School of Chemistry, Chemical Engineering and Biotechnology

Division of Chemical and Biomolecular Engineering

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CB4247

Statistics and Computational Inference to Big Data

India's House Prices Prediction

Name:	Nathan Lawira
Date:	September 9, 2024

Table of Contents

List of Figures

List of Tables

List of Equations

Abstract

A dataset on housing prices in India was explored in this project. The dataset was extracted from Kaggle and analyzed using various Python libraries. Exploratory analysis and pre-processing of the data were conducted via visualization with graphs and removal of extreme outliers. Linear regression, specifically ordinary least squares regression, and various machine learning algorithms were trained on labeled data and used to predict unseen data. ANOVA and residual analysis were conducted for the linear regression model, while feature importance was conducted for the remaining algorithms to find and select the most critical variables. After the critical variables were identified and chosen, all models were retrained on the critical variables, and their performances were reevaluated.

1.0 Introduction

1.1 Aim

The aim of this project is to apply regression and machine learning techniques to analyze a complex real-world dataset and predict values for new data. This dataset contains data on house prices in India and includes multiple regressors and a response variable, also known as regressors and response, respectively. Machine learning algorithms will be trained and used to predict unseen data, and the performance of these algorithms on the dataset will be evaluated. Afterward, the best-performing algorithms will be combined, and their results will be evaluated.

1.2 Background

India's housing landscape, a subject of our study, is remarkably varied, from luxurious palaces once home to ancient maharajas, modern skyscraper apartments in vibrant cities, and simple huts in isolated rural areas. This variety displays the growth in India's housing sector that parallels the country's rising income levels. According to the Human Rights Measurement Initiative, India has achieved 60.9% of its potential in providing its citizens with the fundamental right to housing based on income. Regarding housing arrangements, renting, also known as hiring or letting, involves payments for the temporary use of a resource, service, or property owned by someone else. A gross lease is a specific type of rental agreement where the tenant pays a set rent, and the landlord covers all property-related costs, which may include tax payments and utility bills. Renting supports the principles of the sharing economy by allowing the shared use of assets, enhancing both efficiency and access to housing for diverse groups of people (Banerjee, 2022).

A dataset from Kaggle, titled “House Rent Prediction Dataset,” was used for the aims of this project. The dataset was extracted from Magicbricks, a website that provides service for all real estate needs, including property listings, rent payments, home loans, and expert advice. The dataset contains information on over 4700 property listings in India, with a rich number of attributes such as number of bedrooms, rent, property size, city, furnishing status, and others that will be explored and used to train a model in later sections (Banerjee, 2022).

This dataset was selected amongst other datasets due to its nature and usability score. It is suitable for analysis via linear regression and other forms of regression, as shown on the website, and matches the purposes of this project. A usability score on Kaggle displays the ease of use of a dataset based on completeness, credibility, and compatibility (Devrishi, *New usability rating on* *datasets*). Completeness is evaluated based on the presence of the dataset's title, tags, and description; credibility includes whether the dataset source is specified and traceable; and compatibility depends on whether the dataset’s format is compatible for use and sufficient information is provided for the file and columns. This dataset’s usability score is 10.00, achieving the maximum score in all three criteria mentioned previously, meaning that the dataset is clear, reliable, and easy to use. Therefore, it was selected for analysis in this project.

For this project, the dataset was analyzed using the Python programming language and various libraries, such as NumPy, Pandas, and scikit-learn, to assist with processing data, conducting regression, and training machine learning models. Students were initially required to select either Python or MATLAB for this project’s code, and the former was selected due to greater familiarity. The Python script was written using Jupyter Notebook.

2.0 Body

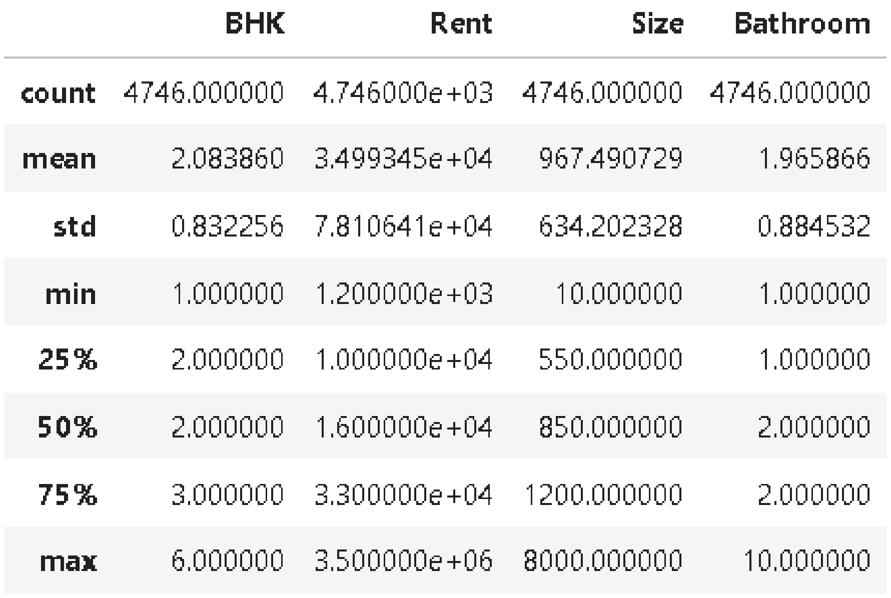
2.1 Exploratory Analysis & Data Pre-Processing

After being loaded into a data frame, the dataset was analyzed to find the column names indicating the dataset attributes. As shown below, there were 12 unique columns, which will be explored in the later sections. On the Kaggle site, the dataset was provided with a glossary that explains the columns (Banerjee, 2022):

- **BHK:** Number of Bedrooms, Hall, Kitchen.
- **Rent:** Rent of the Houses/Apartments/Flats.
- **Size:** Size of the Houses/Apartments/Flats in Square Feet.
- **Floor:** Houses/Apartments/Flats situated in which Floor and Total Number of Floors (Example: Ground out of 2, 3 out of 5, etc.)
- **Area Type:** Size of the Houses/Apartments/Flats calculated on either Super Area or Carpet Area or Build Area.
- **Area Locality:** Locality of the Houses/Apartments/Flats.
- **City:** City where the Houses/Apartments/Flats are Located.
- **Furnishing Status:** Furnishing Status of the Houses/Apartments/Flats, either it is Furnished or Semi-Furnished or Unfurnished.
- **Tenant Preferred:** Type of Tenant Preferred by the Owner or Agent.
- **Bathroom:** Number of Bathrooms.
- **Point of Contact:** Whom should you contact for more information regarding the Houses/Apartments/Flats.

Preliminary analysis showed no missing nor duplicated values. The 12 columns, referred to as variables, were categorized into numerical and categorical variables, as shown by having either “int64” or “object” data types, respectively. The numerical variables are *BHK*, *Rent*, *Size*, and *Bathroom*, while the categorical variables are *Posted On*, *Floor*, *Area Type*, *Area Locality*, *City*, *Furnishing Status*, *Tenant Preferred*, and *Point of Contact*.

Preliminary analysis was done on the numerical variables via descriptive statistics, as shown in Table 1 below. The mean of *Rent* and *Size* are larger than their respective medians, suggesting that these variables are positively skewed.



	BHK	Rent	Size	Bathroom
count	4746.000000	4.746000e+03	4746.000000	4746.000000
mean	2.083860	3.499345e+04	967.490729	1.965866
std	0.832256	7.810641e+04	634.202328	0.884532
min	1.000000	1.200000e+03	10.000000	1.000000
25%	2.000000	1.000000e+04	550.000000	1.000000
50%	2.000000	1.600000e+04	850.000000	2.000000
75%	3.000000	3.300000e+04	1200.000000	2.000000
max	6.000000	3.500000e+06	8000.000000	10.000000

Table 1 Descriptive statistics of the numerical variables

The numerical variables were analyzed using distribution, box, violin, and probability plots. The plots for *Rent* are shown below in Figure 1. As immediately evident in the box and probability plots (top right and bottom right, respectively), there was an extreme outlier on the rightmost side of both graphs. This extreme outlier was then removed to prevent adverse effects on the models. The graphs after the removal of this outlier are shown in Figure 2. The same graphs were plotted for Size, as shown in Figure 3 below. The distribution, box, and violin plots indicate that both *Rent* and *Size* are positively skewed, as confirmed by Table 1. According to (*1.3.3.21. Normal Probability Plot,* n.d.), the probability plot (bottom right) indicates that *Rent* slightly deviated from a normal distribution as some data points deviate from the straight line.

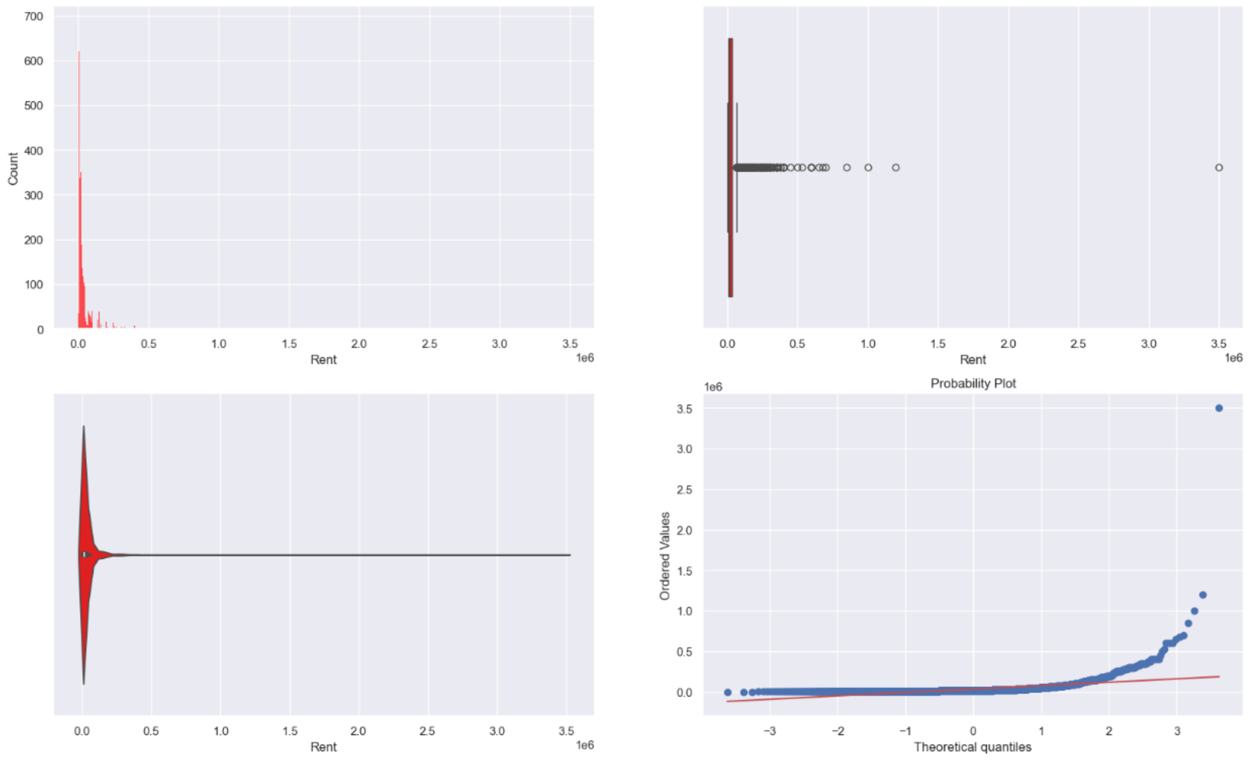


Figure 1 Distribution, box, violin, and probability plots for Rent

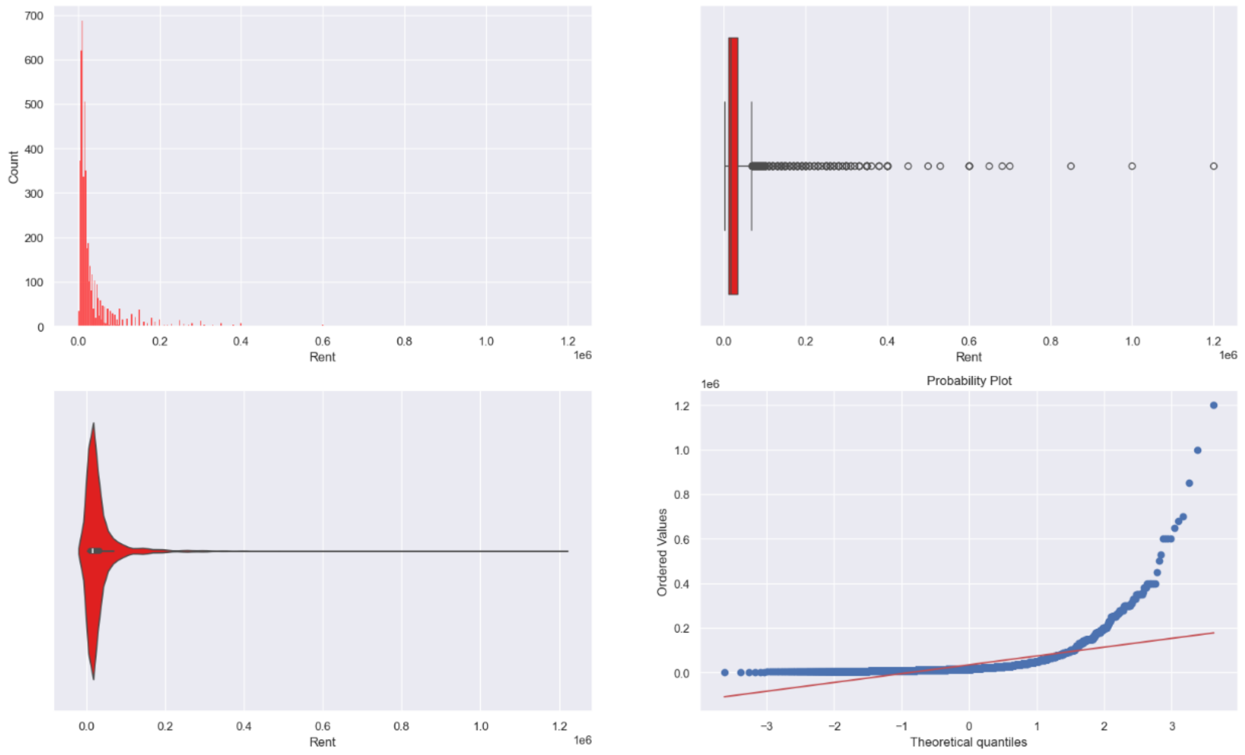


Figure 2 Distribution, box, violin, and probability plots for Rent after removing the extreme outlier

