Exploratory Data Analysis on

Housing Rent Price Prediction

by

Group 25

Mitul Dudhat
ID: 202318024
Course: MSc(DS)

Ayush Patel
ID: 202318036
Course: MSc(DS)

Course Code: IT 462
Semester : Winter 2023

Under the guidance of

Dr. Gopinath Panda

Dhirubhai Ambani Institute of Information and Communication Technology

April 28, 2024

Acknowledgment

I am writing this letter to express my heartfelt gratitude for your guidance and support throughout my project titled "Housing Rent Price Prediction." Your invaluable assistance has played a pivotal role in shaping the successful completion of this endeavor.

I am incredibly fortunate to have had the opportunity to work under your mentorship. Your expertise, encouragement, and willingness to share your knowledge have been instrumental in elevating the quality and scope of my project. Your constructive feedback and insightful suggestions have helped me overcome challenges and develop a deeper understanding of the subject matter.

Furthermore, I would like to thank the entire team at Dhirubhai Ambani Institute of Information and Communication Technology for fostering an environment of collaboration and innovation. The resources and facilities provided have been crucial in conducting comprehensive research and analysis.

I would also like to express my gratitude to my peers and colleagues who have been supportive throughout this journey. Their valuable input and camaraderie have been a constant source of motivation.

Completing this project has been a tremendous learning experience. I am confident that the knowledge and skills acquired during this endeavor will be a solid foundation for my future endeavors.

Sincerely,

Mitul Dudhat, 202318024
Ayush Patel, 202318036

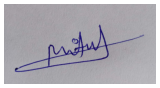
DECLARATION

We, [202318024, 2032318036] now declare that the EDA project work presented in this report is our original work and has not been submitted for any other academic degree. All the sources cited in this report have been appropriately referenced.

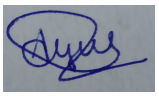
We acknowledge that the data used in this project is obtained from the makaan.com site. We also declare that we have adhered to the terms and conditions mentioned on the website for using the dataset. We confirm that the dataset used in this project is true and accurate to the best of our knowledge.

We acknowledge that we have received no external help or assistance in conducting this project except for the guidance provided by our mentor, Prof. Gopinath Panda. We declare no conflict of interest in conducting this EDA project.

We have now signed the declaration statement and confirmed the submission of this report on 29 April 2024.



Mitul Dudhat
ID: 202318024
Course: MSc(DS)



Ayush Patel
ID: 202318036
Course: MSc(DS)

CERTIFICATE

This is to certify that Group 25 comprising Mitul Dudhat and Ayush Patel has completed an exploratory data analysis (EDA) project on the Housing Rent Price Prediction, which was obtained from makaan.com

The EDA project presented by Group 25 is their original work. It was completed under the guidance of the course instructor, Prof. Gopinath Panda, who provided support and guidance throughout the project. The project is based on a thorough analysis of the Housing Rent Price Prediction dataset, and the results presented in the report are based on the data obtained from the dataset.

This certificate is issued to recognize the successful completion of the EDA project on the Housing Rent Price Prediction, which demonstrates the analytical skills and knowledge of the students of Group 25 in the field of data analysis.

Signed,
Dr. Gopinath Panda,
IT 462 Course Instructor
Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar, Gujarat, INDIA.

April 28, 2024

Contents

Abstract

This project aims to analyze the rent prices of houses in nine metropolitan cities. The data was collected through web scraping from makaan.com and consists of information about rental properties, including 13 features. The collected data was organized into separate datasets for each city and merged into a single dataframe with a shape of (15300, 13). The dataset was then cleaned thoroughly to handle missing values. Uni-variate and multivariate analysis techniques were employed to gain insights into the data. Feature engineering techniques, including extraction and selection, were then applied to enhance the dataset's predictive power. Finally, predictive models such as linear regression and random forest were utilized to forecast house rent prices, providing valuable insights into the rental housing market in metropolitan cities.

Chapter 1. Introduction

1.1 Project Idea

The project's objective is to analyze the prices of rental housing in metropolitan cities of India. We obtained the necessary data by scraping the makaan.com website, which contains various features related to rental properties. As demand for rental accommodation continues to rise in urban areas, it is crucial to understand the factors that influence rental prices. By conducting a comprehensive analysis of the dataset, we can gain insights into the dynamics of the rental market and make informed decisions regarding housing.

1.2 Data Collection

The data for this project was collected from makaan.com, a popular real estate website in India, using web scraping techniques. makaan.com provides a wide range of rental property listings from various metropolitan cities in India, making it an appropriate source for gathering data on housing rent prices.

For this project, we focused on nine metropolitan cities in India, namely Ahmedabad, Bangalore, Chennai, Delhi, Hyderabad, Kolkata, Mumbai, Pune, and Surat. These cities were selected based on their status as major urban centers with high demand for rental accommodations.

We used Python libraries such as Beautiful-Soup to automate accessing and retrieving data from the website's listings to collect the necessary data. Through web scraping, we captured essential details such as rent prices, property features, location coordinates, and other relevant information about rental properties in each of the selected metropolitan cities.

We collected a wide range of features essential for analyzing housing rent prices, including house_type (e.g., apartment, Villa), house_size (in square feet), latitude, longitude, city, location, price, currency, no of bathrooms, verification date, description, Security deposit, and Status The data used for this project was gathered by employing web scraping techniques on makaan.com, a renowned real estate website in India. The website provides many rental property listings from different metropolitan cities, making it an ideal source for gathering data on housing rent prices.

1.3 Dataset Description

The dataset contains information about nine metropolitan cities, namely, Ahmedabad, Bangalore, Chennai, Delhi, Hyderabad, Kolkata, Mumbai, Pune, and Surat. It consists of 15300 records and 13 features, including information about rental properties. The rows contain merged data from all nine cities.

The following are the fields/features included in the dataset:

- House Type: Indicates whether the property is an apartment, villa, or other property type.
- House Size: Represents the size of the property in square feet.
- Location: Specifies the geographical location of the rental property within the city.
- City: Identifies the name of the city where the property is situated.
- Latitude & Longitude: Provide the precise coordinates of the rental property location.
- Price: Denotes the rental price of the property.
- Currency: Specifies the currency used to denote the rental price.
- Number of Bathrooms: Indicates the total number of bathrooms available in the rental property.
- Verification Date: Represents the date when the rental property listing was last verified or updated.
- Description: Provides descriptive information about the rental property, including its amenities, features, and unique selling points.
- Security Deposit: Specifies the amount of the security deposit required for renting the property.
- Status: Indicates the current status of the rental property listing, such as furnished, semi-furnished, or unfurnished.

1.4 Packages Required

In programming, libraries are collections of pre-written code that can be used to perform specific tasks or functions. They are designed to save time and effort by providing readymade solutions for common problems.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno          #for missing value plot
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

LIBRARIES IMPORTED:

1. Pandas: Pandas is an open-source library for data manipulation and analysis in Python. It provides data structures like DataFrame, Series, and Panel for working with structured data in a fast, efficient, and easy way.

2. NumPy: NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays.

3. Seaborn: Seaborn is a Python visualization library built on Matplotlib, offering a high-level interface for creating attractive statistical graphics. It integrates seamlessly with Pandas and provides various plot types and color palettes for efficient data visualization and exploration.

4. Matplotlib: Matplotlib is a plotting library for Python that provides a variety of 2D and 3D plots for visualization. It is widely used for creating static, animated, and interactive visualizations in Python.

5. Missingno: Missingno is a Python library for visualizing missing data in a dataset. It provides a variety of tools for identifying and visualizing missing data patterns in a dataset.

6. LinearRegression: LinearRegression is a class in the scikit learn library that provides linear regression models for machine learning applications.

7. train_test_split: train_test_split is a function in the scikit learn library that splits a dataset into training and testing subsets.

8. mean_squared_error: mean_squared_error is a function in the scikitlearn library that calculates the mean squared error (MSE) between the actual and predicted values of a regression problem. It is a popular metric for evaluating the performance of regression models. A lower MSE value indicates a better model performance.

1.5 Data Understanding

1.5.1 Importing Dataset From Drive

The code imports the dataset named merged_df.csv from Google Drive which is a consolidated dataset of all nine metropolitan cities. The "read_csv ()" function is used to read the CSV file into a DataFrame object.

```
merged_df.head(5)
```

	house_type	house_size	location	city	latitude	longitude	price	currency	numBathrooms	verificationDate	description	SecurityDeposit	Status
0	3 BHK Apartment	1,517 sq ft	Sanathal	Ahmedabad	22.974880	72.456947	50000	INR	2.0	Posted a year ago	It's a 3 bhk multistorey apartment situated in...	No Deposit	Semi-Furnished
1	1 BHK Apartment	540 sq ft	Maninagar	Ahmedabad	22.990902	72.607094	10000	INR	1.0	Posted 3 years ago	It's a 1 bhk multistorey apartment situated in...	No Deposit	Furnished
2	3 BHK Apartment	1,860 sq ft	Shela	Ahmedabad	22.995766	72.467430	45000	INR	3.0	Posted a month ago	Well designed 3 bhk multistorey apartment is a...	No Deposit	Furnished
3	1 BHK Apartment	775 sq ft	Narolgam	Ahmedabad	19.216536	73.087166	16000	INR	2.0	Posted a month ago	This spacious 1 bhk multistorey apartment is a...	No Deposit	Semi-Furnished
4	1 BHK Independent Floor	680 sq ft	Isanpur	Ahmedabad	22.978258	72.600227	8500	INR	1.0	Posted 15 days ago	A spacious 1 bhk builder floor is available fo...	No Deposit	Semi-Furnished

1.5.2 Data Information

```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15300 entries, 0 to 15299
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   house_type        15297 non-null  object
 1   house_size        15297 non-null  object
 2   location          15297 non-null  object
 3   city              15297 non-null  object
 4   latitude          15300 non-null  float64
 5   longitude         15300 non-null  float64
 6   price             15300 non-null  int64
 7   currency          15300 non-null  object
 8   numBathrooms      15278 non-null  float64
 9   verificationDate  15297 non-null  object
 10  description       15186 non-null  object
 11  SecurityDeposit   15297 non-null  object
 12  Status            15297 non-null  object
dtypes: float64(3), int64(1), object(9)
memory usage: 1.5+ MB
```

The merged_df.info() method provides information about the DataFrame(rows=15300, columns=13), including the number of non-null values, the data type of each column, and the memory usage of the DataFrame.

1.5.3 Reading columns in the datasets

```
merged_df.columns
```

```
Index(['house_type', 'house_size', 'location', 'city', 'latitude', 'longitude',
       'price', 'currency', 'numBathrooms', 'verificationDate', 'description',
       'SecurityDeposit', 'Status'],
      dtype='object')
```

When we call merged_df.columns, it returns a Pandas Index object containing the names of all the columns in the merged_df DataFrame. This tells us the names of all the columns in the DataFrame, and we can use these names to access specific columns or perform operations on the entire DataFrame.

1.5.4 Dataset Shape

```
#Shape of the dataset
merged_df.shape
```

```
(15300, 13)
```

We can note the shape of the dataset as 15300 rows and 13 columns. Knowing the shape of the dataset is important because it gives an idea about the size of the dataset and the amount of data it contains.

