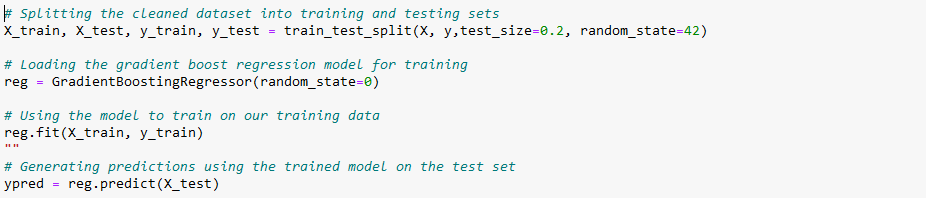
**3.2 Splitting dataset into target and predictor variables**

We will now divide the features into the target feature (the feature that we want to predict i.e. Rent) and the independent variables that will be used for predicting the target feature:



In our case the column Rent is the target feature and the rest of the columns that we identified in the previous section are the features that will be used as parameters to predict Rent.

As can be seen in the above code snippet, we have assigned all the columns except Rent into the variable X which we will be using to predict the target variable whereas we have assigned the target feature Rent to the variable Y.

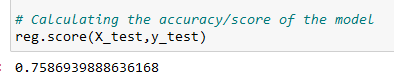


After we have scaled the values, we can finally begin training our model. For training purpose, we will first divide our dataset into two subsets i.e. the training set and the testing set. This is achieved using the train\_test\_split() function of Scikit Learn library as seen above. We have split the entire data into the training set and testing set 0.2 as the ratio for the testing set meaning that 20% of the data has been split to be used as a test set for a trained model. Whereas we will use 80% of the data to train our model. [5]

We have successfully split the data, now we will now use the Gradient Boosting Regression model as the model for training on our training set. We first create an instance of the model using the GradientBoostingRegressor() function and assign it to the variable *reg*. We then proceed to train the instance of the model on our training set using the model.fit() function giving the X and Y training sets as the parameters.

In the next step, we use our trained model to predict target values using the training set and store the predictions in the *ypred* variable.

**3.3 Calculating accuracy**



To calculate the accuracy of our model, we use the model.score() method to find out how well our model performs. In our case, the score is 0.76 when rounded off to 2 significant figures which translate to 76%. This means that whenever we use our model to predict the rent of houses, it will be correct 76% of the times.

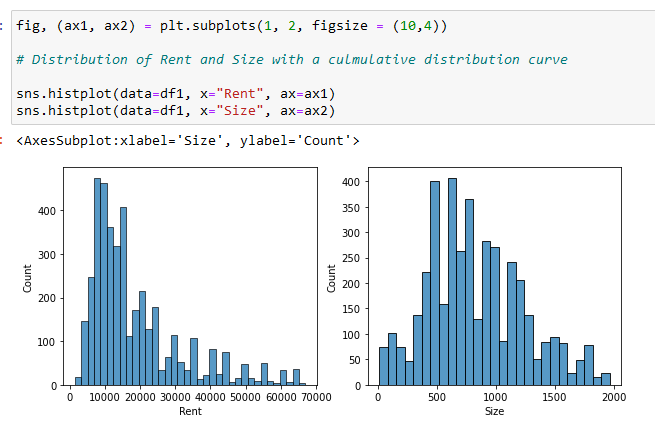
**Section-4: Conclusion**

**4.1 Findings from Exploratory Data Analysis (EDA)**

**4.1.1 What is the relationship of Rent and Size?**

In our analysis, by far the most obvious relationship would be between the size of a house and the rent of the house.

To analyze this relationship, we first plot two cumulative distribution curves to see the distribution of the Rent prices and Size of the houses in our dataset.

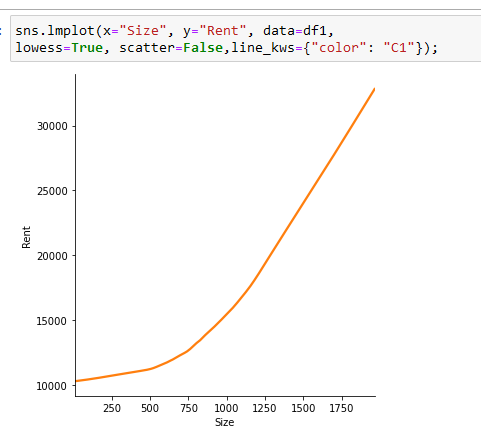


We can analyze from the graphs above that the distributions of both of the columns vary from each other.

The histogram of the values for Rent shows us that the majority of houses have comparatively low rent. We can also see that the graph of Rent is skewed to the left as the majority of the population lies towards the left side of the x-axis. For example, a majority of houses have their rent less than 20,000 and the number of houses decreases as we go onwards from 20,000.