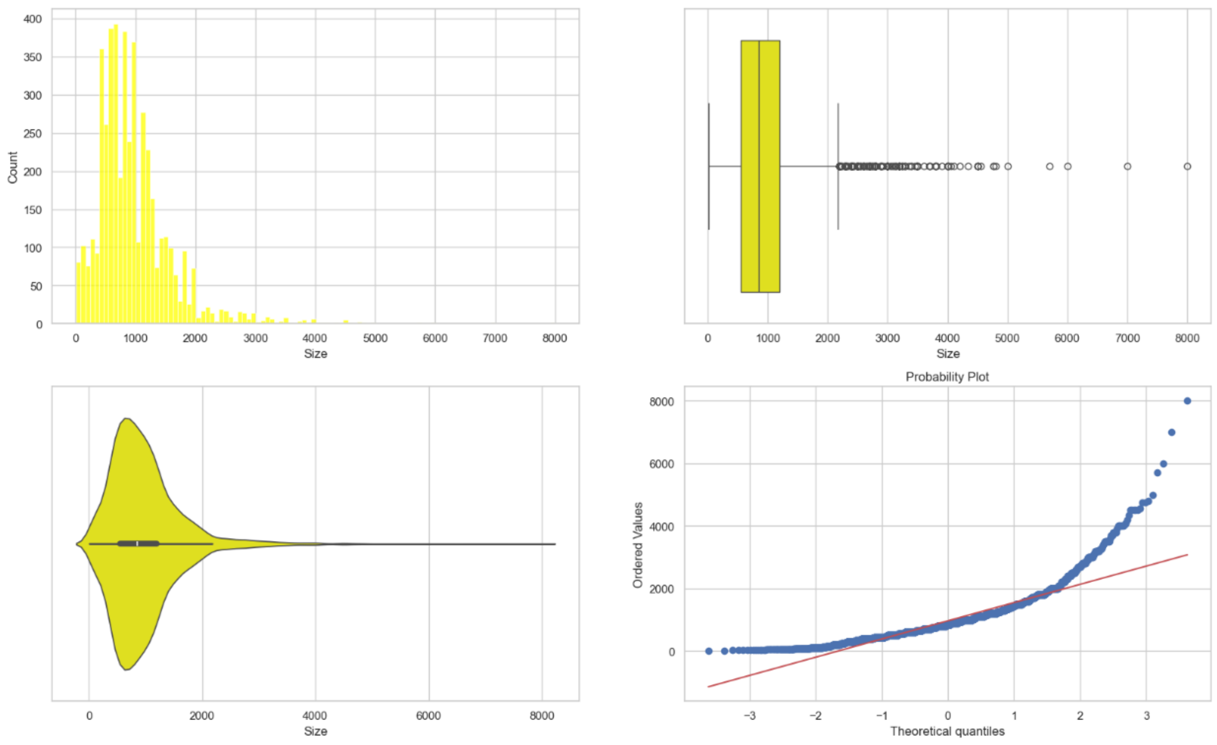


Figure 3 A distribution, box, violin, and probability plot for Size

For *BHK* and *Bathroom*, a count and a box plot were used instead since they took on discrete values in a smaller interval than *Rent* and *Size*. The plots for *BHK* and *Bathroom* are shown below in Figures 4 and 5, respectively, indicating that both variables are positively skewed. From the graphs below, most houses available for rent have a *BHK* of 2 and either 1 or 2 bathrooms, potentially indicating higher demand for smaller houses than bigger houses in India.

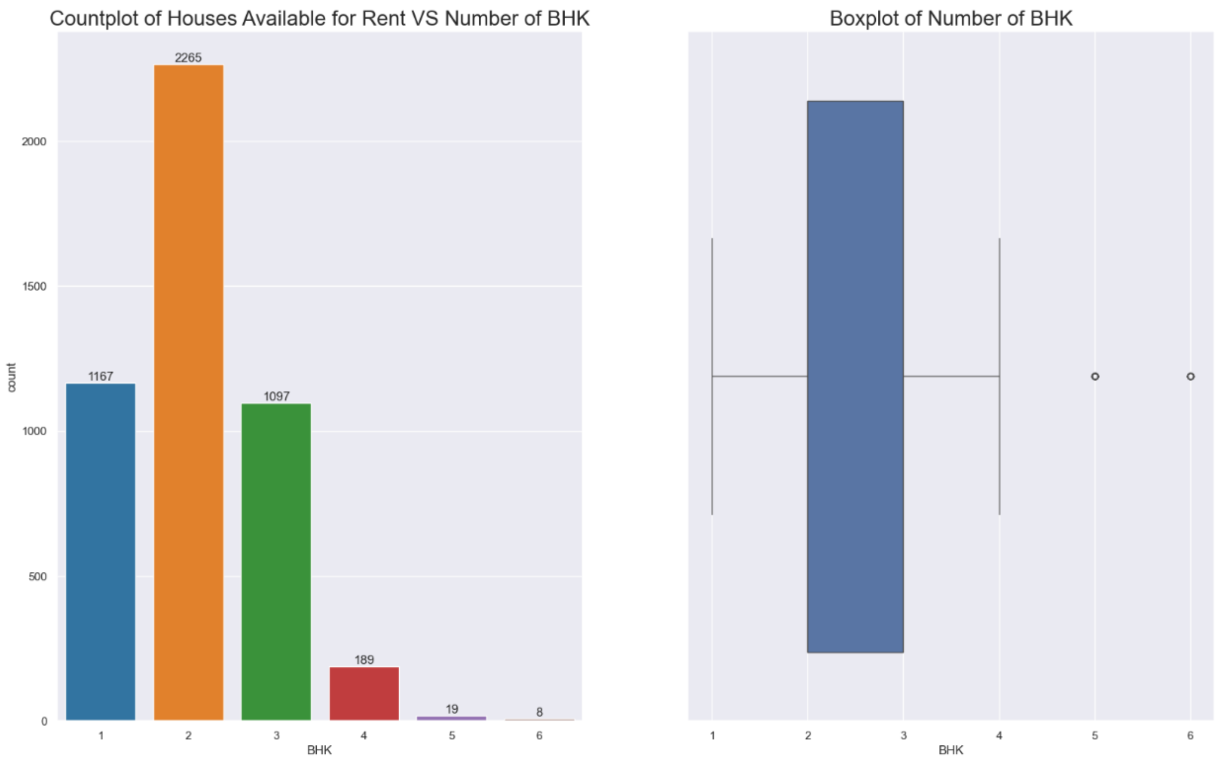


Figure 4 A count and a box plot of BHK

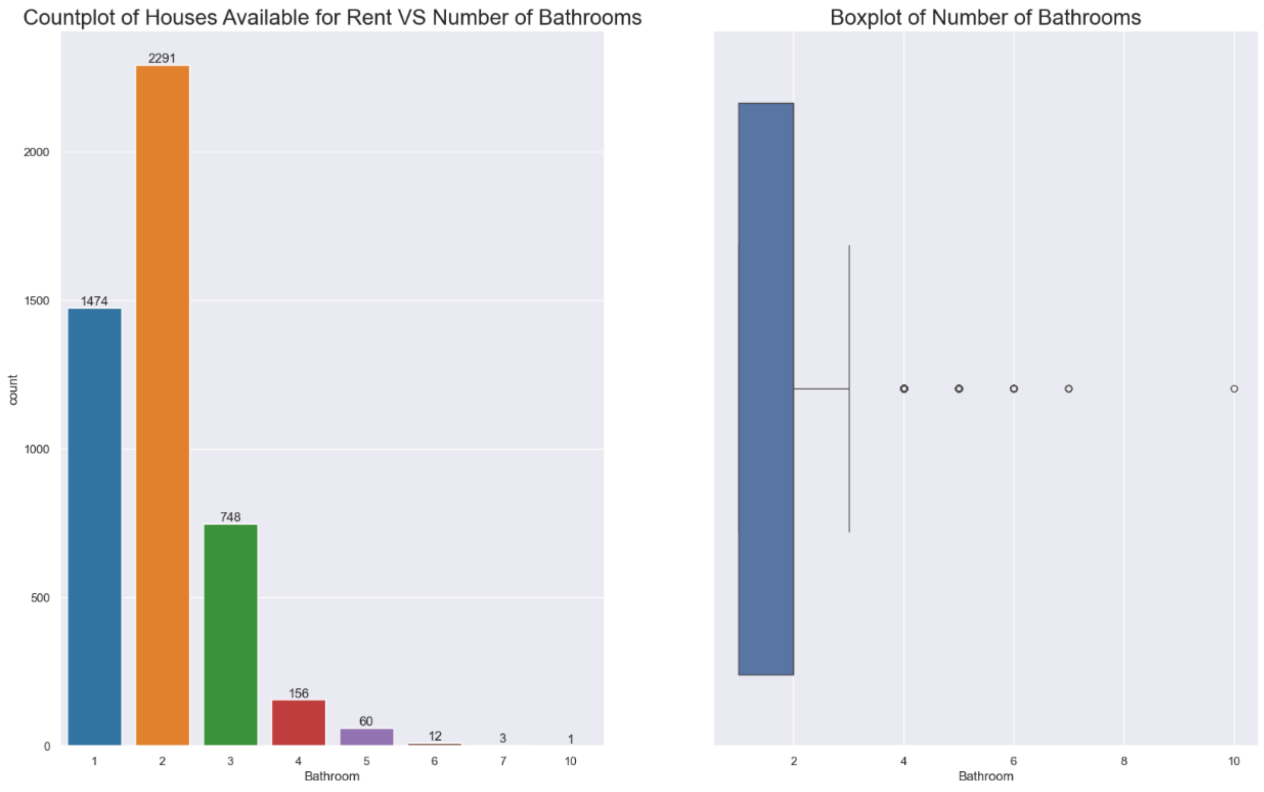


Figure 5 A count and a box plot of Bathroom

Afterward, an exploratory analysis was conducted on the categorical variables. Count plots via Seaborn were used to observe their trends. The counterplots for *City*, *Furnishing Status*, *Area Type*, *Tenant Preferred*, *Point of Contact*, and *Posted On* are shown below in Figures 6, 7, 8, 9, 10, and 11, respectively. Each graph will be discussed separately below.

As shown in Figure 6 below, the data of houses were extracted for houses available for rent in Kolkata, Mumbai, Bangalore, Delhi, Chennai, and Hyderabad only, not for other Indian cities. Mumbai had the highest number of houses available for rent, followed by Chennai. This may be because Mumbai and Chennai are two of India's most densely populated cities (Kolb, 2019) and the richest and most developed cities (NoBroker, 2024), which attracts people to work there and thus increases the demand for housing. Meanwhile, Kolkata had the lowest number of houses available for rent.