1.5.5 Descriptive Analysis

```
# Selecting numeric columns
numeric_columns = ['latitude', 'longitude', 'price', 'numBathrooms']

# Computing summary statistics
merged_df[numeric_columns].describe()
```

	latitude	longitude	price	numBathrooms
count	15300.000000	15300.000000	1.530000e+04	15278.000000
mean	21.251965	77.281211	6.986165e+04	2.247415
std	5.336932	4.872246	1.190595e+05	0.979223
min	11.351545	72.320129	2.000000e+03	1.000000
25%	18.970388	72.941322	2.000000e+04	2.000000
50%	19.258764	77.165222	3.500000e+04	2.000000
75%	28.501354	77.694761	6.500000e+04	3.000000
max	30.888254	91.792137	2.162955e+06	11.000000

It applies the describe() method to only the selected columns(i.e numerical columns) to generate summary statistics such as count, mean, standard deviation, minimum, and maximum values for each column.

Chapter 2. Data Cleaning

2.1 Part A: Missing Values (Interpreting Numerically)

2.1.1 Checking for Missing Values

```
#Checking null values (NUMERICAL)
merged_df.isna().sum()
```

```
house_type            3
house_size            3
location              3
city                  3
latitude              0
longitude             0
price                 0
currency              0
numBathrooms         22
verificationDate      3
description         114
SecurityDeposit       3
Status                3
dtype: int64
```

This is used to count the number of missing values (NaN values) in each column of ”merged_df”.The method ”.isna()” creates a Boolean mask indicating where there are missing values in the DataFrame, returning ”True” for missing values and ”False” otherwise. The method ”.sum()” is then applied which counts the number of ”True” values (i.e. the number of missing values) for each column of the DataFrame. Checking for missing values is important because it can affect the accuracy and reliability of any data analysis.

2.1.2 Checking Missing Percentage

```
merged_df.isnull().mean()*100
merged_mis_pcent=100*merged_df.isnull().sum()/len(merged_df)
print('\n\nMissing percentage:\n\n', merged_mis_pcent)
```

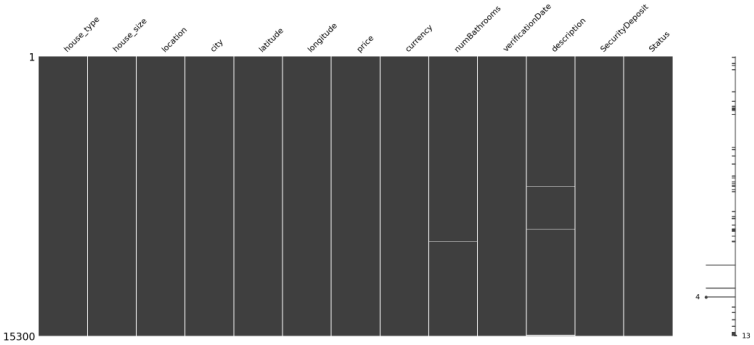
```
Missing percentage:

 house_type           0.019608
house_size           0.019608
location             0.019608
city                 0.019608
latitude             0.000000
longitude            0.000000
price                0.000000
currency             0.000000
numBathrooms         0.143791
verificationDate     0.019608
description          0.745098
SecurityDeposit      0.019608
Status               0.019608
dtype: float64
```

The code is used to compute the percentage of missing values in a Pandas DataFrame called merged_df. It computes the total percentage of missing values across all columns in the DataFrame and returns a Pandas Series object that shows the total percentage of missing values for each column.
The percentage of missing values can give you an idea of the quality of the data in each column. If a column has a very high percentage of missing values, it may not be very useful for analysis or modeling and may need to be removed or imputed with appropriate values.

2.2 Part B: Visualizing the Missing Values (Interpreting Graphically)

Visualizing the locations of the missing data:
msno. matrix(merged_df) is a function call that creates a matrix plot visualization of missing values in the DataFrame merged_df.In a given dataset, the plot will appear white in areas where there are missing values. On the right-hand side, there is a sparkline that provides an overview of the data's completeness and indicates the row with the least number of null values as well as the total number of columns. This information can be found at the bottom of the sparkline.

- msno.matrix(merged_df)



2.3 Part C: Dealing With Missing Values

```
# we show the row which house_type column has null value
rows_with_missing_house_type = merged_df[merged_df['house_type'].isnull()]
rows_with_missing_house_type
```

	house_type	house_size	location	city	latitude	longitude	price	currency	numBathrooms	verificationDate	description	SecurityDeposit	Status
11424	NaN	NaN	NaN	NaN	19.132290	72.860413	85000	INR	NaN	NaN	NaN	NaN	NaN
12681	NaN	NaN	NaN	NaN	19.155031	72.855659	78000	INR	NaN	NaN	NaN	NaN	NaN
13168	NaN	NaN	NaN	NaN	19.123636	72.830460	50000	INR	NaN	NaN	NaN	NaN	NaN

2.3.1 Drop/Delete Missing Values

```
#DELETING THIS ROWS BECAUSE OF MOST OF FEATURE HAS NULL VALLUE
merged_df.dropna(subset=['house_type'], inplace=True)
```

```
# Identify features with missing values
vars_with_na = [var for var in merged_df.columns if merged_df[var].isnull().sum() > 0]
print("After deleteing row, features with missing values : " , vars_with_na)
```

```
After deleteing row, features with missing values :  ['numBathrooms', 'description']
```

2.4 Part D: Reasons And Imputing Null Values

2.4.1 Types Of Missing Values :

1. **Missing Completely At Random (MCAR)**: In this mechanism, the missingness is completely random and unrelated to any variables in the dataset, whether observed or unobserved.
2. **Missing At Random (MAR):** In this mechanism, the missingness depends on other observed variables in the dataset, but not on the missing variable itself.
3. **Missing Not At Random (MNAR):** In this mechanism, the missingness depends on the value of the missing variable itself, or on other variables that are not observed.

Imputing Missing Values For numBathrooms

- Here missing data is MAR. It is a continuous numerical variable, so median imputation would be the most appropriate technique. This is because the median is robust to outliers and provides a reliable estimate of central tendency, making it suitable for numerical variables with skewed distributions or the presence of outliers.

Reason For Missing 74% Description Data

- Optional Information
- Data Entry
- Data Availability
- Quality of Listings
- Purpose of Description

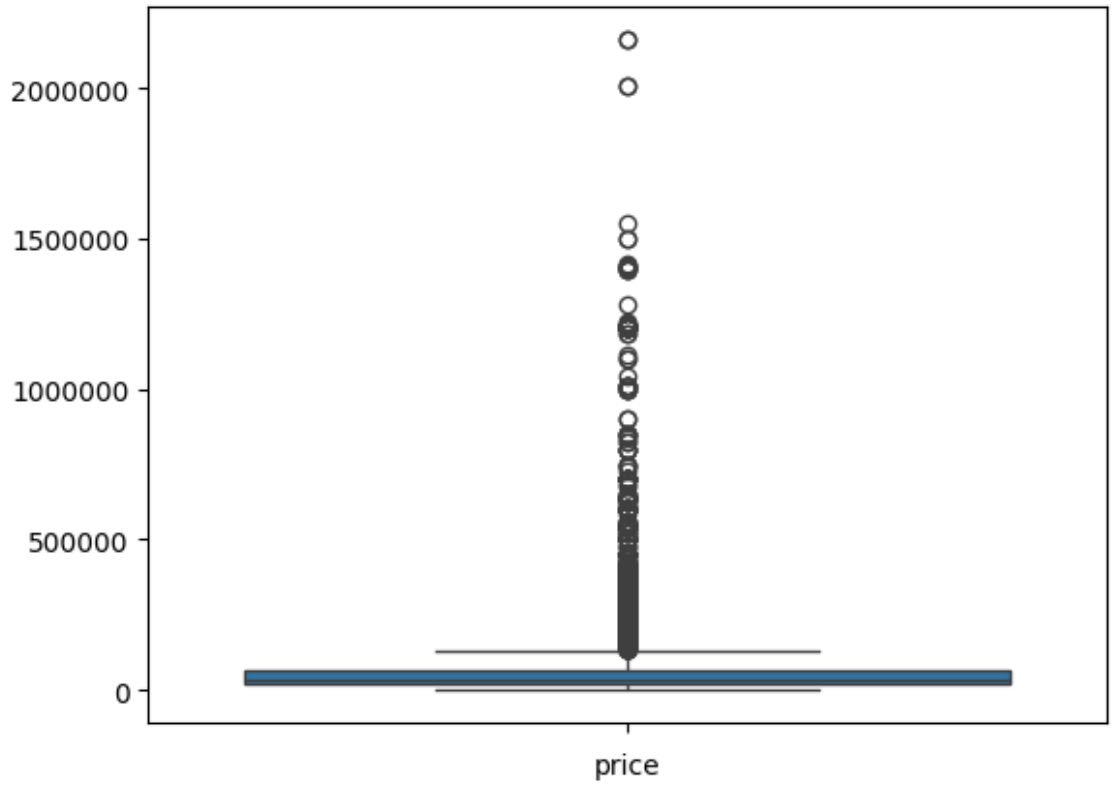
It is a type of MCAR. From the above reasons, we can fill the NULL value with ”unknown”

```
# Impute missing values with the median
merged_df['numBathrooms'].fillna(merged_df['numBathrooms'].median(), inplace=True)
```

2.5 Identifying & Removing The Outlier

2.5.1 Checking Any Outlier

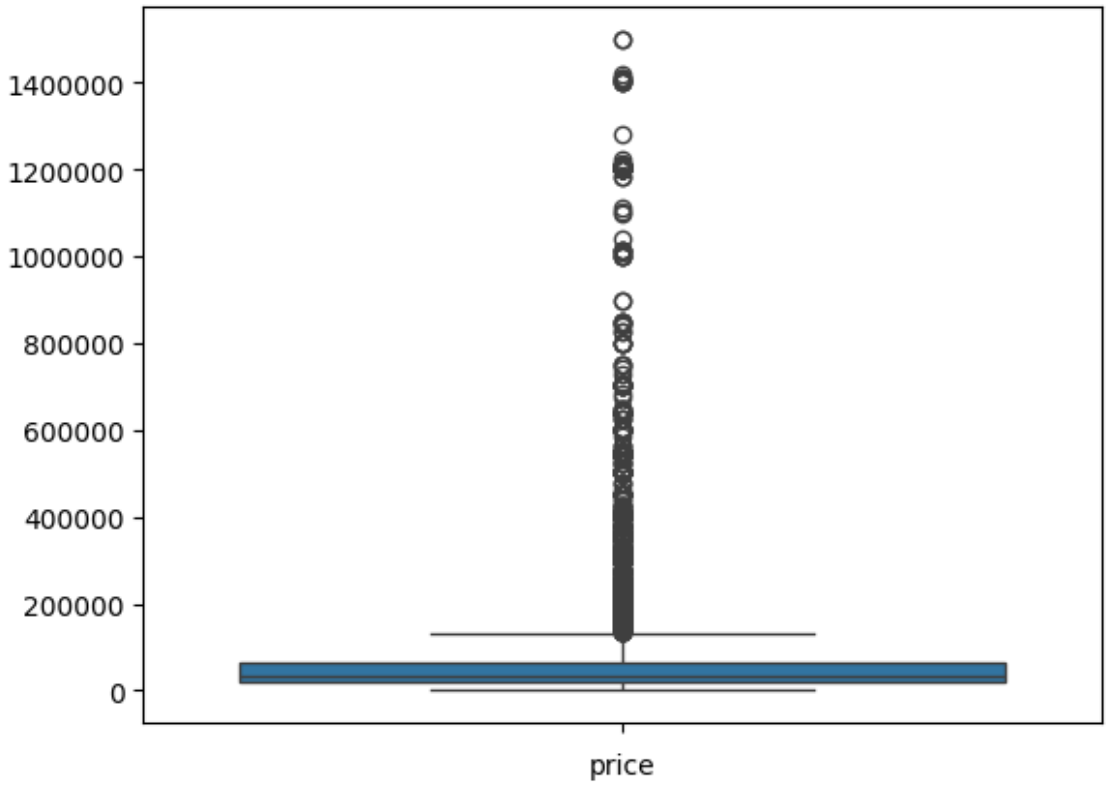
```
plt.ticklabel_format(style='plain')
sns.boxplot(x=None, y=merged_df['price'], data=merged_df)
plt.xlabel("price")
plt.ylabel("")
plt.show()
```



- We will eliminate the outliers in the data that are higher than 150000, as we can see them.