On the other hand, the values of the Size of a house seem to be a much more normal distribution when compared with that of Rent. This means that the values for the size of the house are more evenly distributed with a high concentration of values towards the middle of the x-axis with decreasing houses on both ends of the graph. The majority of houses are of size in the range of 500 to 1000.

As the distributions of Rent and Size are different to each other, we would expect a non-linear relationship between them.



Using the lmplot() function of seaborn, we plot a regression line with Rent on the y-axis, and Size on the x-axis to visualize the relationship between the rent and the size of a house. For plotting this graph, we have used a lowess type which stands for locally weighted linear regression [] which uses a non-linear approach to plot a regression fit.

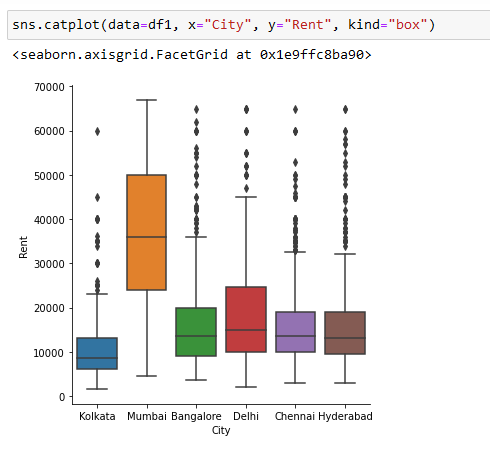
As expected, we see a trend of rent increasing as the size of a house increases. At first, when the size varies from 0 to 500, there is a relatively low increase in the rent.

Afterwards, for values of Size greater than 750, we see a sudden rise in the gradient meaning that for the same increase in Size, the Rent changes by a larger factor. The rate at which Rent increases with Size then becomes constant as we move towards houses with large values of Size.

To summarize from the above graph, for a house with a larger size, the Rent will increase. For relatively small houses (with Size < 600) we can expect the Rent to increase at a slower rate than for large houses. As the size of a house increase, it becomes more unaffordable for the general population.

**4.1.2 Which city is the most expensive to live in?**

Our dataset consists of a total of 6 cities: Kolkata, Mumbai, Bangalore, Hyderabad, and Delhi. We want to find that which city out of all these has on average, the most expensive to live in on the basis of rent prices.

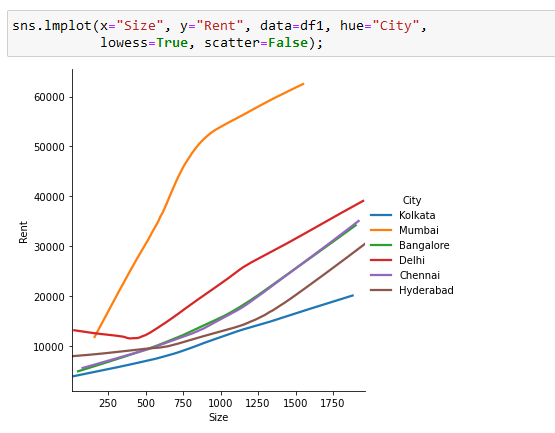


Using the catplot function of seaborn, we plot a boxplot for the Rent of houses for each City. A total of 6 boxplots are plotted on the figure as shown above with different colours representing each city.

By just looking at the graph, it is quite evident that the range of values for Rent of houses in Mumbai (highlighted by the color orange) are much higher than the rest of the cities. Infact, the first quarter of the values is at a higher level than almost all of the 3rd quartiles of the other cities.

This means that 75% of the houses located in Mumbai are expensive than 75% of houses located in Kolkata, Bangalore, Delhi, Chennai, and Hyderabad. Just based on this fact, we can confidently say that Mumbai is by far the city with the highest prices of house rents.

Other cities have less prominent differences in the rent prices when compared with Mumbai. Kolkata seems to have the cheapest rent prices on average, with Delhi being the 2nd most expensive city to live in on average.



To get a more clear idea about how the rent changes with each city, we have plotted a regression graph with Rent on the y-axis, Size on the x-axis, and the hue factor set to “City” as we are interested in comparing different cities.