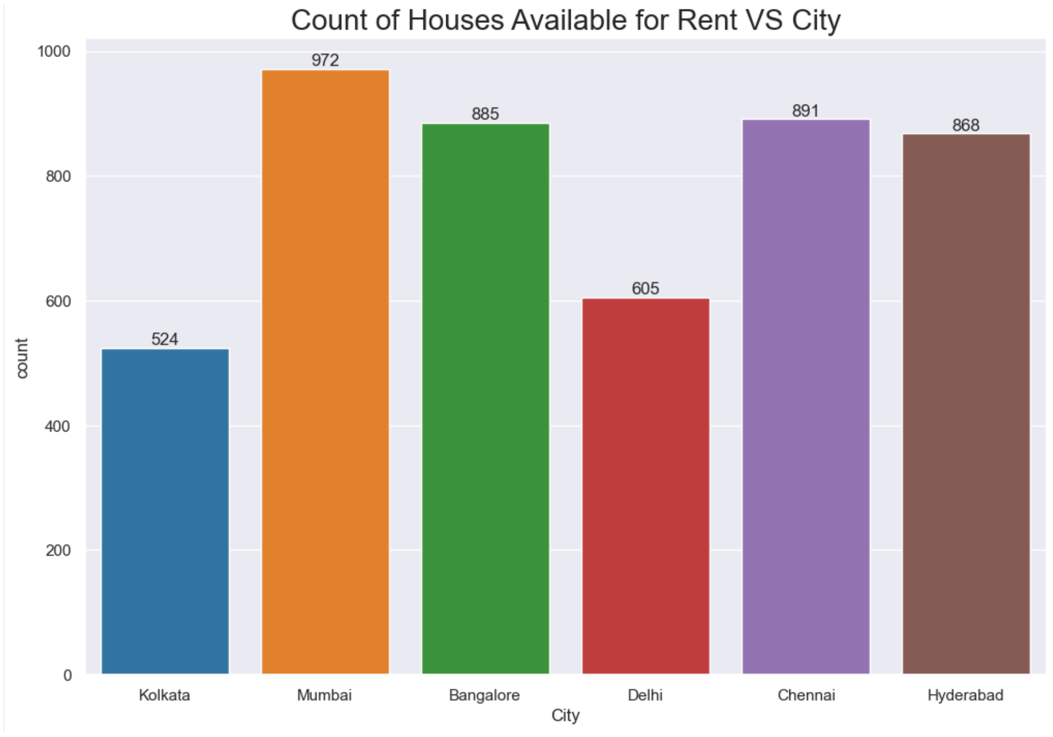


Figure 6 A count plot of houses for rent grouped by City

As shown in Figure 7 below, semi-furnished houses were the most common type, followed by unfurnished and furnished houses. This may indicate tenants' preferences for budget-friendly houses and the freedom to furnish them.

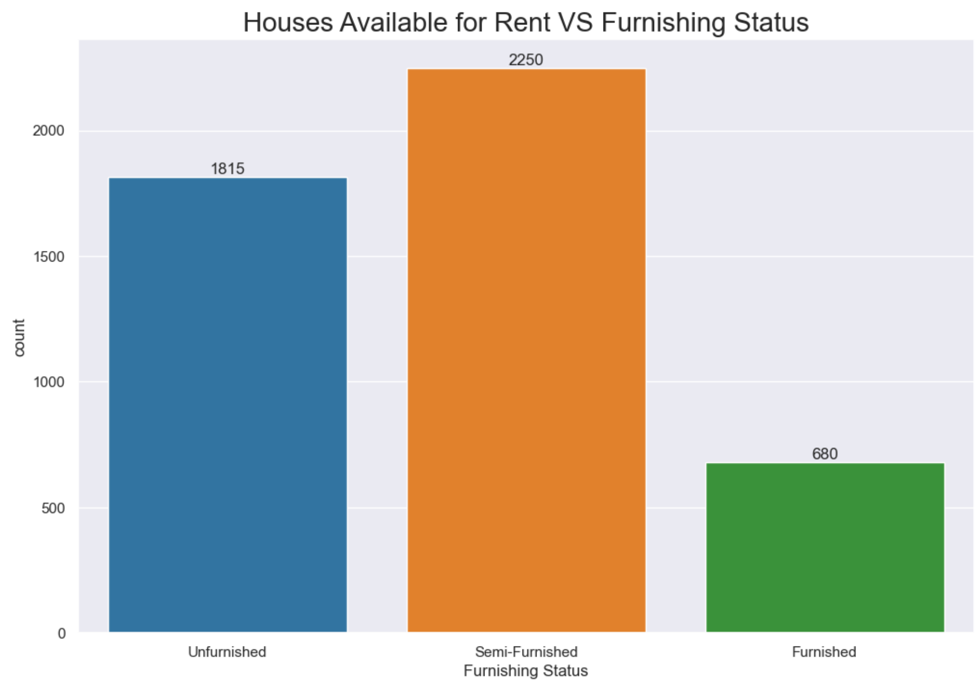


Figure 7 A count plot of houses for rent grouped by Furnishing Status

As shown in Figure 8 below, most houses' sizes were displayed in the super or carpet areas and scarcely built areas. Carpet area means all the floor area covered by a carpet, while the built house is the total carpet area added by the wall area. The super area includes the carpet area and the area of common spaces (Mishra, 2024). Since the houses listed using built area were much lower than the other two categories, these data points were outliers and thus removed.

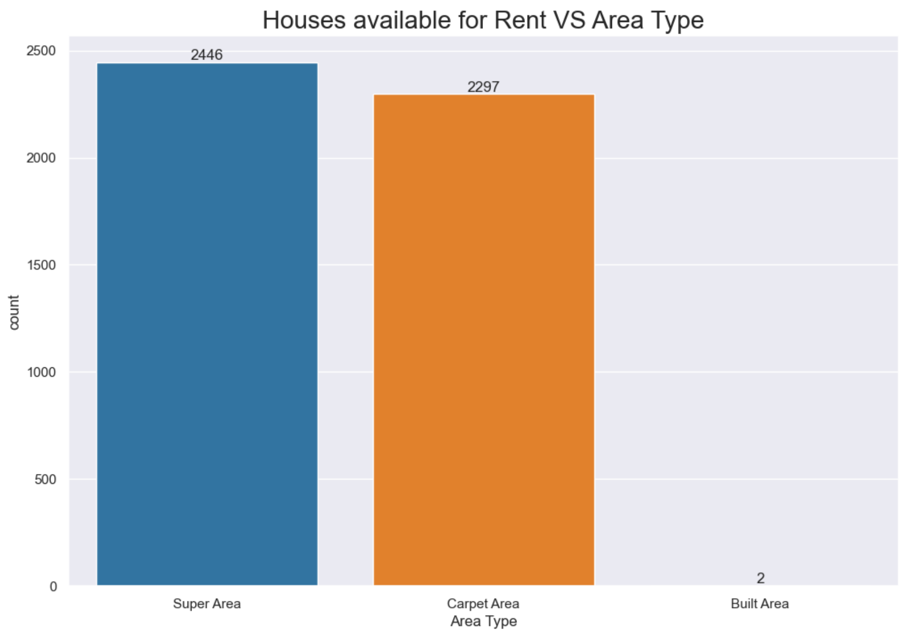


Figure 8 A count plot of houses for rent grouped by Area Type

As shown in Figure 9 below, most house owners, 3442 out of 4743, do not have a preference for tenants.

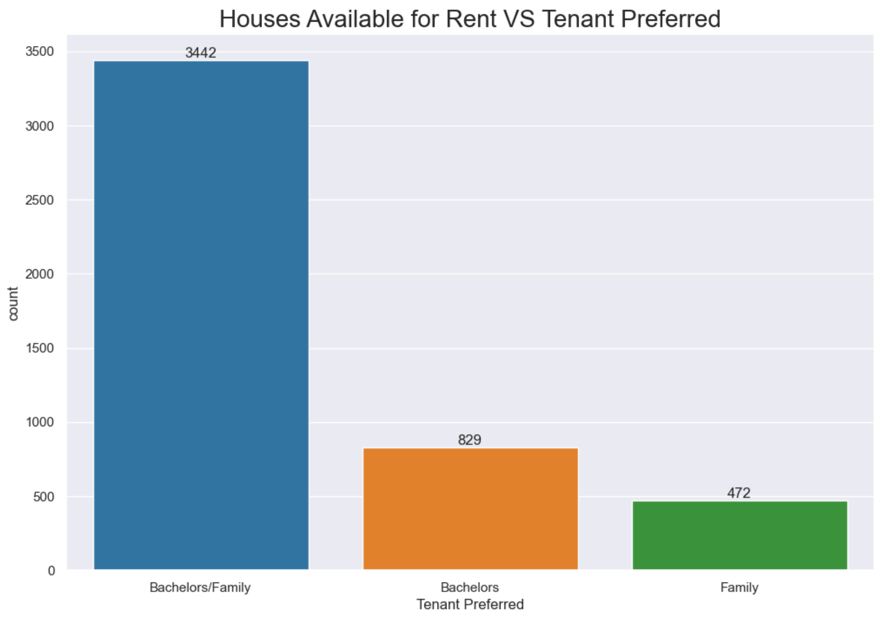


Figure 9 A count plot of houses for rent grouped by Tenant Preferred

As shown in Figure 10 below, the point of contact for most houses available for rent was the owner, followed by agents. This may indicate landowners' preferences to negotiate directly with tenants rather than through an agent. Since the number of houses preferring to contact builders was much less than the other two categories, this data point was removed as an extreme outlier.

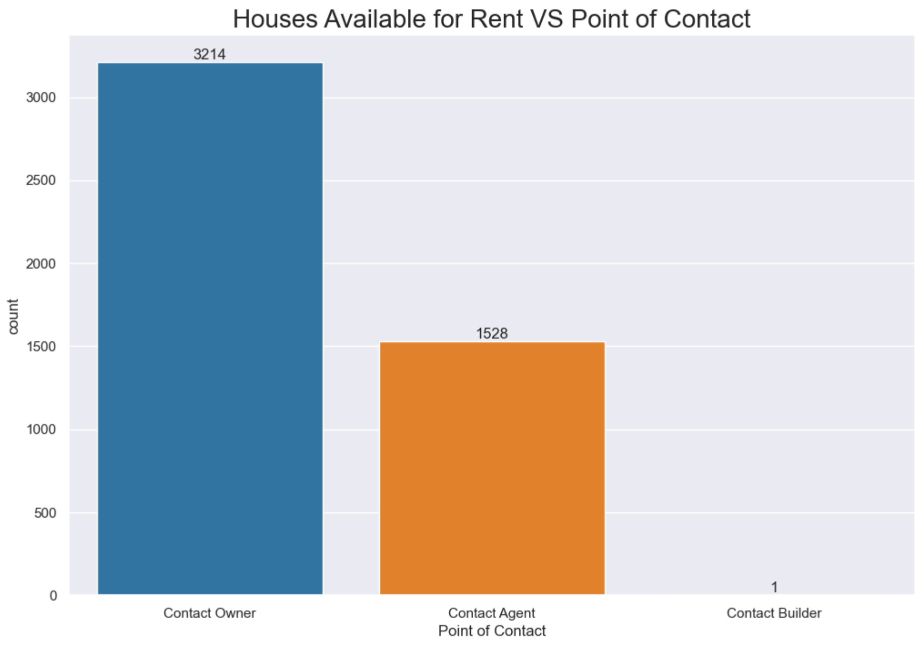


Figure 10 A count plot of houses for rent grouped by Point of Contact

As shown in Figure 11 below, *Posted On*, a categorical variable, had many unique values, with some having very little count. Thus, this variable in this state will not be useful in regression and training the machine learning models. To tackle this issue, *Posted On* was divided into four numerical variables, called *Day Posted*, *Day of the Week Posted*, *Month Posted*, and *Quarter Posted*, to preserve information and consider the potential effect of the posting date on rent prices. The year of posting is not considered because all listings were posted in 2022, so it is not a variable but a constant value. The code that achieved this is attached to Appendix A.

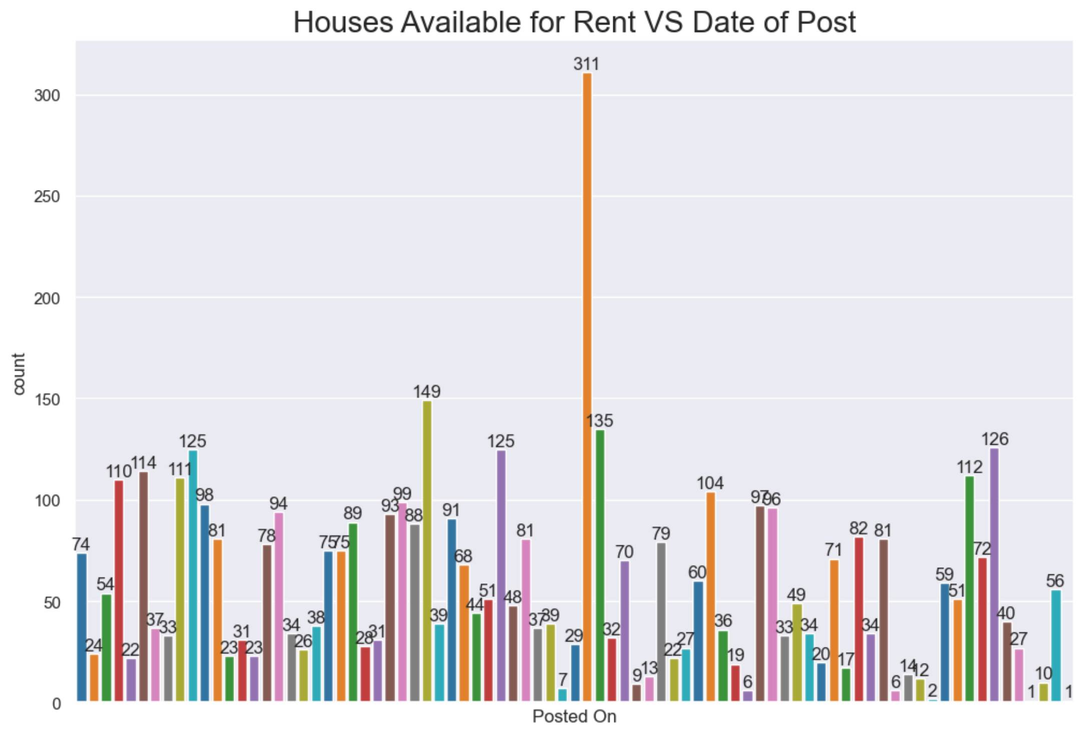
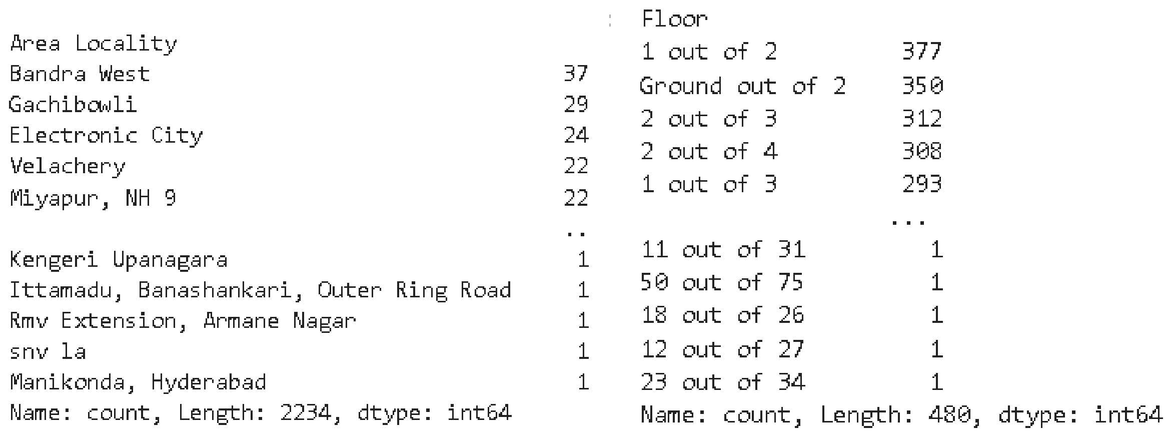