2.5.2 Removing The Outliers

```
merged_df = merged_df[merged_df['price'] <= 1500000]

plt.ticklabel_format(style='plain')
sns.boxplot(x=None, y=merged_df['price'], data=merged_df)
plt.xlabel("price")
plt.ylabel("")
plt.show()
```



Chapter 3. Visualization

3.1 Univariate Analysis

3.1.1 Numerical Data

```
# Analyze each numerical feature individually
numerical_features = ['house_size', 'latitude', 'longitude', 'price', 'numBathrooms']
```

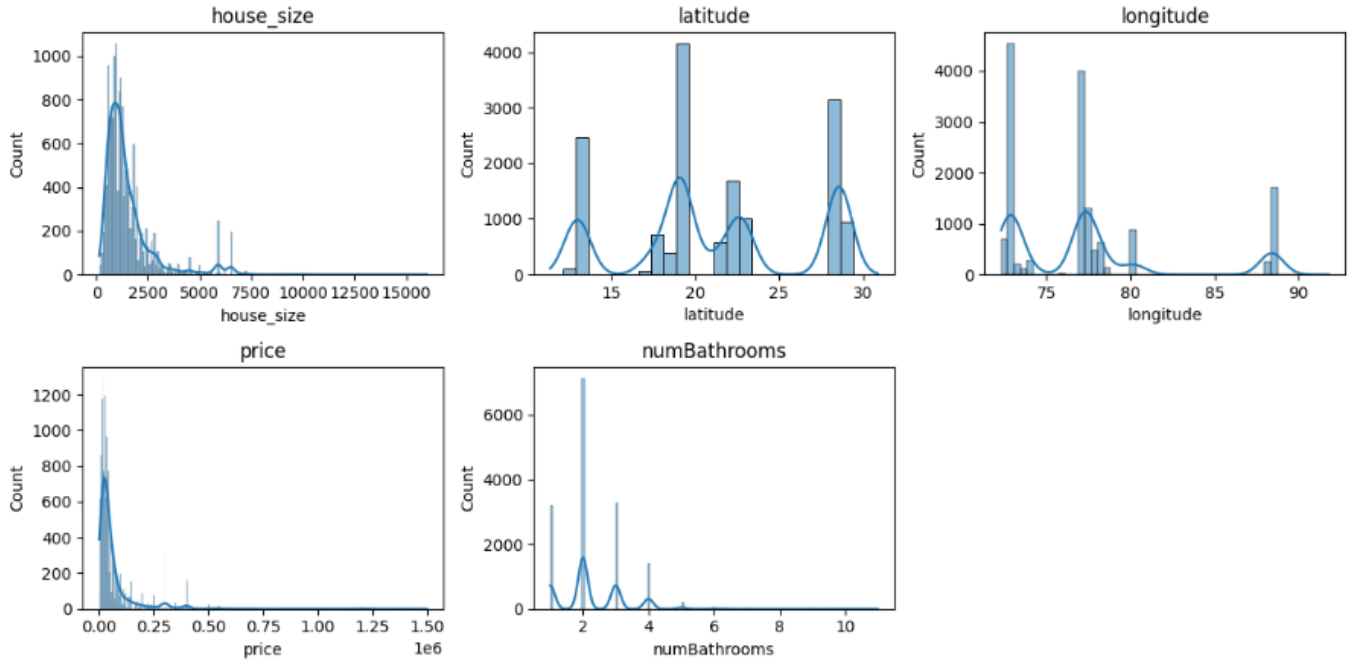
```
# Calculate summary statistics
merged_df[numerical_features].describe()
```

	house_size	latitude	longitude	price	numBathrooms
count	15292.000000	15292.000000	15292.000000	1.529200e+04	15292.000000
mean	1508.943565	21.249977	77.282107	6.923720e+04	2.246534
std	1244.701812	5.336592	4.873124	1.139024e+05	0.978300
min	150.000000	11.351545	72.320129	2.000000e+03	1.000000
25%	800.000000	18.970377	72.941322	2.000000e+04	2.000000
50%	1163.500000	19.258760	77.165180	3.500000e+04	2.000000
75%	1763.250000	28.501100	77.695559	6.500000e+04	3.000000
max	16000.000000	30.888254	91.792137	1.500000e+06	11.000000

3.1.2 Graphical Data

Histograms

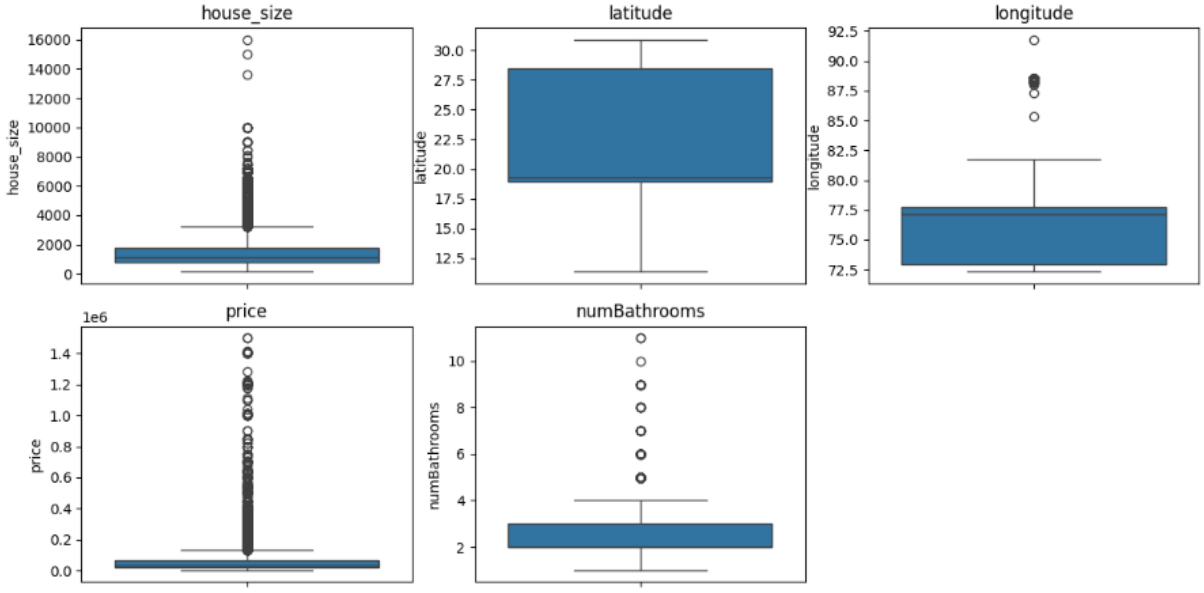
```
# Histograms
plt.figure(figsize=(12, 6))
for i, feature in enumerate(numerical_features, 1):
    plt.subplot(2, 3, i)
    sns.histplot(merged_df[feature], kde=True)
    plt.title(feature)
plt.tight_layout()
plt.show()
```



- Histograms are visual representations of data distribution. They divide data into bins and count the number of data points in each bin to show the frequency or probability distribution of values in a dataset.
- Adding a KDE to the histogram provides a smoothed estimate of the probability density function of the data.
- Observation: We notice that our numerical values do not follow a normal distribution. Therefore, we aim to transform our numerical values into a normal distribution.

Boxplots

```
# Box plots
plt.figure(figsize=(12, 6))
for i, feature in enumerate(numerical_features, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=merged_df[feature])
    plt.title(feature)
plt.tight_layout()
plt.show()
```

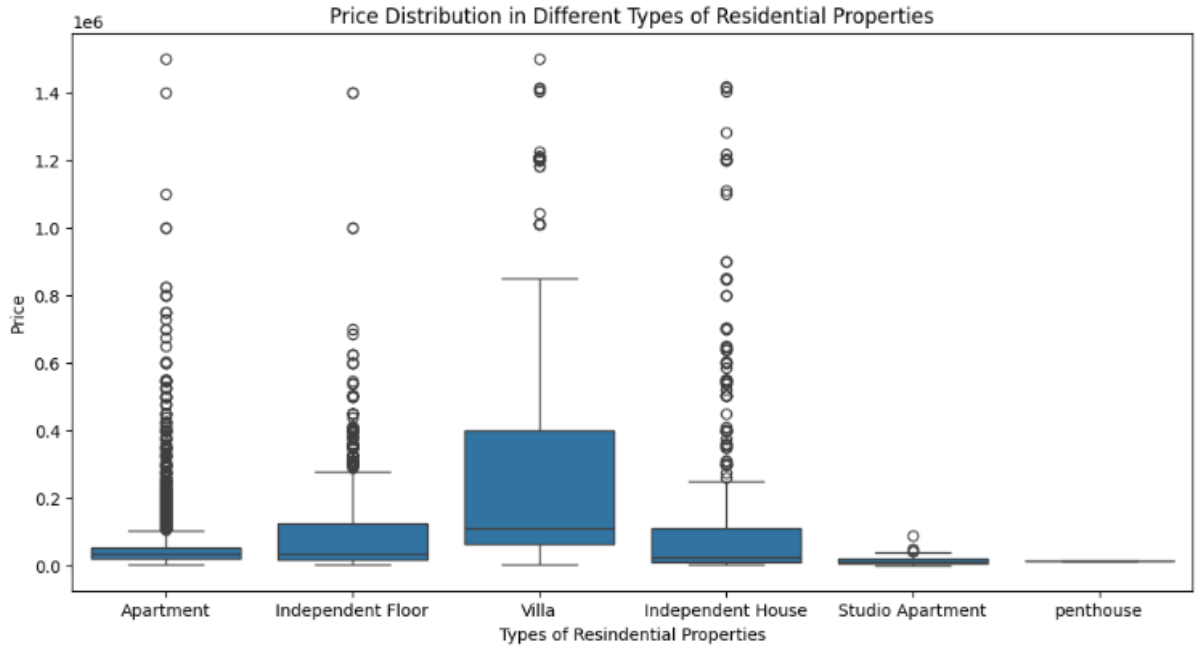


Observation:

- A box plot is a graph showing the distribution of data. The box represents the middle 50 percent, with the line inside showing the median. The whiskers show the range of minimum and maximum values, usually 1.5 times the IQR. Outliers may be plotted beyond the whiskers.
- House Size: The boxplot analysis indicates a small range of variance in house sizes, indicating that properties in the dataset are mostly of similar size. This could point to a particular housing trend or market sector.
- Latitude: The majority of rental properties within the examined metropolitan regions appear to be located at higher latitudes, as indicated by the positively skewed distribution of latitudes.
- Longitude: The distribution of longitudes is negatively skewed, suggesting that properties located at higher latitudes are preferred over those that are dispersed over a large geographic range.
- Number of Bathrooms: The dataset's similarity is further shown by the short-range in the number of bathrooms, which shows that rental houses often provide a standard number of bathrooms. Understanding housing preferences and market demands for rental houses in urban areas may benefit from this perspective.