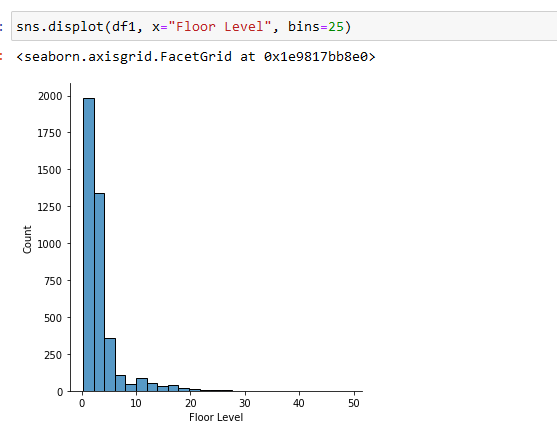
It is immediately apparent that rent prices for houses in Mumbai are on a much higher scale than other cities. Houses irrespective of their size cost more than houses located in other cities. Moreover, the rent climbs at a much faster pace in Mumbai as the Size of the house goes up.

From the graph, the order of the cities from the most affordable housing to unaffordable is: Kolkata, Hyderabad, Bangalore, Chennai, Delhi and Mumbai.

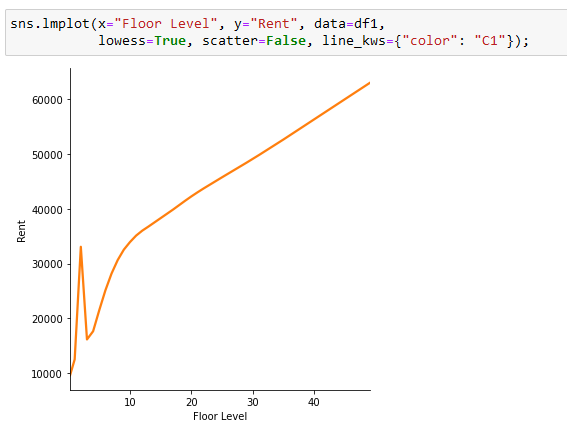
Another interesting point to note is that the cities of Chennai and Bangalore have an almost identical trend of increase in Rent with Size.

**4.1.3 Does the floor level of a house affect its rent?**

Another interesting comparison that can be made is that how the rent of a house change with the floor level that it is on?



To get a general idea about the distribution of the values of Floor Level, we plot a histogram using the displot() function of Seaborn. It can be seen from the graph that the majority of the houses are located on a low floor level with most of the homes being located at the ground floor. Fewer houses are located at higher floor levels.



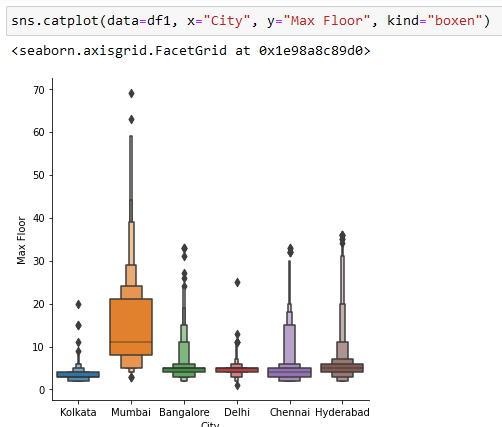
For visualizing the relationship between Rent and the Floor Level of a house, we plot a regression graph for Rent against Floor Level using the implot() function of Seaborn.

It can be seen that on average, there is a positive correlation between Rent and the Floor Level of a house. In general, houses located at higher floor levels are considerably more expensive than houses located on lower floor levels.

There is a sharp spike in the graph at around Floor Level 2 which doesn’t fit along with the rest of the curve. This may indicate that houses that are located on Floor Levels 2 and 3 are on average higher in demand than houses at the rest of floors. Another possible explanation for this could be errors while recording data for floor levels.

**4.1.4 How do the houses in each city compare with each other in terms of the size of the house and the number of floors?**

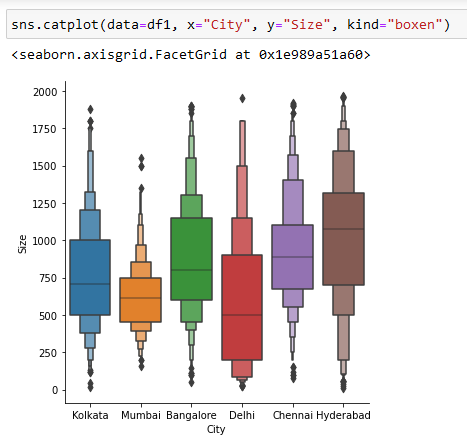
It can be safe to assume that the structure of the building of a house will be expected to vary region by region, and specifically cities as the number of floors and rooms in a house will vary with different geographical locations.



To determine if our hypothesis is true or not, we have visualized how the top floor of a house building changes with different cities. For this visualization, we have used the catplot() function of Seaborn with the kind parameter set to “boxen” to plot an enhanced box plot with Cities on the x-axis and the topmost floor of a house on the y-axis. This gives us a more clear view of the distribution of values for each city as using a simple boxplot would not have provided elaborate details.