Figure 11 A count plot of houses for rent grouped by Posted On

Area Locality and *Floor* were not plotted on count plots as they had too many unique values, so running a Python script to plot a count plot for both was very time-consuming. Instead, their value counts were obtained using the methods shown below in Figure 12. Like *Posted On*, these two variables in this state had many unique values, with some having little count, so they will not be useful in regression and training the machine learning models.



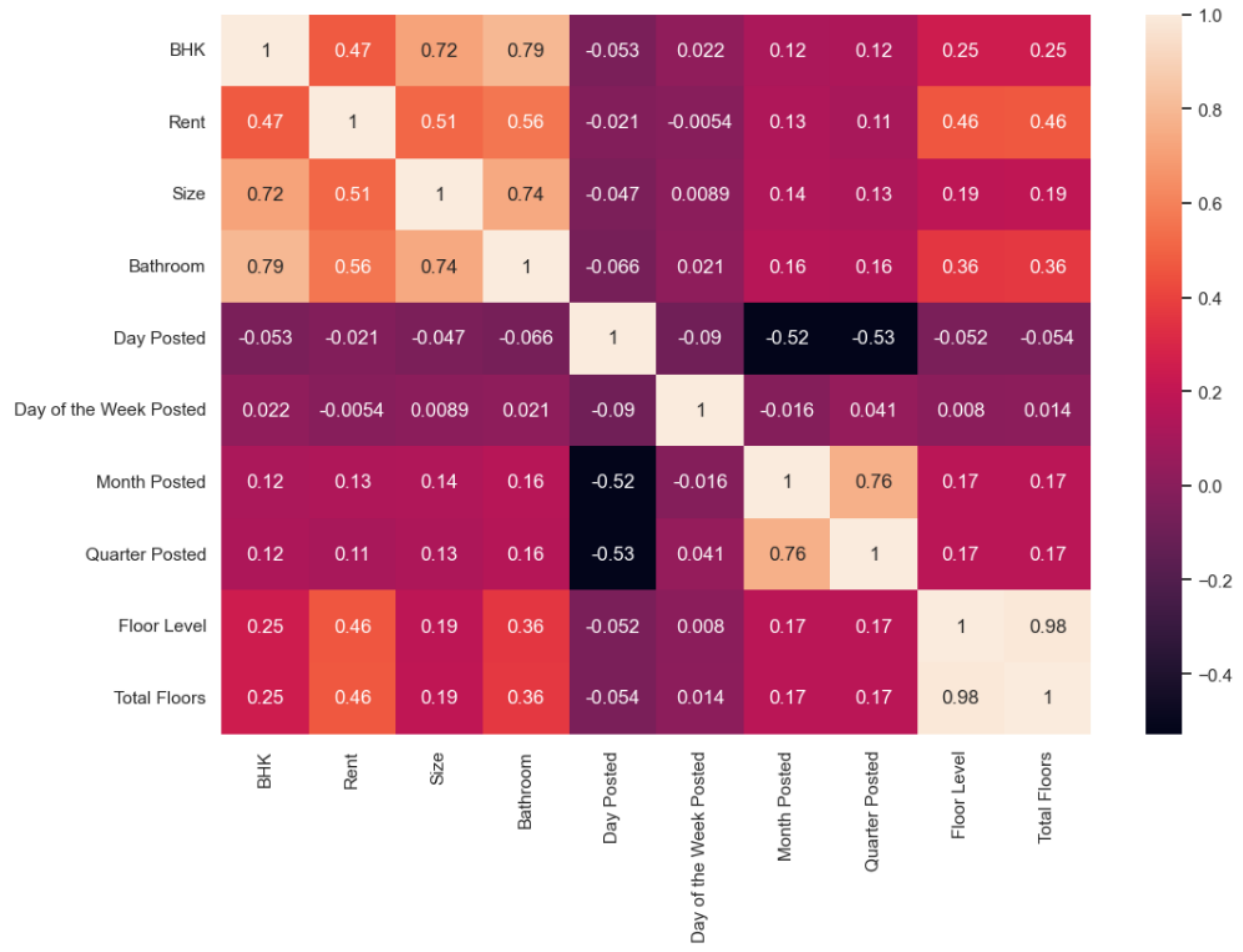
```
Area Locality
Bandra West                                  37
Gachibowli                                   29
Electronic City                              24
Velachery                                    22
Miyapur, NH 9                                22
                                             ..
Kengeri Upanagara                             1
Ittamadu, Banashankari, Outer Ring Road       1
Rmv Extension, Armane Nagar                   1
snv la                                        1
Manikonda, Hyderabad                          1
Name: count, Length: 2234, dtype: int64
```

```
Floor
1 out of 2          377
Ground out of 2     350
2 out of 3          312
2 out of 4          308
1 out of 3          293
                   ...
11 out of 31          1
50 out of 75          1
18 out of 26          1
12 out of 27          1
23 out of 34          1
Name: count, Length: 480, dtype: int64
```

Figure 12 Value counts of Area Locality (left) and Floor (right)

To tackle this problem, *Floor* was split into two numerical variables, *Floor Level* and *Total Floors,* by splitting each string value in *Floor* with “out of” as a delimiter. Thus, *Floor Level* contained the number preceding “out of” while *Total Floors* contained the number trailing “out of.” Four values in *Floor* did not have the “out of” string and were removed. *Area Locality*, on the other hand, was simply removed since converting this variable suitable for regression and machine learning is difficult. Afterward, all the duplicated values in the data were removed. The code that achieved this is attached to Appendix B.

A correlation between the numerical variables was obtained and plotted using a heatmap shown below in Figure 13. According to this heatmap, *Bathroom* had the highest correlation with *Rent*, followed by *Size* and *BHK*. The highest correlation was found between *Bathroom* and *BHK*, which is reasonable since the number of bathrooms is included in *BHK*. The next highest correlations are found between *Bathroom* and *Size*, then *BHK* and *Size*, which is also reasonable because as the number of rooms in the house increases, the area of the house should follow a similar trend. However, for this project, only the relationship between the response variable, *Rent*, and the regressors, the other variables, will be used to analyze and train machine learning algorithms, not the relationship between one regressor and another. Additionally, *Day Posted* and *Day of the Week Posted* had the weakest correlation with rent, with values close to 0.



	BHK	Rent	Size	Bathroom	Day Posted	Day of the Week Posted	Month Posted	Quarter Posted	Floor Level	Total Floors
BHK	1	0.47	0.72	0.79	-0.053	0.022	0.12	0.12	0.25	0.25
Rent	0.47	1	0.51	0.56	-0.021	-0.0054	0.13	0.11	0.46	0.46
Size	0.72	0.51	1	0.74	-0.047	0.0089	0.14	0.13	0.19	0.19
Bathroom	0.79	0.56	0.74	1	-0.066	0.021	0.16	0.16	0.36	0.36
Day Posted	-0.053	-0.021	-0.047	-0.066	1	-0.09	-0.52	-0.53	-0.052	-0.054
Day of the Week Posted	0.022	-0.0054	0.0089	0.021	-0.09	1	-0.016	0.041	0.008	0.014
Month Posted	0.12	0.13	0.14	0.16	-0.52	-0.016	1	0.76	0.17	0.17
Quarter Posted	0.12	0.11	0.13	0.16	-0.53	0.041	0.76	1	0.17	0.17
Floor Level	0.25	0.46	0.19	0.36	-0.052	0.008	0.17	0.17	1	0.98
Total Floors	0.25	0.46	0.19	0.36	-0.054	0.014	0.17	0.17	0.98	1

Figure 13 A heatmap of the numeric variables' correlation

3.2 Regression and Modeling Methods

This project used linear regression, specifically ordinary least squares (OLS) regression, to conduct regression analysis on the dataset. It was selected over nonlinear regression for simplicity. A linear regression model was trained on the training data and predicted the response for the test data. Afterward, its performance was evaluated via the mean absolute error, mean squared error, root mean squared error, and R^2 values. ANOVA analysis was conducted to determine whether there was a significant statistical relationship between the response and at least one of the regressors. Also, residual analysis was performed to validate the assumptions of using OLS, particularly whether the error follows a normal distribution with constant variance. The results of these analyses will be discussed in the "Results and Discussions" section.

After OLS regression analysis, various machine learning algorithms were trained on the same training data and predicted the response for the same test data. These algorithms are Ridge, Lasso, Bayesian Ridge, K-Nearest Neighbors, Decision Tree, Random Forest, Gradient Boosting, Support Vector, CatBoost, LightGBM, and XGBoost. Their performances were evaluated via the mean absolute error, mean squared error, root mean squared error, and R^2 values.

Rent was chosen as the response variable. In contrast, *Area Type*, *Size*, *BHK*, *Bathroom*, *City*, *Furnishing Status*, *Tenant Preferred*, *Day Posted*, *Day of the Week Posted*, *Month Posted*, *Quarter Posted*, *Floor Level*, and *Total Floor* were chosen as the regressors. The original dataset was split into the training and test dataset in a 0.8 to 0.2 ratio, specifying a random state of 42 to achieve the same results every time the script runs. Before the models were trained, the value of the numerical regressors was scaled because some models are sensitive to the magnitude of the values, namely K-Nearest Neighbors, Support Vector, Ridge, and Lasso (Jaadi, 2023) and transformed via Box-Cox transformation to convert them into a normally-distributed variable. Additionally, dummy variables were used to represent categorical variables by creating Boolean variables representing each unique categorical value (Garavaglia et al., n.d.). The final list of variables used for modeling is shown below in Table 2.

Response Variable	**Regressors**				
Rent (y)	BHK (x_1)	Size (x_2)	Bathroom (x_3)	Day Posted (x_4)	Day of the Week Posted (x_5)
	Month Posted (x_6)	Quarter Posted (x_7)	Floor Level (x_8)	Total Floors (x_9)	Area Type_Super Area (x_{10})
	City_Chennai (x_{11})	City_Delhi (x_{12})	City_Hyderabad (x_{13})	City_Kolkata (x_{14})	City_Mumbai (x_{15})
	Furnishing Status_Semi Furnished (x_{16})	Furnishing Status_Unfurnished (x_{17})	Tenant Preferred_Bachelors/Family (x_{18})	Tenant Preferred_Family (x_{19})	Point of Contact_Owner (x_{20})

Table 2 List of the response variable and regressors used for modeling

After the machine learning algorithms were trained, their performance metrics were calculated and observed, and high-performing algorithms were identified and selected for feature importance analysis. Feature importance is an important step in improving machine learning models where variables that have a larger effect on the model are given a higher score. Lower scoring variables can be removed to reduce the dimensionality and speed up the model, thereby improving the model's performance (Shin, 2023). In this project, features of high importance across these models were chosen, and the selected models were retrained on these features alone. The performance of these new models was re-evaluated via the mean absolute error, mean squared error, root mean squared error, and R^2 values. The R^2-adjusted values will be used as a metric along with the errors and R^2 values, before and after feature importance, to identify whether removing regressors with lower scores improved the model (Team, 2023).

3.0 Results and Discussions

3.1 Linear Regression

A linear regression model was fitted using the *LinearRegression()* method from the sklearn library. The intercept and coefficients of the fitted model were retrieved, and the equation is shown below.

$$\begin{aligned} y = 34864.28 &+ 2709.94x_1 + 24686.72x_2 + 7308.70x_3 - 755.93x_4 - 715.74x_5 + 156.40x_6 \\ &- 1810.20x_7 + 1244.84x_8 + 6928.93x_9 - 2355.09x_{10} - 670.07x_{11} \\ &+ 5260.05x_{12} - 4267.51x_{13} + 866.17x_{14} + 19585.53x_{15} - 5286.31x_{16} \\ &- 4232.97x_{17} + 2325.08x_{18} - 968.22x_{19} - 1710.53x_{20} \end{aligned}$$

Equation 1 Fitted linear regression model equation

The performance metric of this linear regression model is shown in Table 3 below. From the R^2 value shown below, the regressors in this model can explain roughly 56.06% of the variance in the response variable.