Box plot for Price Distribution in Different Types of Residential Properties

```
# Box plot for Price Distribution in Different Types of Residential Properties
plt.figure(figsize=(12, 6))
sns.boxplot(data=merged_df, x='house_type_name', y='price')
plt.title("Price Distribution in Different Types of Residential Properties")
plt.xlabel("Types of Resindential Properties")
plt.ylabel("Price")
plt.show()
```



Observations :

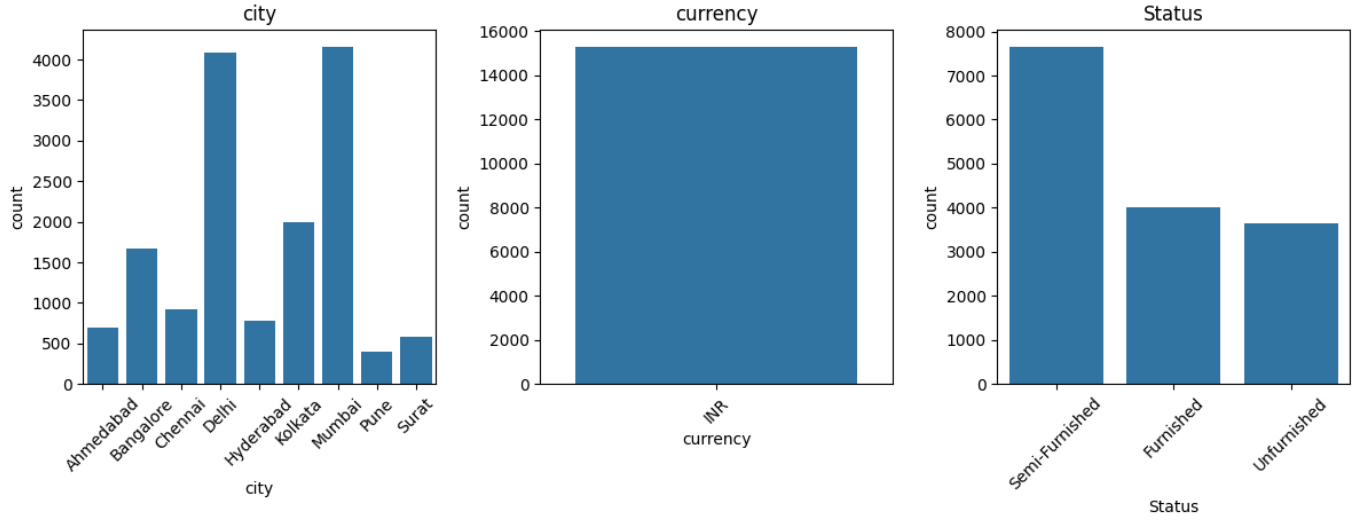
- The data map depicts several types of residential properties, including apartments, independent floors, villas, independent houses, studio apartments, and penthouses. Observations show a negative skew across all categories of attributes. However, villas have a significantly wider interquartile range (IQR) than the rest.
- Apartments, separate floors, and studio apartments have particularly low IQRs. Furthermore, studio flats and penthouses have significantly narrow ranges, distinguishing them from other property categories.