We can see that each city has a different distribution of houses based on their topmost floor. On average, every city except Mumbai has the majority of houses with the top floor not exceeding 10 floors with the city with the lowest average top floor being Kolkata.

The city of Mumbai has the most share of high-storey buildings out of all the cities with the rest of the cities having relatively fewer tall buildings. Another observation that we can deduce is that the median top floor level in Mumbai is greater than 10, the vast majority of house buildings in Mumbai are in fact apartments/flats which can accommodate a large number of people. This makes sense as Mumbai is the most populated city located in India.



To analyze how the sizes of houses in different cities compare with one another, we have again plotted a boxen boxplot but in this case for Size against City.

Here, we see some interesting results as we can see that each city has its own variety of sizes for a building. We can notice that on average, the largest houses, in terms of size, occur in the city of Hyderabad whereas the houses located in Mumbai are on average the smallest out of all the cities.

In our previous analysis, we found out that Mumbai had the highest number of floors on average indicating that it has a large number of high-rise apartment buildings. In that context, it makes sense that it would have the lowest average house size because clearly, the size of an apartment is usually less than that of a villa or a bungalow.

The city of Delhi has the most diverse with the house sizes being spread over a large range of values.

**4.2 Summary of the results**

As a result of the data analysis which we have performed, we can summarize our findings in the following insights:

1. The affordability of a house decreases as its size (total area covered by the house) increases. Therefore large and even medium-sized houses would be realistically out of reach for an average citizen limiting them to houses that are relatively small.
2. The distribution of rent prices is concentrated below 25,000 meaning that the majority of the houses which were included in the dataset were comparatively inexpensive. We can speculate from this that the survey was undertaken in low to middle income areas.
3. We can conclude from our findings that out of all the cities in our dataset, Mumbai is by far the most expensive to rent a property as the rent prices in Mumbai are significantly more expensive than in other cities.
4. Kolkata has the most affordable housing available for rent compared to other cities as the rent prices are lower than all other cities in the dataset.
5. An overwhelming majority of houses in our data are located on either the ground floor or close to the ground. Fewer houses are located on high floors.
6. As we move to higher floors, we can expect the rents of a house to increase as based on our findings, houses located on higher floors are more expensive than those located closer to ground.
7. On average, houses in a city tend to be multi-storey buildings with the mean number of floors in a building being six.
8. The city with the highest percentage of high-storey houses is Mumbai with the majority of house buildings having more than 10 total floors. This indicates that Mumbai has a high population density.
9. In terms of size, Hyderabad has the largest houses with mean house size greater than 1000.

**4.3 Limitations of model:**

From what we gathered from research, the model that we used i.e. Gradient Boosting Regression was best suited for the dataset that we have been provided with as the dataset had more than 4700 rows in total and for that depth of data, the aforementioned model was best suited.

However good the model may be, there were still some areas that we think were constraining us from achieving best possible results. The main limitations that we encountered while training the model are as follows:

* While tuning hyperparameters, there were some variables that we just did not know much about as those were for advanced model tuning. Therefore, we were unable to properly tune those variables.
* Some of the data was reserved for testing the model after it had been trained which could not be used for training data. This may affect the model’s overall accuracy as some of the data would be left out while training.
* The model took a large amount of time to train on the training data due the limited hardware capability on the computer which we were training the model on.
* In order to save time, we increased the learning rate of the model training/fitting process however increasing it too much was leading to overfitting and a consequent loss in accuracy.

**Section-5: References**

1. housing.com (2022, September 23) *What is a BHK*

[*https://housing.com/news/what-is-a-bhk/*](https://housing.com/news/what-is-a-bhk/)

2. Aslanyan T. (2021, April 17) *Fundamentals Of Statistics For Data Scientists and Analysts*

<https://towardsdatascience.com/fundamentals-of-statistics-for-data-scientists-and-> data-analysts-69d93a05aae7

3. Pritha B. (2022, November 11) *How to Find Outliers | 4 Ways with Examples & Explanation*

<https://www.scribbr.com/statistics/outliers/>

4. <https://seaborn.pydata.org/tutorial.html>

5. https://scikit-learn.org/0.21/tutorial/index.html

**Section-6: Appendix**

**Distribution of tasks:**

**1. Group member 1: Toyabul Islam Foysal (SUKD1902057):**

* Writing down the questions to be answered from our research
* Importing the dataset
* Performing initial EDA
* Removing outliers
* Answering research question 1

**2. Group member 2: Humaira Khanom Adhora (SUKD2002013):**

* Feature engineering
* Encoding data
* Data pre-processing for model training
* Answering research questions 2, and 3

**3. Group member 3: ABDUL AZIZ SATTAR (SUKD1801980):**

* Training the model
* Answering research question 4