Metrics	**Values**
Mean Absolute Error (MAE)	21125.56
Mean Squared Error (MSE)	1256192283.59
Root Mean Squared Error (RMSE)	35442.80
R^2 Value	0.5606

Table 3 Linear regression model's performance metrics

After the linear regression model was fitted, an ANOVA analysis was conducted to determine whether a significant linear relationship existed between the response and at least one of the regressors (*Lesson 3: SLR Evaluation | STAT 462*, n.d.). This is done by first assuming the response variable follows an F distribution, of which the value can be calculated by dividing the mean squared error (MSE) from the regression mean squared (MSR) shown in equations 2 and 3 below (*3.5 - the Analysis of Variance (ANOVA) Table and the F-test | STAT 462*, n.d.). The final F value, F_0, shown in equation 4, will determine whether to reject the initial assumption. y_i is the actual value of each sample, $\bar{y}$ is the sample mean of y, $\hat{y}_i$ is the predicted value of each sample of y by the linear regression model, n is the number of samples, p is the number of estimators, and k is the number of regressors.

$$MSE = \frac{\sum(y_i - \hat{y_i})^2}{n - p}$$

Equation 2 Mean Squared Error for ANOVA

$$MSR = \frac{\sum(\hat{y_i} - \bar{\hat{y}})^2}{k}$$

Equation 3 Regression Mean Squared for ANOVA

$$F_0 = \frac{MSR}{MSE}$$

Equation 4 F_0 value

The number of regressors, k, was 20 as more variables were created due to dummy variables. The number of estimators, p, was 21, as there was an intercept in the linear regression model. The number of samples, n, was the number of rows of data points, which was 4722. Inputting all these values along with the values of y_i, $\hat{y}_i$, and $\bar{y}$ into equations 2, 3, and 4, the final F_0 value was 258.86.

To determine the conclusion of the ANOVA analysis, the F critical value must be determined with a right-tail probability, α, of 0.05. This was easily found via the *f.ppf()* method from Python's scipy.stats. The critical value found was 1.573, much lower than the F_0 value, 258.86. Therefore, it can be sufficiently concluded that there is a significant linear relationship between the response variable and at least one of the regressors.

Afterward, residual analysis of the linear regression was conducted by obtaining the standardized residual value using Equation 5, shown below. d_i represents the standardized residuals, and σ is the standard deviation. This ensures that the residuals follow a standardized normal distribution with a mean of 0 and a standard deviation of 1. Given that roughly 99.7% of data points lie between the interval -3 and 3 in a standardized normal distribution (Libretexts, 2021), this step allows simple identification of outliers, which will be explored later.

$$d_i = \frac{y_i - \hat{y}_i}{\sigma}$$

Equation 5 Standardized residuals equation

The standardized residuals were plotted in three different plots, as shown in Figure 14 below. The first is a distribution plot of the residuals, showing that they roughly follow a normal distribution. The second is a scatter plot where the predicted values are plotted against the actual values with a 45° line. Some points are pretty close to the line, yet a lot deviates quite far, indicating that the model has room for improvement. The third is another scatter plot where the standardized residuals are plotted against the predicted value. Since standardized residuals are calculated by the difference between the predicted and actual values, a positive value indicates that the predicted values are too low, while a negative value indicates that the predicted values are too high. A value of 0 indicates that the model made a perfect prediction. The third plot shows that most points do not cluster close to the line, indicating that the model can be improved to make more accurate predictions. Additionally, there is a clear downward trend, showing that the model tends to predict lower for houses with lower prices and predict higher for houses with higher prices, which is confirmed by the second plot.

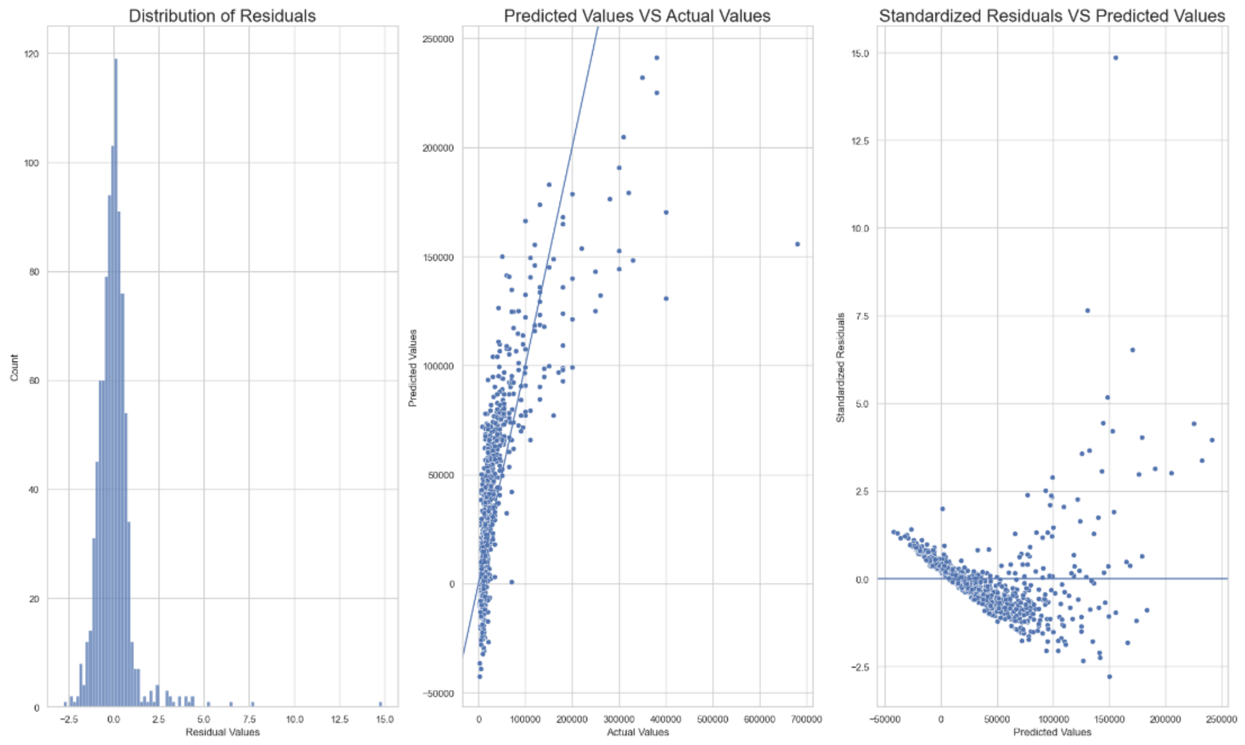


Figure 14 Residual plots

From the first plot shown above, it is assumed that the residuals roughly follow a normal distribution, which means that the error/noise is approximately normally distributed with constant variance. Since the residuals are standardized, 99.7% of the points should be within the interval [-3, 3]. Using Python, 98.25% of the residuals were found to lie within [-3, 3], indicating that there were outliers that may affect the model's performance. These points can be removed to improve the accuracy of the Linear Regression model.

3.2 Other Machine Learning Models

Since many machine learning models assume a normal distribution, the response variable *Rent* was Box-Cox transformed before training the models. The plots for *Rent* after the Box-Cox transformation are shown below in Figure 15. As evident by comparison with Figure 2, the transformed data is closer to a normal distribution.

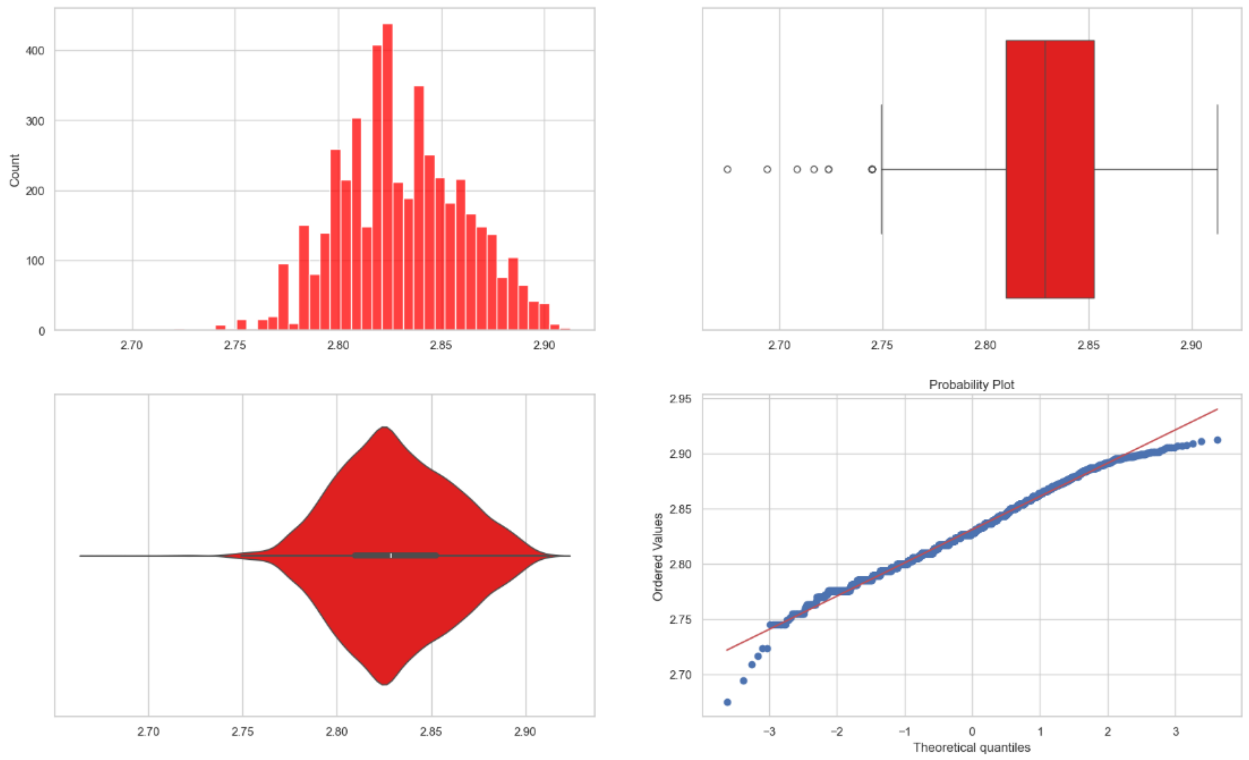


Figure 15 Distribution, box, violin, and probability plots for Rent after Box-Cox transformation

After the Box-Cox transformation of *Rent*, the data is split into train and test data in an 8:2 ratio, and the models are trained. The performance of each model on the test data is shown below in Table 4.

	Mean Absolute Error (×10^{-2})	**Mean Squared Error (×10^{-4})**	**Root Mean Squared Error (×10^{-2})**	**R^2**	**R^2 Adjusted**
Ridge	1.098	2.18	1.475	0.757	0.751
Lasso	2.392	8.98	2.996	-0.005	-0.026
Bayesian Ridge	1.098	2.18	1.475	0.757	0.751
K-Nearest Neighbor	1.182	2.57	1.602	0.713	0.707
Decision Tree	1.380	3.48	1.865	0.611	0.602

Gradient Boosting	1.036	2.02	1.422	0.774	0.769
Random Forest	1.049	2.07	1.439	0.768	0.763
Support Vector	3.528	18.09	4.254	-1.025	-1.069
CatBoost	0.993	1.89	1.376	0.788	0.783
LightGBM	1.034	2.03	1.426	0.772	0.767
XGBoost	1.055	2.12	1.456	0.763	0.758

Table 4 Performance metrics of the machine learning models

Bar graphs are also plotted to compare each model's errors against each other. These graphs are displayed below in Figures 16 to 19. The graphs below show that Lasso and Support Vector are the worst-performing models, while CatBoost, Gradient Boosting, LightGBM, Random Forest, and XGBoost are consistently in the top 5 performing models. Thus, these models will be used for feature importance analysis.