3.1.3 Categorical Data

Count Plot

```
# Analyze each categorical feature individually
categorical_features = ['city', 'currency', 'Status']
```

```
# Count plots
plt.figure(figsize=(12, 8))
for i, feature in enumerate(categorical_features, 1):
    plt.subplot(2, 3, i)
    sns.countplot(data=merged_df, x=feature)
    plt.title(feature)
    plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



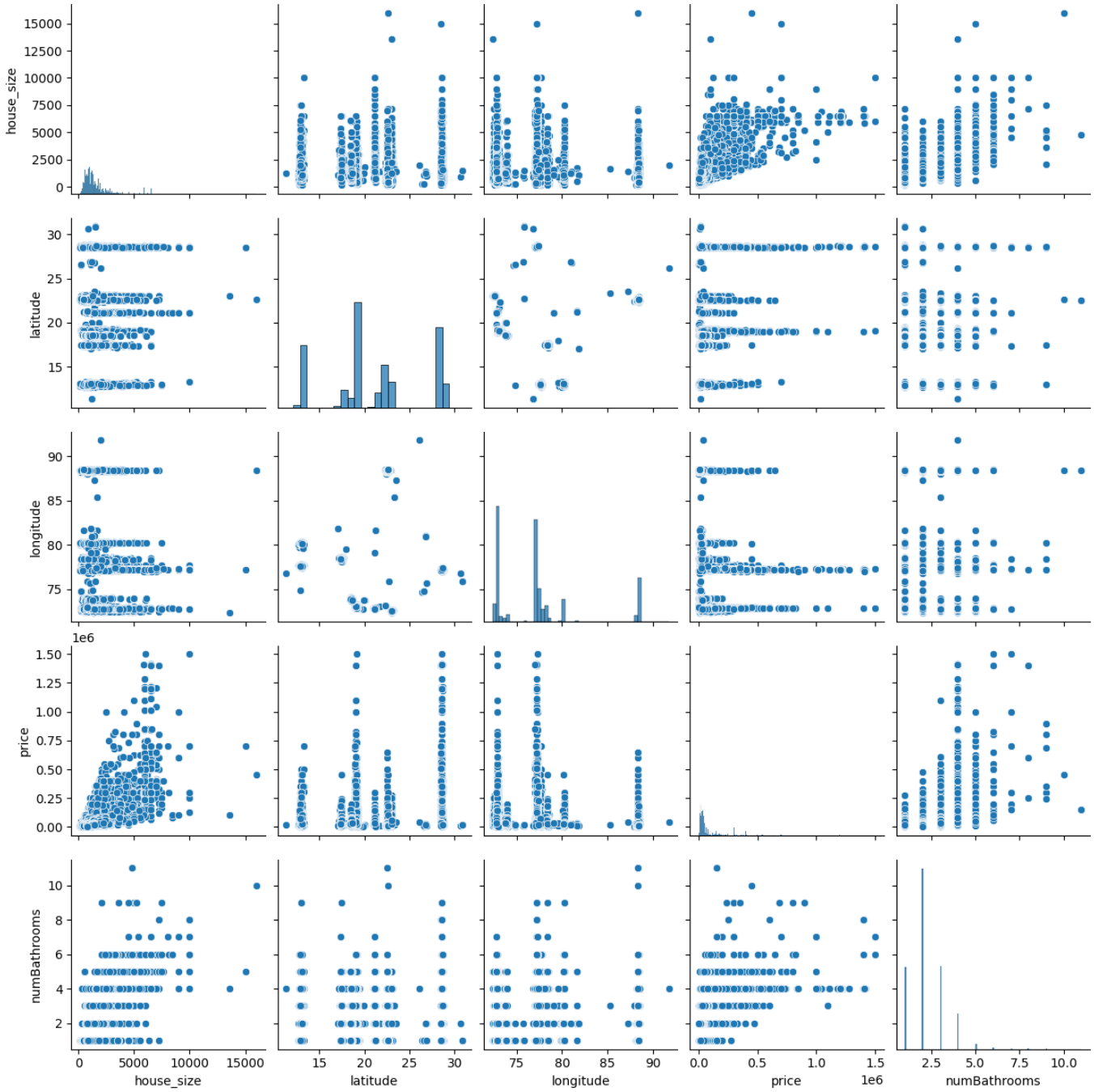
Observations :

- Based on the graph study, Delhi and Mumbai have the highest number of rental flats, followed by Bangalore. In contrast, Pune, Surat, and Ahmedabad have the lowest availability of rental units among the cities examined.
- The data shows that semi-furnished flats are in higher demand than furnished and unfurnished flats, indicating that consumers prefer to buy semi-furnished homes.

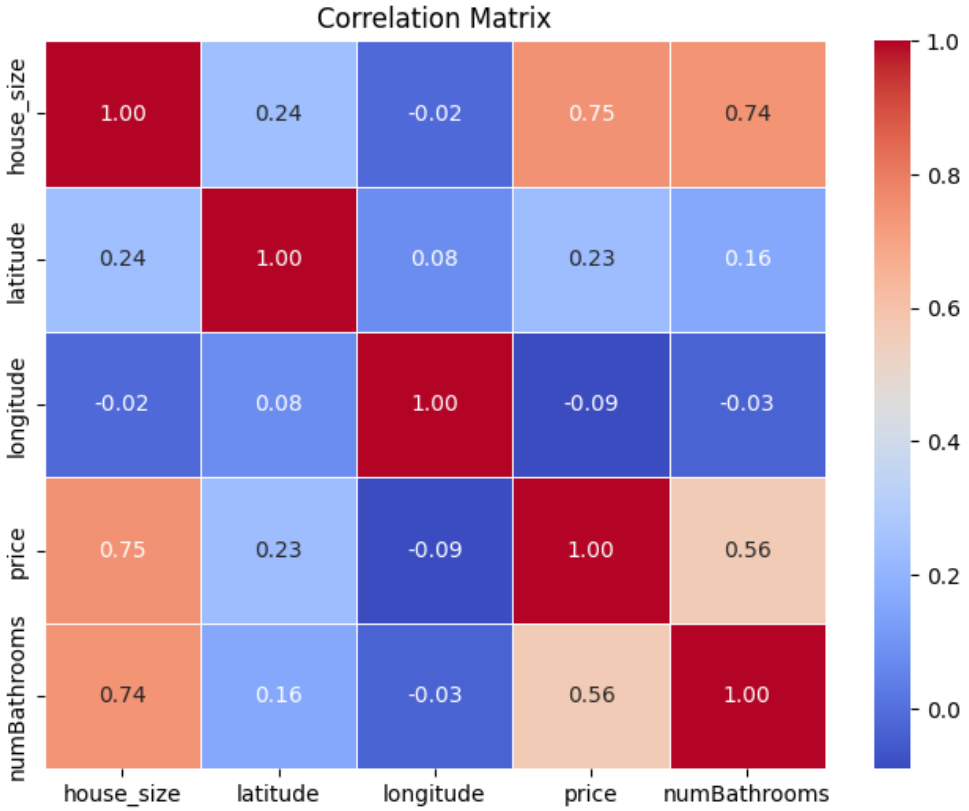
3.2 Multivariate analysis

3.2.1 Numerical Data

Pair Plot



Correlation matrix

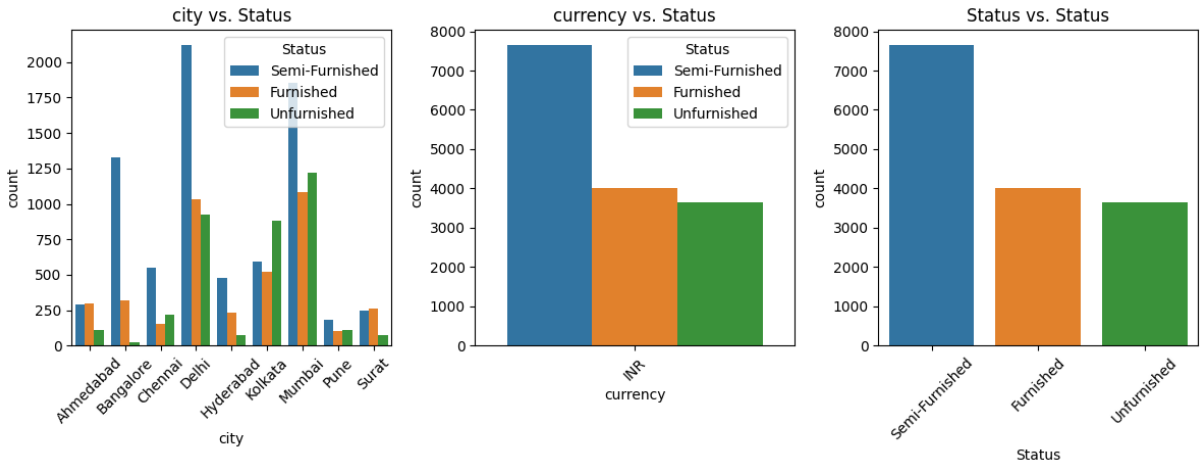


Observations :

- The correlation matrix shows that house size and price are strongly associated (0.75), while number of bathrooms and house size are highly correlated (0.74), and price and number of bathrooms have a positive correlation (0.56). As from the above results, the overall size of the house is a more crucial element in pricing

.

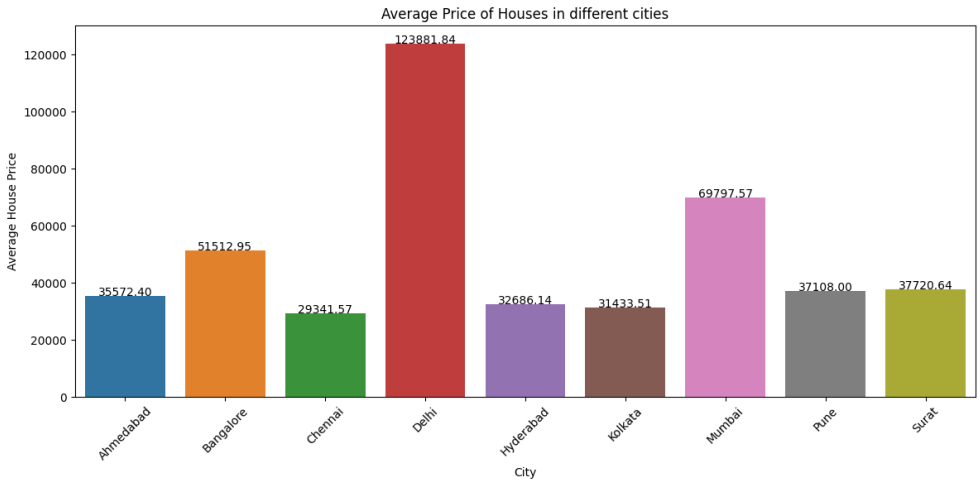
3.2.2 Categorical Data



Observations :

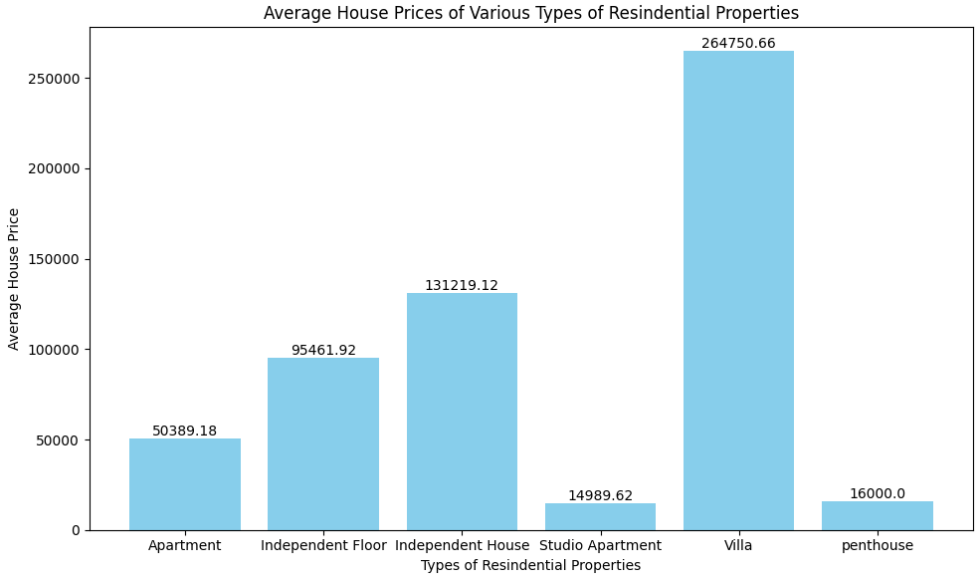
- From the graph, we can infer that Delhi has the largest demand for rental apartments and that the desire for semi-furnished apartments is higher than that for furnished apartments. Mumbai and Bangalore are the next two most popular cities in demand for apartment rentals, although Pune has the lowest demand overall.

3.3 Bar Graph



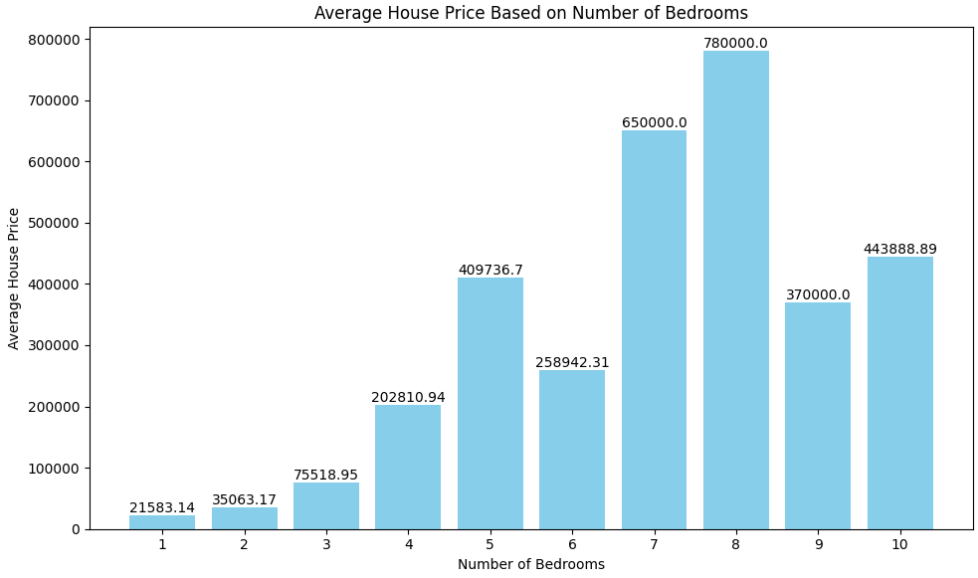
Observations :

- The graph indicates that there is a greater demand for rental homes in Delhi, which may be the cause of the highest home prices. The graph also shows that Mumbai City has the second-highest rental prices.



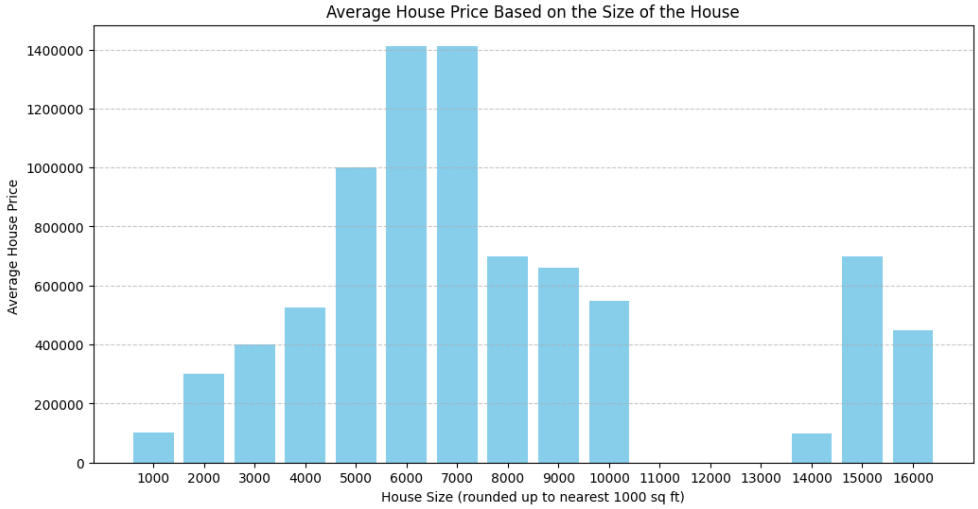
Observations :

- Villa has the highest average price of 264750 among the various types of residential properties, followed by independent houses with an average price of 131219.



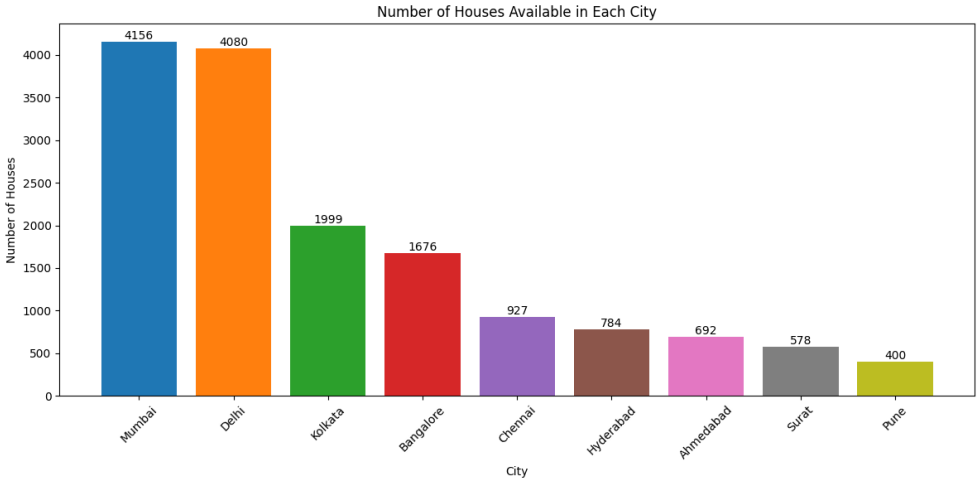
Observations :

- Based on our observations, we can conclude that the number of bedrooms significantly impacts the price. Notably, eight bedrooms have the highest average rental price of 780000.



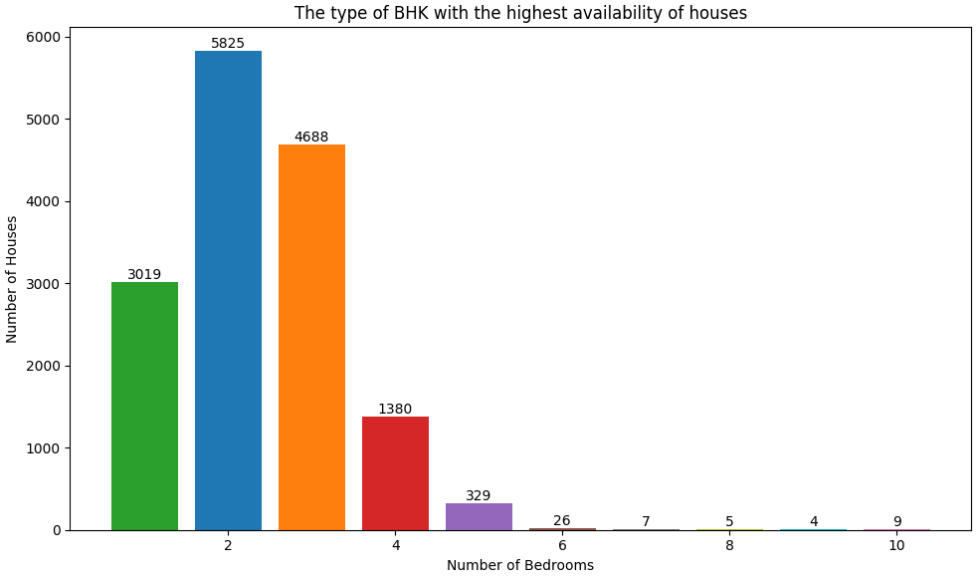
Observations :

- It is evident from the graph that a particular range of dwelling areas attracts a high level of demand in the rental market. In particular, the most expensive average rental costs are seen in homes between 6000 and 7000 square feet. Furthermore, there is still a significant demand for homes that are 5000 square feet.



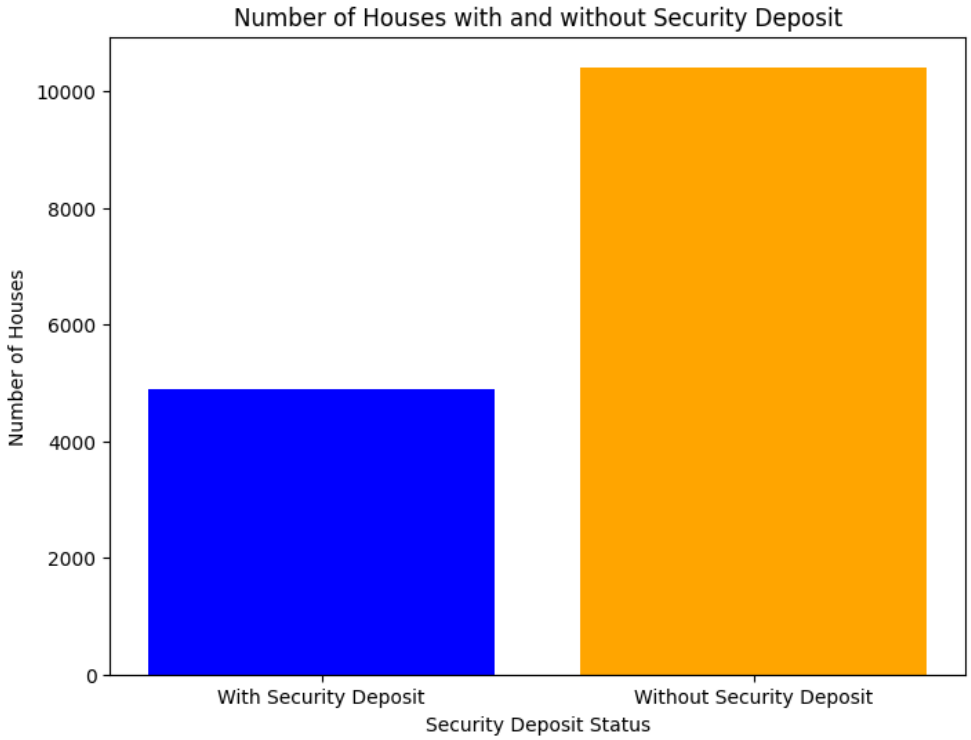
Observations :

- We can notice that there are more available homes in Delhi and Mumbai, which increases demand for rental housing overall.



Observations :

- we can have remarkable insights from the graph that there is a higher demand for 2BHK houses for rental compared to others. Followed by 3BHK.



Observations :

- It is evident that a greater number of homes are available for rental without requiring a security deposit.

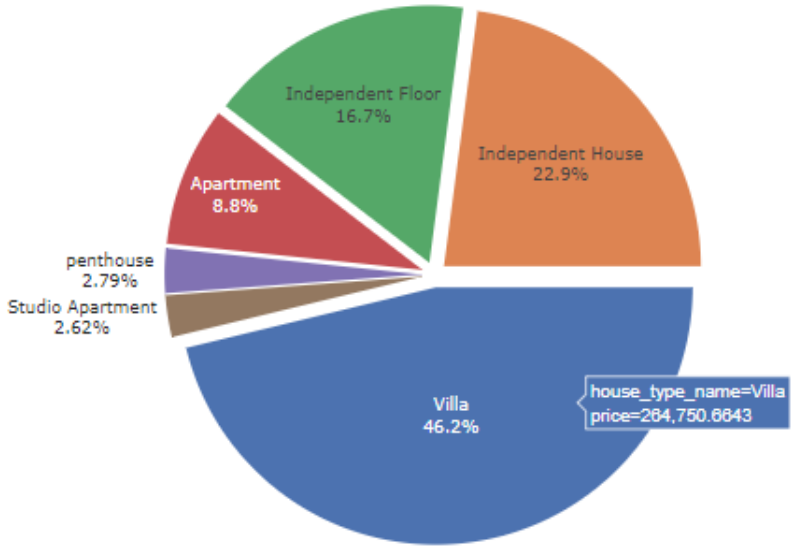
3.4 Scatter Plot

```
# Scatter plot for Price vs. House Size features
plt.figure(figsize=(10, 6))
sns.scatterplot(data=merged_df, x='price', y='house_size', hue='city')
plt.title("Price of houses varies with their size in different cities")
plt.xlabel("Price")
plt.ylabel("House Size")
plt.show()
```

Observations :

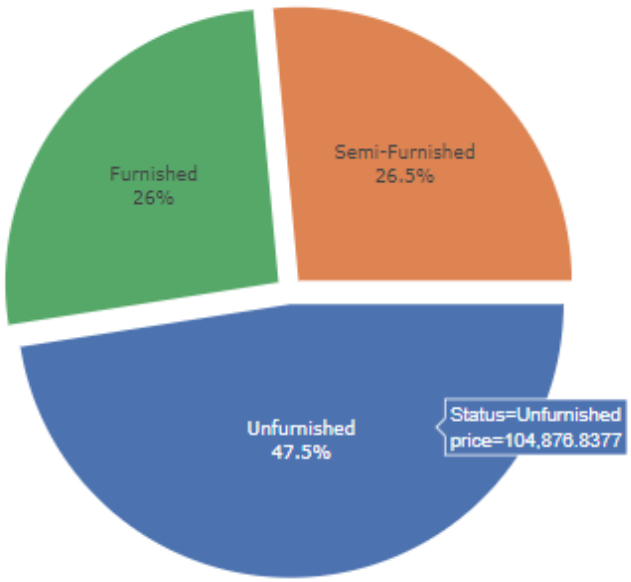
- As we can see, the majority of possible housing sizes fall between 0 and 6000 square feet. additionally, the majority of the data are concentrated in the price range of 0 to 4 lakh and the home range of 0 to 4000 square feet.
- We can therefore conclude that there is a need for homes in India's metropolitan areas, although at a restricted price range and square feet.

3.5 Pie-Chart



Observations :

- It is evident from the pie chart study that there is a considerable demand for villas in the rental market.

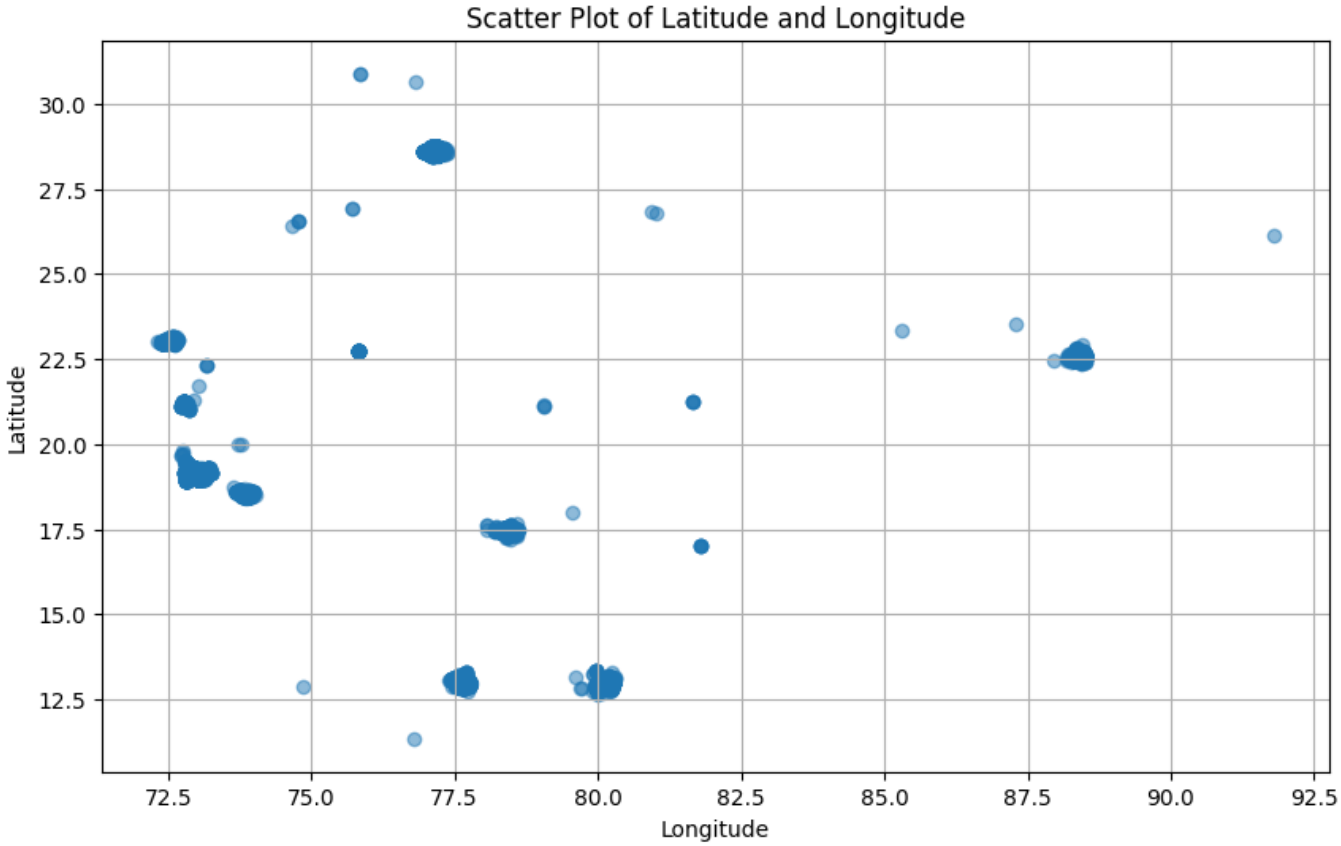


Observations :

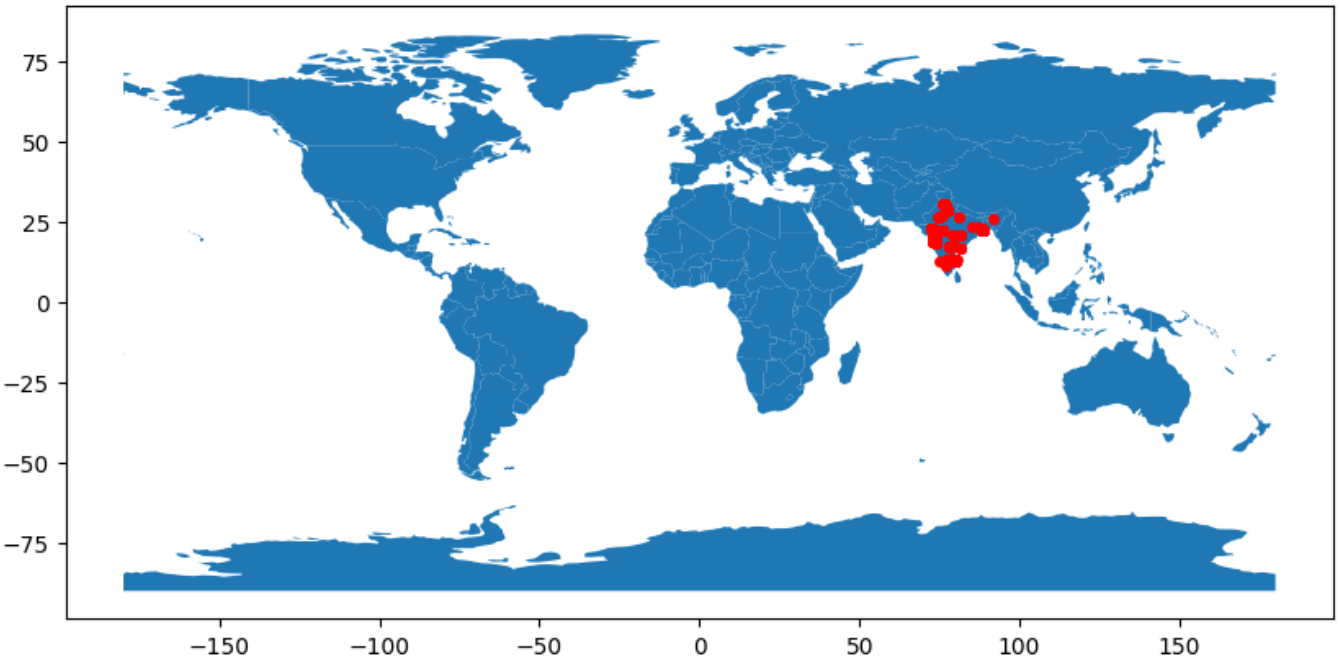
- According to the pie chart, unfurnished properties make up the largest portion of the rental housing market.

3.6 Geographical-Variables