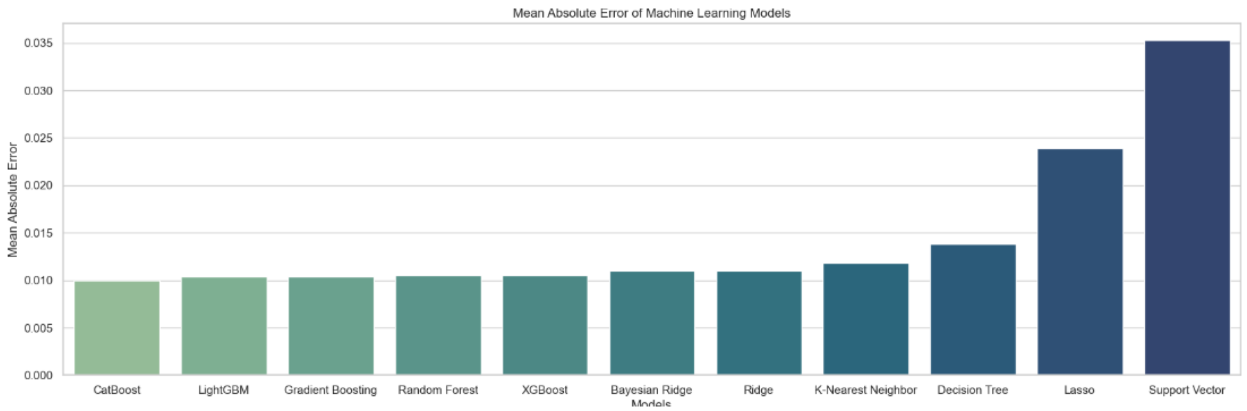


Figure 16 A bar graph of the mean absolute error (MAE) of trained models

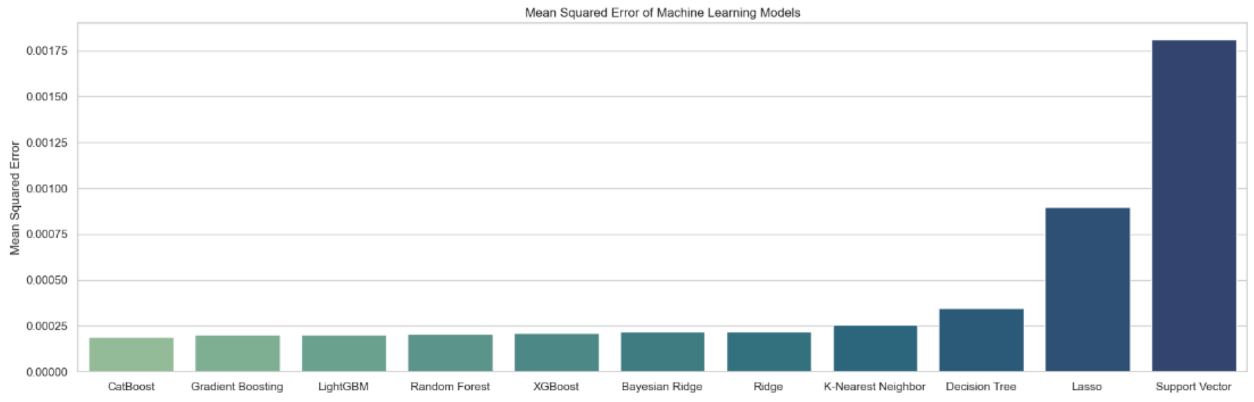


Figure 17 A bar graph of the mean squared error (MSE) of trained models

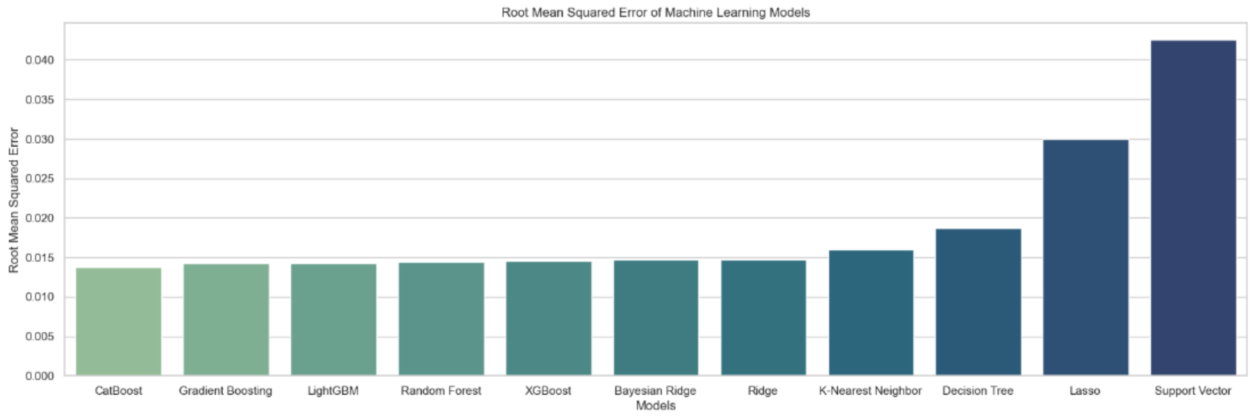


Figure 18 A bar graph of the root mean squared error (RMSE) of trained models

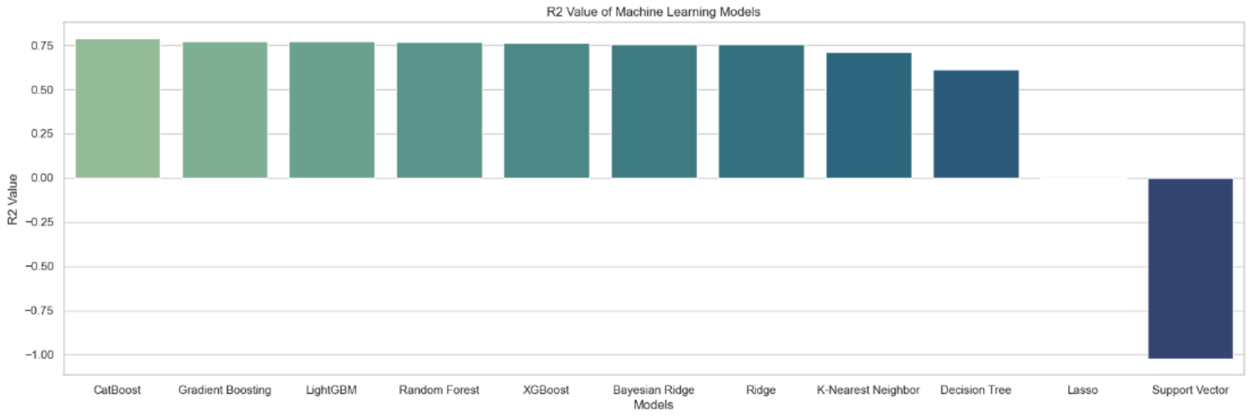


Figure 19 A bar graph of the R^2 value of trained models

3.3 Feature Importance Analysis

Feature importance analysis is conducted to understand the importance of individual features in the model’s decision-making process. The benefits include identifying the irrelevant features, improving models by reducing dimensionality, and interpreting the model to various stakeholders (Shin, 2023). Each feature’s scores on the model's decision-making process will be plotted in a horizontal bar graph. The feature importance graphs of CatBoost, Gradient Boosting, LightGBM, Random Forest, and XGBoost are shown below in Figures 20 to 24 respectively.

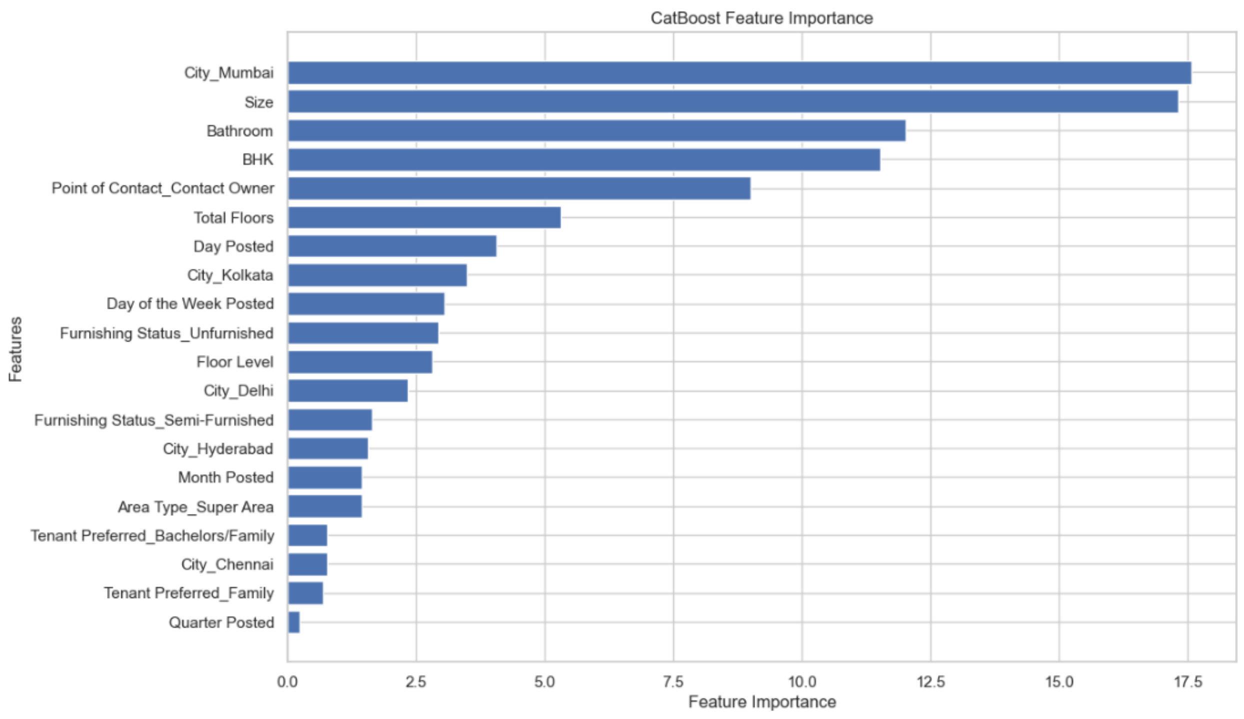


Figure 20 Feature importance graph for CatBoost

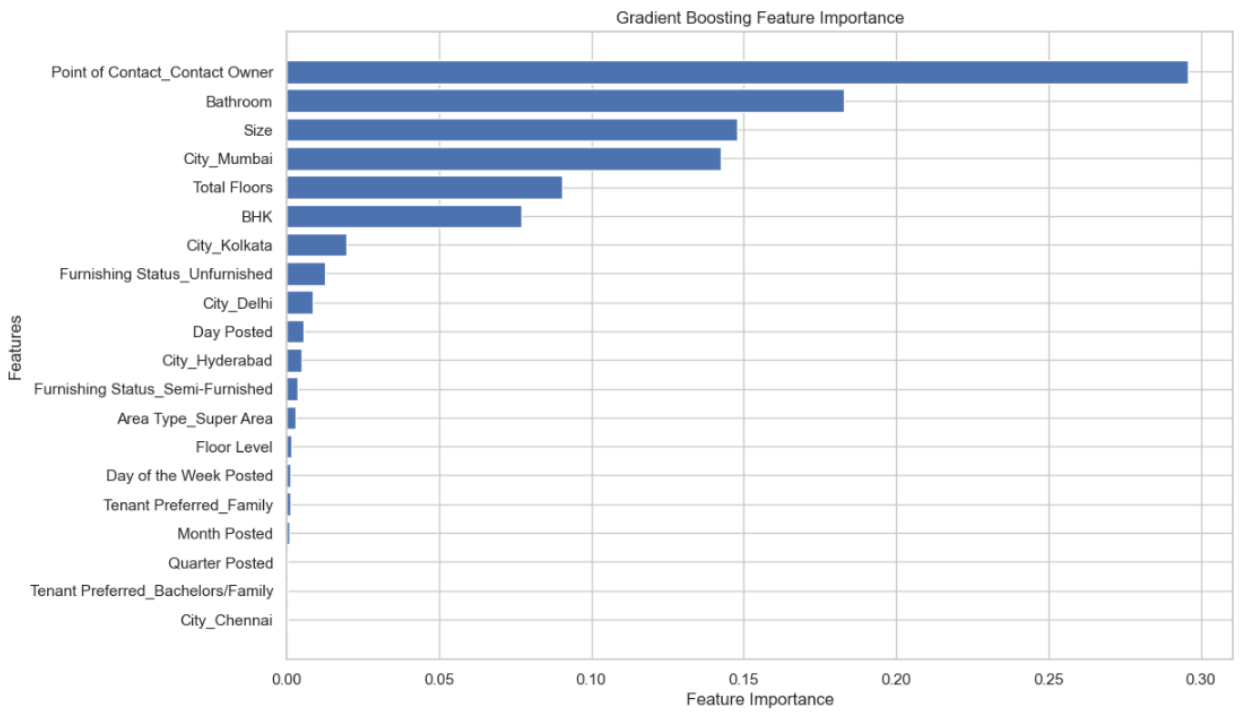


Figure 21 Feature importance graph for Gradient Boosting

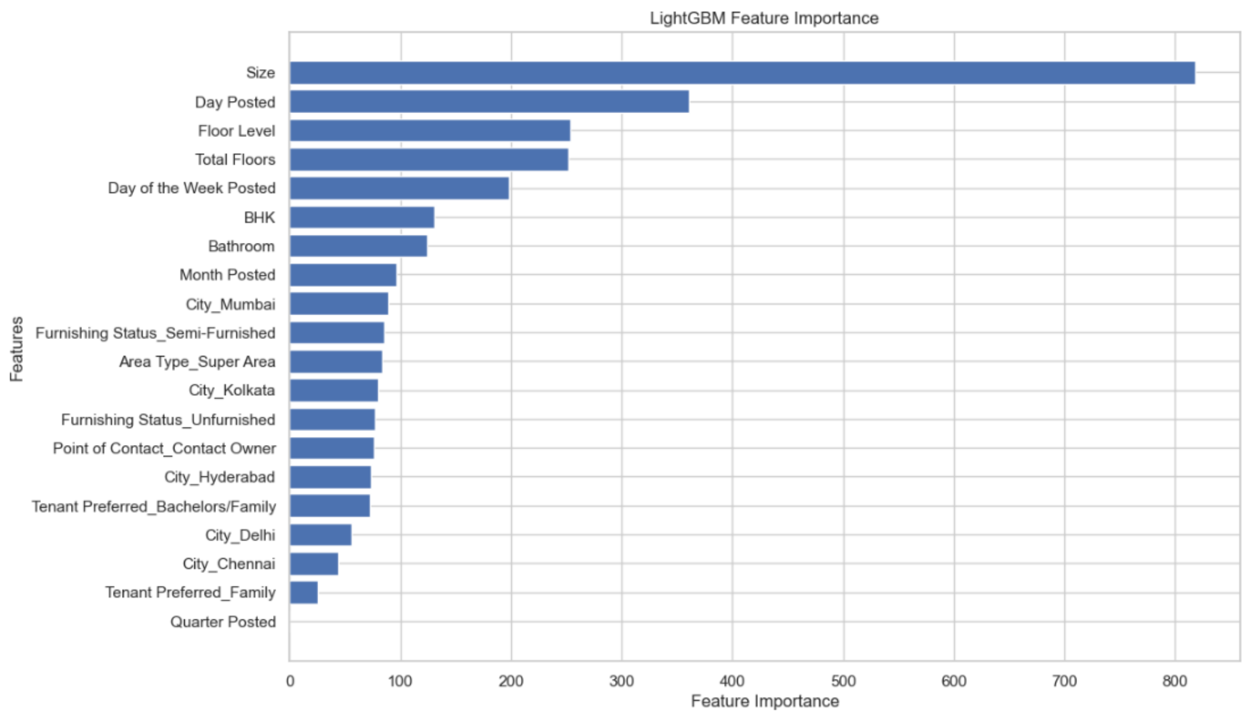


Figure 22 Feature importance graph for LightGBM

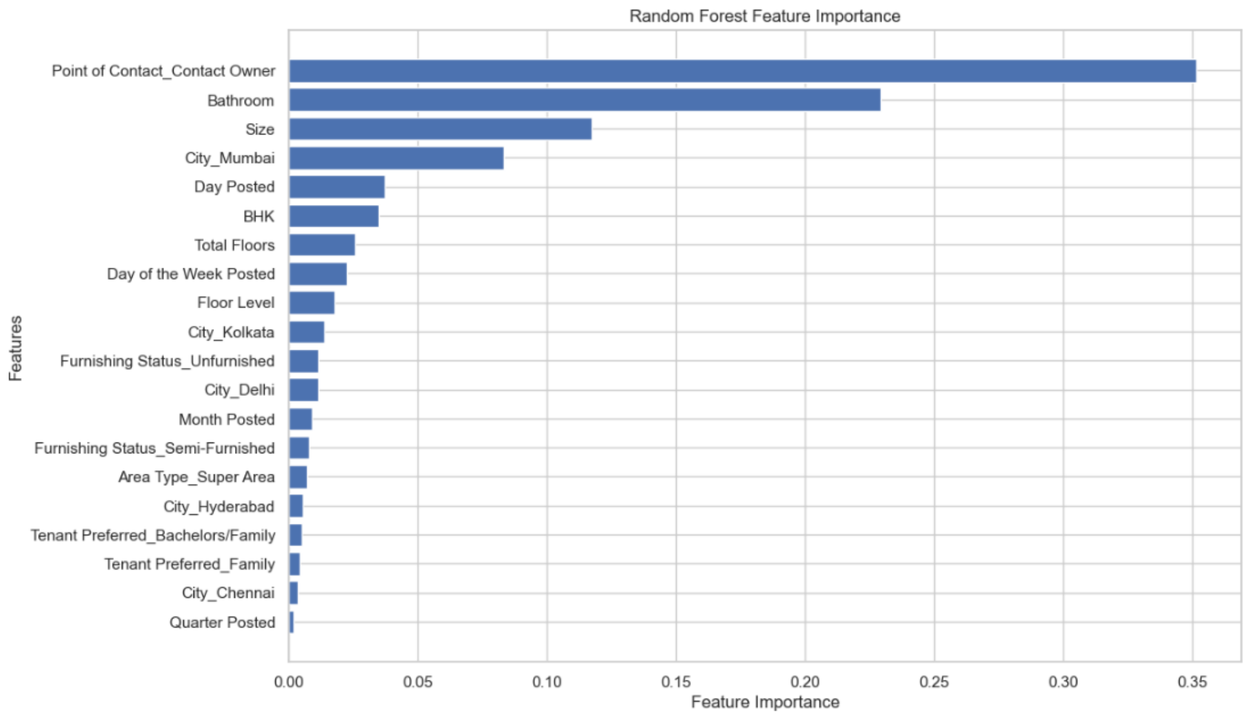


Figure 23 Feature importance graph for Random Forest

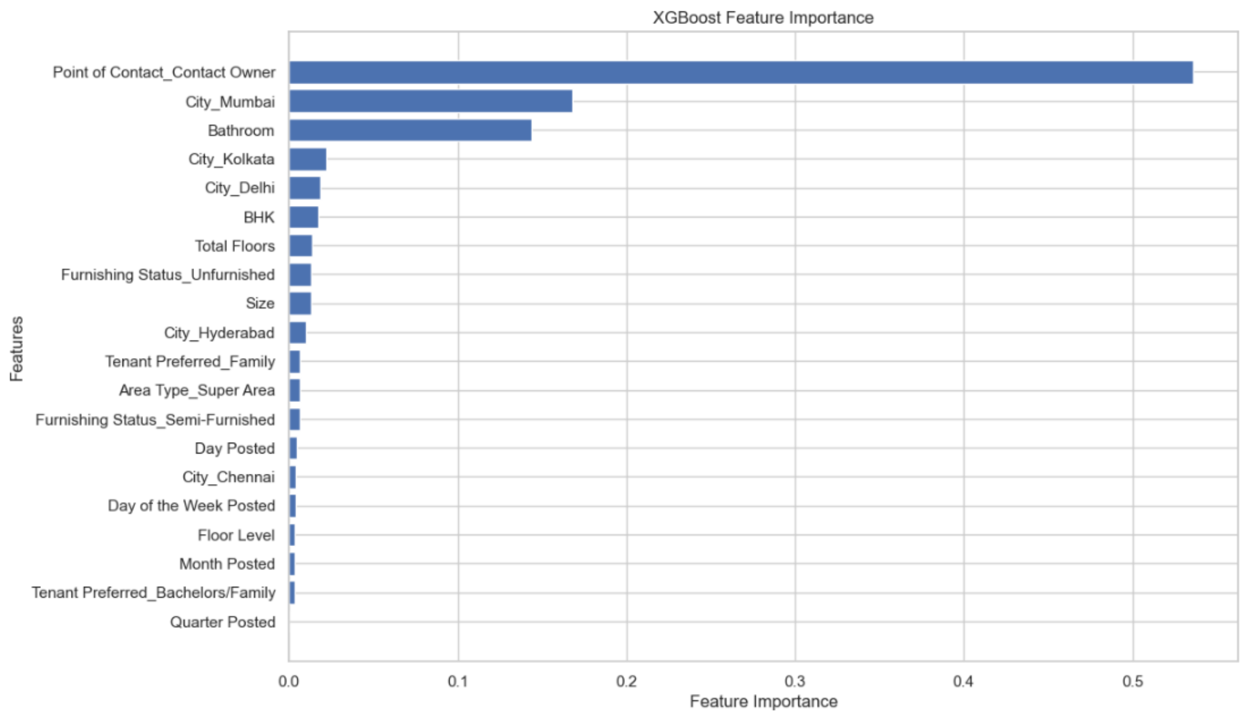


Figure 24 Feature importance graph for XGBoost

Table 5 summarizes each model's top 5 important features in descending order. According to the table below, *Point of Contact*, *Bathroom*, *Size*, and *City* are the most shared features across all models except for LightGBM. Thus, LightGBM will be excluded from the further analysis and validation of feature selection. *Area Type* will also be included in the validation testing since *Size* depends on it.

	1st Feature	**2nd Feature**	**3rd Feature**	**4th Feature**	**5th Feature**
CatBoost	City_Mumbai	Size	Bathroom	BHK	Point of Contact_Contact Owner
Gradient Boosting	Point of Contact_Contact Owner	Bathroom	Size	City_Mumbai	Total Floors
LightGBM	Size	Day Posted	Floor Level	Total Floors	Day of the Week Posted
Random Forest	Point of Contact_Contact Owner	Bathroom	Size	City_Mumbai	Day Posted

XGBoost	Point of Contact_Contact Owner	City_Mumbai	Bathroom	City_Kolkata	City_Delhi

Table 5 Top 5 important features of CatBoost, Gradient Boosting, LightGBM, Random Forest, and XGBoost

The models except LightGBM were then retrained on the five features previously selected. Table 6 below shows the performance summary of the four models before and after feature importance.

	Initial MAE ($\times10^{-2}$)	**Final MAE ($\times10^{-2}$)**	**Initial MSE ($\times10^{-4}$)**	**Final MSE ($\times10^{-4}$)**	**Initial RMSE ($\times10^{-2}$)**	**Final RMSE ($\times10^{-2}$)**	**Initial R^2**	**Final R^2**	**Initial R^2 Adjusted**	**Final R^2 Adjusted**
Gradient Boosting	1.036	1.081	2.02	2.21	1.422	1.485	0.774	0.753	0.769	0.751
Random Forest	1.049	1.159	2.07	2.54	1.439	1.595	0.768	0.715	0.763	0.712
CatBoost	0.993	1.077	1.89	2.23	1.376	1.494	0.788	0.750	0.783	0.748
XGBoost	1.055	1.109	2.12	2.36	1.456	1.536	0.763	0.736	0.758	0.733

Table 6 Performance of selected models before and after feature importance selection

As shown in Table 6, contrary to expectations, feature importance analysis did not result in more accurate models, as all models yielded higher errors and lower R^2 values. Furthermore, the final R^2 adjusted value decreases after feature importance, indicating that selecting critical features and removing non-significant features did not improve the models. Therefore, in merging high-performing models, all regressors will be used to test the combined model.

3.3 Merged Model Testing

The top five performing models, CatBoost, Gradient Boosting, LightGBM, Random Forest, and XGBoost, were combined, and their performance was tested against the original, non-Box-Cox transformed data. Table 7 below shows the results of the testing. The combined model performed reasonably well, explaining 77.54% of the variability in the data. Although this R^2 value is less than that of standalone CatBoost, CatBoost was tested against the Box-Cox transformed data, which means that it will yield a more accurate result due to the normalized data. Therefore, this combination of models performed sufficiently well, with an R^2 of 0.7754, and will produce an adequately accurate result in predicting Indian house rent prices.

	Mean Absolute Error ($\times 10^3$)	**Mean Squared Error ($\times 10^8$)**	**Root Mean Squared Error ($\times 10^4$)**	**R^2 Value**
Combined Model	9.423	6.422	2.534	0.7754

Table 7 Performance metrics of the combined model

3.4 Recommendations

The following recommendations may be considered for future projects to improve regression analysis and the performance of the machine learning models.

1. Different pre-processing techniques can be applied, such as removing outliers beyond the interquartile range. This technique may yield a more accurate result since it is more robust to outliers and does not depend on the assumption that data is normalized. However, caution must be taken when using this technique, as valid data points may be removed if the data is heavily skewed.
2. Other machine learning algorithms beyond the ones used here can be explored and used for prediction. Algorithms such as Artificial Neural Networks tend to be more robust against data that deviate from linearity and may perform better than the models used in this project.
3. Other regression analyses, such as weighted least squares and nonlinear regression, can be used in addition to ordinary least squares regression analysis. Since the data deviates from a normalized distribution, ordinary least squares regression did not perform well, and more complex regression analysis techniques should be employed to account for the data's nonlinearity.

4.0 Conclusion

The aims of this project were met by deploying ordinary least squares regression and various machine learning models. The models were trained on supervised data and used to predict unseen data. Their performances were evaluated using mean absolute error, mean squared error, root mean squared error, and R^2 values. ANOVA analysis was also conducted on the ordinary least squares model, and it was determined that there is a significant relationship between at least one of the regressors and the response variable, albeit with mediocre performance. A residual analysis was also conducted to determine whether the assumption holds true for the error term following a normal distribution with constant variance, and it was shown that aside from some outliers, the assumption was plausible. Feature importance analysis was conducted on the rest of the machine learning models. Then, feature importance was assessed, and five features with the highest contribution to the top-performing models were selected. The selected models were retrained only on the selected features. However, contrary to expectations, these models' performances declined. Further reinforced by the decreasing R^2 adjusted value, feature importance did not improve the performance of these models. Therefore, the top-performing models before feature importance were combined and tested, yielding a decently good accuracy with an R^2 value of 77.54%. Finally, recommendations are made that can be considered for future projects to improve the accuracy of trained models.