```
import matplotlib.pyplot as plt

# Scatter plot of latitude and longitude
plt.figure(figsize=(10, 6))
plt.scatter(merged_df['longitude'], merged_df['latitude'], alpha=0.5)
plt.title('Scatter Plot of Latitude and Longitude')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid(True)
plt.show()
```



```
from shapely.geometry import Point
import geopandas as gpd
from geopandas import GeoDataFrame


geometry = [Point(xy) for xy in zip(merged_df['longitude'], merged_df['latitude'])]
gdf = GeoDataFrame(merged_df, geometry=geometry)

#this is a simple map that goes with geopandas
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
gdf.plot(ax=world.plot(figsize=(10, 6)), marker='o', color='red', markersize=15);
```



Observations :

- In the above map, the red dots show the latitude and longitude of the 9 metropolitan cities of India.

Chapter 4. Feature Engineering

4.1 Feature Extraction & Selection

```
merged_df['premium_house'] = np.where((merged_df['numBathrooms']>=merged_df['num_of_bedrooms'])&(merged_df['num_of_bedrooms']>1),1,0)
merged_df['beds_per_sqfeet'] = merged_df['num_of_bedrooms'] / merged_df['house_size']
merged_df['baths_per_beds'] = merged_df['numBathrooms'] / merged_df['num_of_bedrooms']
merged_df['has_pool'] = merged_df['description'].apply(lambda x: 1 if 'pool' in x.lower() or 'swimming' in x.lower() else 0)
merged_df['gym_nearby'] = merged_df['description'].apply(lambda x: 1 if 'gym' in x.lower() or 'fitness' in x.lower() else 0)
merged_df['school/clg_nearby'] = merged_df['description'].apply(lambda x: 1 if 'school' in x.lower() or 'college' in x.lower() else 0)
merged_df['wifi_facilities'] = merged_df['description'].apply(lambda x: 1 if 'wifi' in x.lower() or 'wi-fi' in x.lower() else 0)
merged_df['valet_service'] = merged_df['description'].apply(lambda x: 1 if 'valet' in x.lower() else 0)
merged_df['shopping_nearby'] = merged_df['description'].apply(lambda x: 1 if 'shopping' in x.lower() else 0)
merged_df['sports_playground'] = merged_df['description'].apply(lambda x: 1 if 'sport' in x.lower()  or 'sports' in x.lower()
                                                or 'tennis' in x.lower() or 'soccer' in x.lower()
                                                or 'soccers' in x.lower() or 'court' in x.lower() else 0)
merged_df['dining_nearby'] = merged_df['description'].apply(lambda x: 1 if 'dining' in x.lower() else 0)
```

New features have been developed by using existing features to improve the predictive power of the dataset and provide additional insights into the rental housing market.
Here are some of the features that have been engineered:

- Premium House Indicator: A binary feature called 'premium_house' has been created to identify properties with more bathrooms than bedrooms. This indicates higher-end accommodations.
- Beds per Square Feet: This feature calculates the ratio of the number of bedrooms to the size of the house. It offers insights into the density of bedrooms within each property.
- Baths per Beds: Another ratio, 'baths_per_beds', has been calculated to determine the average number of bathrooms per bedroom in each property.
- Amenities Indicators: A series of binary indicators have been generated based on the property descriptions to identify the presence of amenities such as a pool, gym, nearby schools or colleges, Wi-Fi facilities, valet service, shopping facilities, sports playgrounds, and dining options.

4.2 Feature Transformation

4.2.1 Numerical Variable Transformation

```
for var in ['price','house_size','baths_per_beds','beds_per_sqfeet']:
    merged_df[var] = np.log(merged_df[var])
```

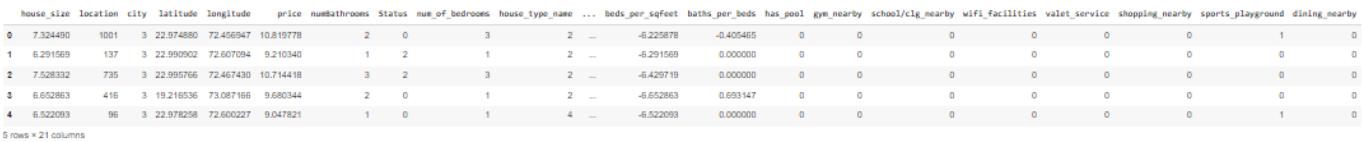
As a part of our feature engineering process, we have applied certain transformations to our numerical features to normalize them. Specifically, we have taken the logarithm of 'price', 'house_size', 'baths_per_beds', and 'beds_per_sqfeet'. This helps us achieve a more Gaussian-like distribution, which can improve the performance of linear machine-learning models in predictive modeling tasks. By doing this, we ensure that these features are more normally distributed and exhibit better performance in predictive modeling.

4.2.2 Categorical Variable Transformation

```
def replace_categories(df, feature, output):
    ordered_labels = df.groupby([feature])[output].mean().sort_values().index
    ordinal_label = {k: i for i, k in enumerate(ordered_labels, 0)}
    df[var] = df[feature].map(ordinal_label)

Categorical_features = ['city', 'currency', 'Status','location','house_type_name']
for var in Categorical_features:
    replace_categories(merged_df, var, 'price')
```

To transform categorical variables, we need to convert their string values into numerical representations. This is done in a way that captures the monotonic relationship between the variable's label and the target. Ordinal encoding is used to transform categorical features into numerical inputs, which is necessary for certain machine-learning algorithms to work effectively.

4.3 Final Dataset



	house_size	location	city	latitude	longitude	price	numBathrooms	Status	num_of_bedrooms	house_type_name	...	beds_per_sqfeet	baths_per_beds	has_pool	gym_nearby	school/clg_nearby	wifi_facilities	valet_service	shopping_nearby	sports_playground	dining_nearby
0	7.324490	1001	3	22.974880	72.456947	10.819778	2	0	3	2	...	-6.225878	-0.405465	0	0	0	0	0	0	1	0
1	6.291569	137	3	22.990902	72.607094	9.210340	1	2	1	2	...	-6.291569	0.000000	0	0	0	0	0	0	0	0
2	7.528332	735	3	22.995766	72.467430	10.714418	3	2	3	2	...	-6.429719	0.000000	0	0	0	0	0	0	0	0
3	6.652863	416	3	19.216536	73.087166	9.680344	2	0	1	2	...	-6.652863	0.693147	0	0	0	0	0	0	0	0
4	6.522093	96	3	22.978258	72.600227	9.047821	1	0	1	4	...	-6.522093	0.000000	0	0	0	0	0	0	1	0

5 rows × 21 columns