References

1.3.3.21. Normal probability plot. (n.d.). https://www.itl.nist.gov/div898/handbook/eda/section3/normprpl.htm

3.5 - The Analysis of Variance (ANOVA) table and the F-test | STAT 462. (n.d.). https://online.stat.psu.edu/stat462/node/107/

Banerjee, S. (2022, August 20). *House Rent Prediction Dataset*. Kaggle. https://www.kaggle.com/datasets/iamsouravbanerjee/house-rent-prediction-dataset?resource=download

Devrishi. (n.d.). *New usability rating on datasets*. Kaggle. https://www.kaggle.com/discussions/product-feedback/93922

Garavaglia, S., Sharma, A., & Dun & Bradstreet. (n.d.). *A SMART GUIDE TO DUMMY VARIABLES: FOUR APPLICATIONS AND a MACRO*. https://stats.oarc.ucla.edu/wp-content/uploads/2016/02/p046.pdf

Jaadi, Z. (2023, August 4). *When and why to standardize your data*. Built In. https://builtin.com/data-science/when-and-why-standardize-your-data

Kolb, E. (2019, July 11). 75,000 people per square mile? These are the most densely populated cities in the world. *USA Today*. Retrieved April 12, 2024, from https://www.usatoday.com/picture-gallery/news/world/2019/07/11/the-50-most-densely-populated-cities-in-the-world/39664445/

Lesson 3: SLR Evaluation | STAT 462. (n.d.). https://online.stat.psu.edu/stat462/node/80/

Libretexts. (2021, January 2). 2.4: the normal distribution. Mathematics LibreTexts. https://math.libretexts.org/Bookshelves/Applied_Mathematics/Book%3A_College_Mathematics_for_Everyday_Life_(Inigo_et_al)/02%3A_Statistics_-_Part_2/2.04%3A_The_Normal_Distribution

Mishra, S. (2024, April 1). *Difference in carpet area, built-up area and super built-up area*. Housing News. https://housing.com/news/real-estate-basics-part-1-carpet-area-built-up-area-super-built-up-area/

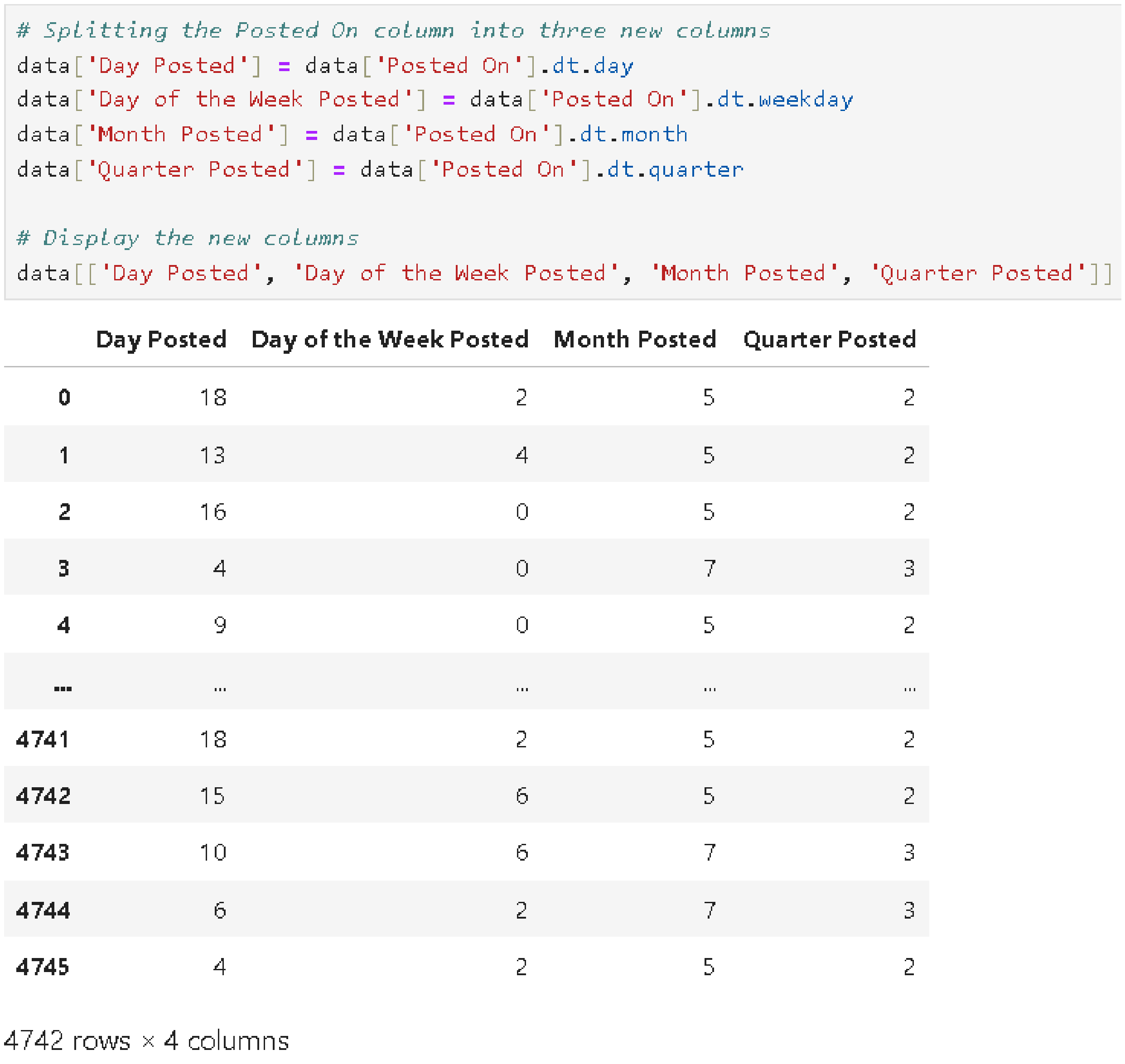
NoBroker. (2024, February 23). *Top 10 richest cities in India – Wealthiest urban centers in the country*. The NoBroker Times. https://www.nobroker.in/blog/richest-cities-in-india/

Shin, T. (2023, November 7). Understanding feature importance in machine learning. Built In. https://builtin.com/data-science/feature-importance#:~:text=Feature%20importance%20is%20a%20step,to%20predict%20a%20certain%20variable.

Team, C. (2023, November 21). *Adjusted R-squared*. Corporate Finance Institute. https://corporatefinanceinstitute.com/resources/data-science/adjusted-r-squared/#:~:text=What%20is%20the%20Adjusted%20R,a%20regression%20model%20or%20not.

Appendix A

Below is the code script to convert the *Posted On* variable into four numerical variables: *Day Posted*, *Day of the Week Posted*, *Month Posted*, and *Quarter Posted*.



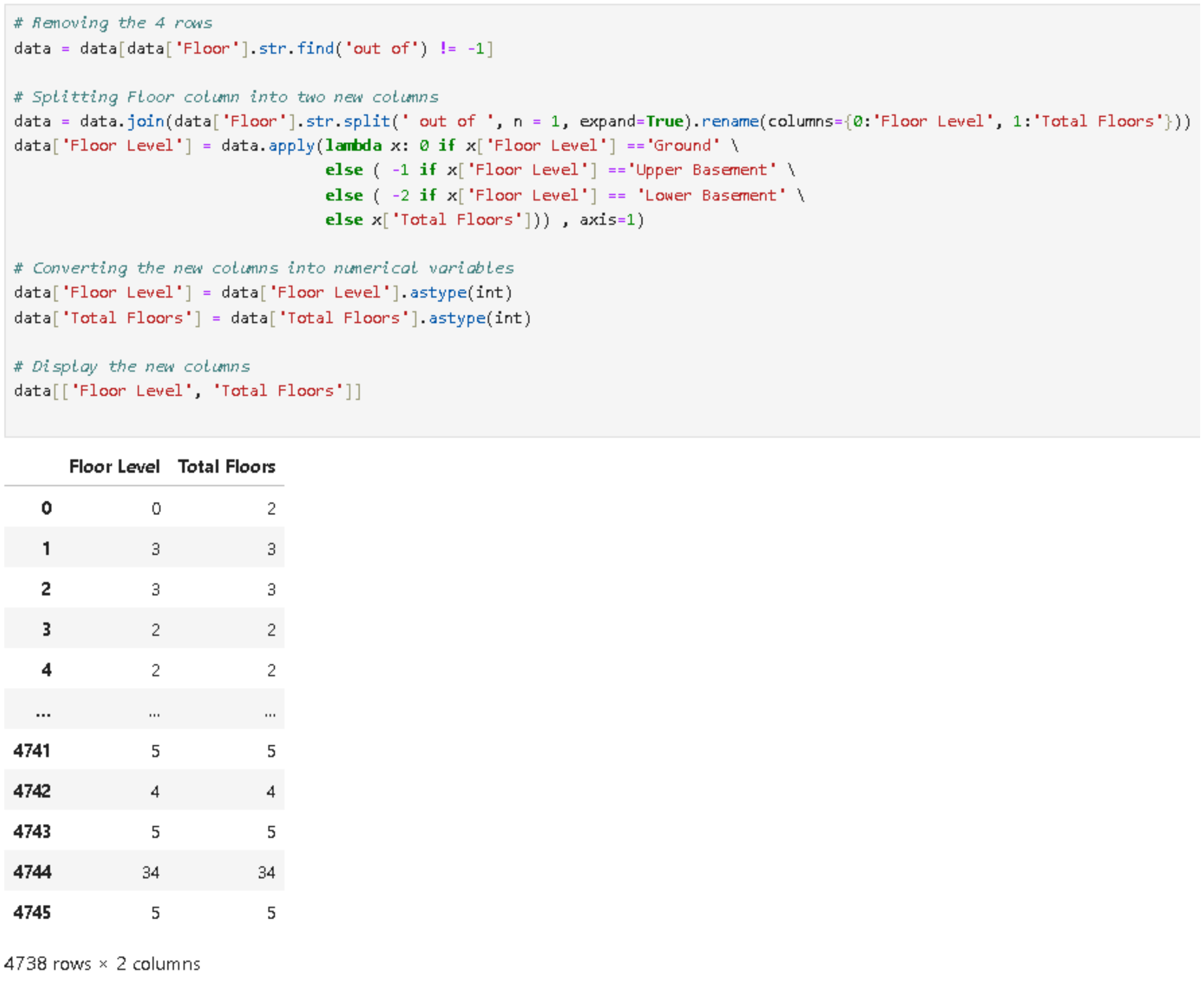
```
# Splitting the Posted On column into three new columns
data['Day Posted'] = data['Posted On'].dt.day
data['Day of the Week Posted'] = data['Posted On'].dt.weekday
data['Month Posted'] = data['Posted On'].dt.month
data['Quarter Posted'] = data['Posted On'].dt.quarter

# Display the new columns
data[['Day Posted', 'Day of the Week Posted', 'Month Posted', 'Quarter Posted']]
```

	Day Posted	Day of the Week Posted	Month Posted	Quarter Posted
0	18	2	5	2
1	13	4	5	2
2	16	0	5	2
3	4	0	7	3
4	9	0	5	2
...	...	...	...	...
4741	18	2	5	2
4742	15	6	5	2
4743	10	6	7	3
4744	6	2	7	3
4745	4	2	5	2

4742 rows × 4 columns

Appendix B

Below is the code script to convert the *Floor* variable into two numerical variables, *Floor Level* and *Total Floors*. Afterward, duplicated values were removed.



```python
# Removing the 4 rows
data = data[data['Floor'].str.find('out of') != -1]

# Splitting Floor column into two new columns
data = data.join(data['Floor'].str.split(' out of ', n = 1, expand=True).rename(columns={0:'Floor Level', 1:'Total Floors'}))
data['Floor Level'] = data.apply(lambda x: 0 if x['Floor Level'] =='Ground' \
                                 else ( -1 if x['Floor Level'] =='Upper Basement' \
                                 else ( -2 if x['Floor Level'] == 'Lower Basement' \
                                 else x['Total Floors'])) , axis=1)

# Converting the new columns into numerical variables
data['Floor Level'] = data['Floor Level'].astype(int)
data['Total Floors'] = data['Total Floors'].astype(int)

# Display the new columns
data[['Floor Level', 'Total Floors']]
```

```
      Floor Level  Total Floors
0               0             2
1               3             3
2               3             3
3               2             2
4               2             2
...           ...           ...
4741            5             5
4742            4             4
4743            5             5
4744           34            34
4745            5             5

4738 rows × 2 columns
```

```
# Dropping duplicate rows
print(data.duplicated().sum())
data = data.drop_duplicates()
```

```
16
```

```
# Reviewing information about the updated data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4722 entries, 0 to 4745
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   BHK                     4722 non-null   int64 
 1   Rent                    4722 non-null   int64 
 2   Size                    4722 non-null   int64 
 3   Area Type               4722 non-null   object
 4   City                    4722 non-null   object
 5   Furnishing Status       4722 non-null   object
 6   Tenant Preferred        4722 non-null   object
 7   Bathroom                4722 non-null   int64 
 8   Point of Contact        4722 non-null   object
 9   Day Posted              4722 non-null   int32 
 10  Day of the Week Posted  4722 non-null   int32 
 11  Month Posted            4722 non-null   int32 
 12  Quarter Posted          4722 non-null   int32 
 13  Floor Level             4722 non-null   int32 
 14  Total Floors            4722 non-null   int32 
dtypes: int32(6), int64(4), object(5)
memory usage: 479.6+ KB
```