```
final_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15292 entries, 0 to 15299
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   house_size          15292 non-null  float64
 1   location            15292 non-null  int64
 2   city                15292 non-null  int64
 3   latitude            15292 non-null  float64
 4   longitude           15292 non-null  float64
 5   price               15292 non-null  float64
 6   numBathrooms        15292 non-null  int64
 7   Status              15292 non-null  int64
 8   num_of_bedrooms     15292 non-null  int64
 9   house_type_name     15292 non-null  int64
 10  premium_house       15292 non-null  int64
 11  beds_per_sqfeet     15292 non-null  float64
 12  baths_per_beds      15292 non-null  float64
 13  has_pool            15292 non-null  int64
 14  gym_nearby          15292 non-null  int64
 15  school/clg_nearby   15292 non-null  int64
 16  wifi_facilities     15292 non-null  int64
 17  valet_service       15292 non-null  int64
 18  shopping_nearby     15292 non-null  int64
 19  sports_playground   15292 non-null  int64
 20  dining_nearby       15292 non-null  int64
dtypes: float64(6), int64(15)
memory usage: 3.1 MB
```

The final dataset consists of 15292 records and 21 features, including engineered and transformed numerical features. This dataset provides a comprehensive view of rental properties in metropolitan cities and can be utilized to uncover meaningful insights into the rental housing market.

Chapter 5. Model fitting

5.1 Linear Regression

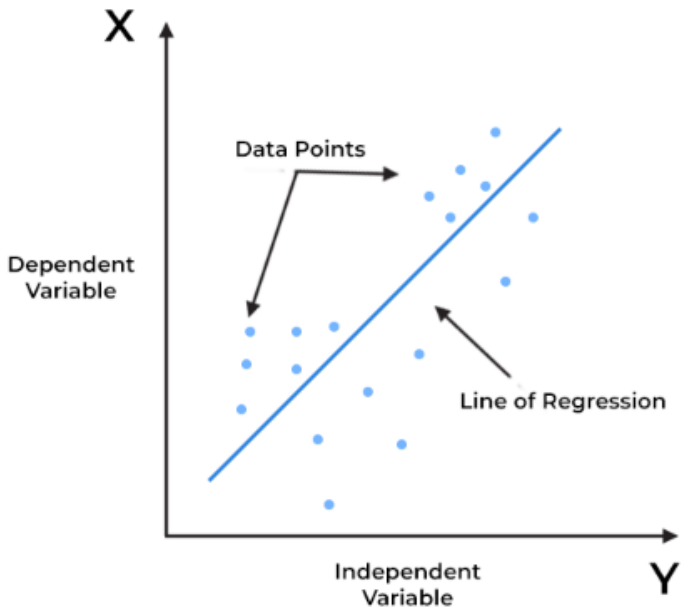
Linear regression is a statistical method used to analyze the relationship between a dependent variable (usually denoted as ”y”) and one or more independent variables (usually denoted as ”x”). The goal of linear regression is to find the linear relationship between the variables, which can be used to make predictions and understand the strength and direction of the relationship.

The linear regression model, which may be written as follows, where:

$$\hat{y}i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \varepsilon$$

where:

- β0 is the intercept.
- β0, β1, β2, ..., βn are the coefficients or slopes.
- **x1**, **x2**, ..., **xn** are dependent variables (features)
- **ε** is the error term.
- $\hat{y}i$ is predicted values.



Linear Regression aims to find the best set of coefficients (β_0, β_1, β_2, etc.) to fit the data by minimizing the difference between actual and predicted values. To achieve this, residuals are calculated for each data point, and the sum of squared residuals is computed to determine the coefficients that best fit the data.

$$residual\ sum\ of\ squares = \sum_{i=1}^{n}(y_i - \widehat{y_i})^2$$

Evaluation Metrics :

1. Mean Absolute Error (MAE):

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

2. Mean Squared Error (MSE):

$$MSE - \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

3. Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

4. R2 Score :

$$R2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \widehat{y_i})^2}{\sum_{i=1}^{N}(y_i - \bar{y})}, \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

5.1.1 LR Algorithm

Imported Libraries

```
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
```

```
# Initialize and train linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Initialize and train linear regression model
model_sc = LinearRegression()
model_sc.fit(X1_train, y1_train)
```

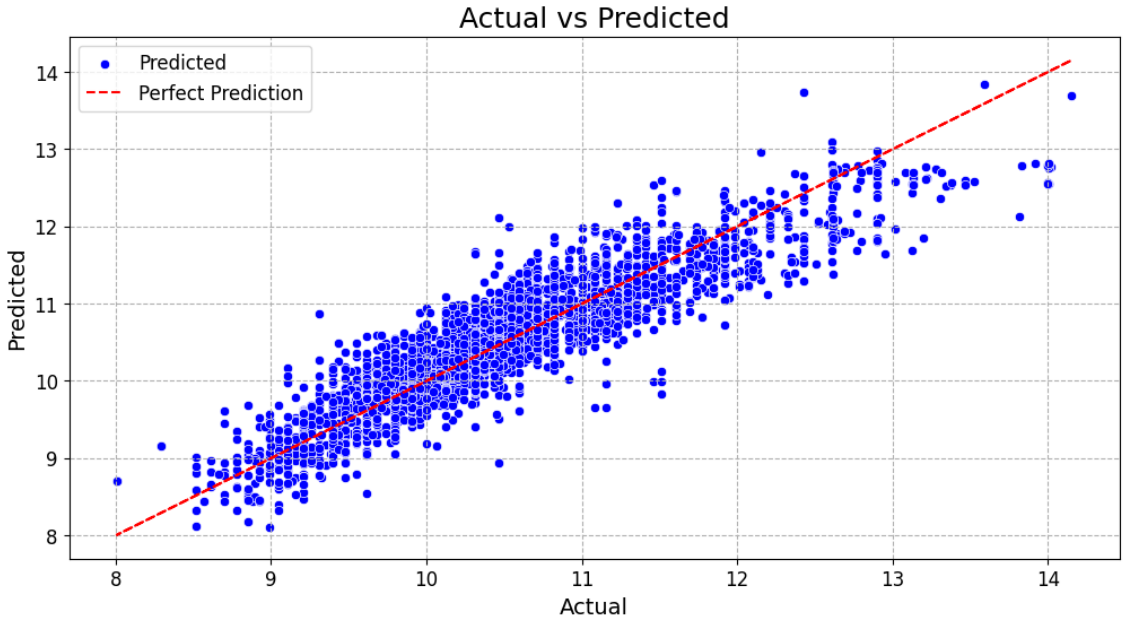
```
# Make predictions without sc
y_pred = model.predict(X_test)

# Make predictions with sc
y1_pred = model_sc.predict(X1_test)
```

```
comparison = pd.DataFrame({'actual': y_test, 'pred': y_pred})
comparison.head(10)
```

	actual	pred
5339	12.206073	12.211710
2140	11.512925	12.244077
895	10.668955	10.714526
3950	9.680344	9.958503
12716	10.373491	10.109955
6480	9.104980	9.370739
5318	11.736069	11.694357
6298	11.156251	11.294964
9971	8.853665	8.510300
4629	10.130623	10.408584

```
comparison_with_sc = pd.DataFrame({'actual': y_test, 'pred': y1_pred})
comparison_with_sc.head(10)
```

	actual	pred
5339	12.206073	12.231473
2140	11.512925	12.268666
895	10.668955	10.721136
3950	9.680344	9.960278
12716	10.373491	10.107733
6480	9.104980	9.369041
5318	11.736069	11.710547
6298	11.156251	11.307187
9971	8.853665	8.504586
4629	10.130623	10.412318



Observations :

- This graph shows how well our model's predictions match the actual values. Each blue dot represents a pair of actual and predicted values.
- The closer the dots are to the red dashed line, the more accurate our predictions are.

Evaluation :

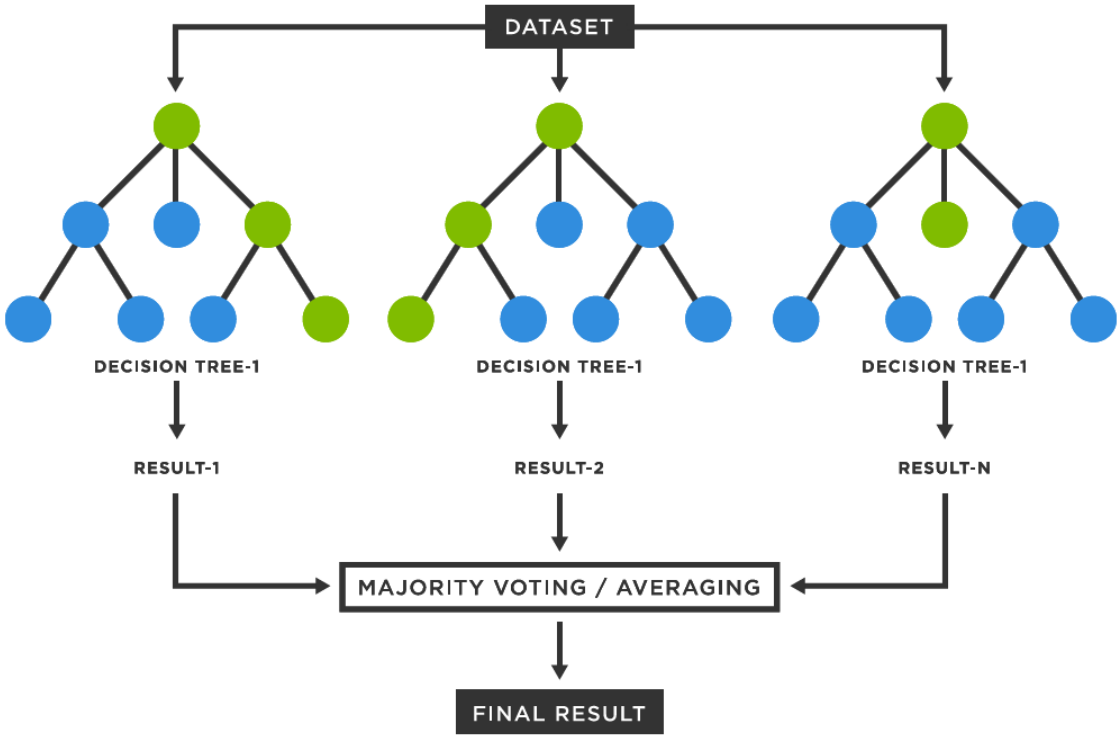
```
without scaling :

MAE :  0.296247399682877 44
RMSE :  0.3856726047432508
R2 :  0.8369372525295793

with scaling :

MAE :  0.29714647319038
RMSE :  0.38635290641902803
R2 :  0.8363614807618687
```

- Minimal differences were found between the model's performance without scaling and with it. This implies that scaling did not significantly improve the model's accuracy or goodness of fit. Therefore, scaling might not be important for this particular model and dataset.
- The model seems to perform well in predicting housing rent prices, as indicated by the low MAE and RMSE values and the high R2 score.

5.2 Random Forest

Random forest is a machine learning algorithm commonly used for supervised learning tasks such as classification and regression. It is an ensemble method that combines multiple decision trees to improve accuracy and prevent overfitting.

In a random forest, a large number of decision trees are created. Each tree is trained on a randomly selected subset of the data and a randomly selected subset of the features. Once all the trees make their predictions, the final prediction is made by taking the majority vote of all the trees. This process helps to reduce the risk of overfitting and increase the accuracy of the predictions.



5.2.1 RF Algorithm

Imported Libraries

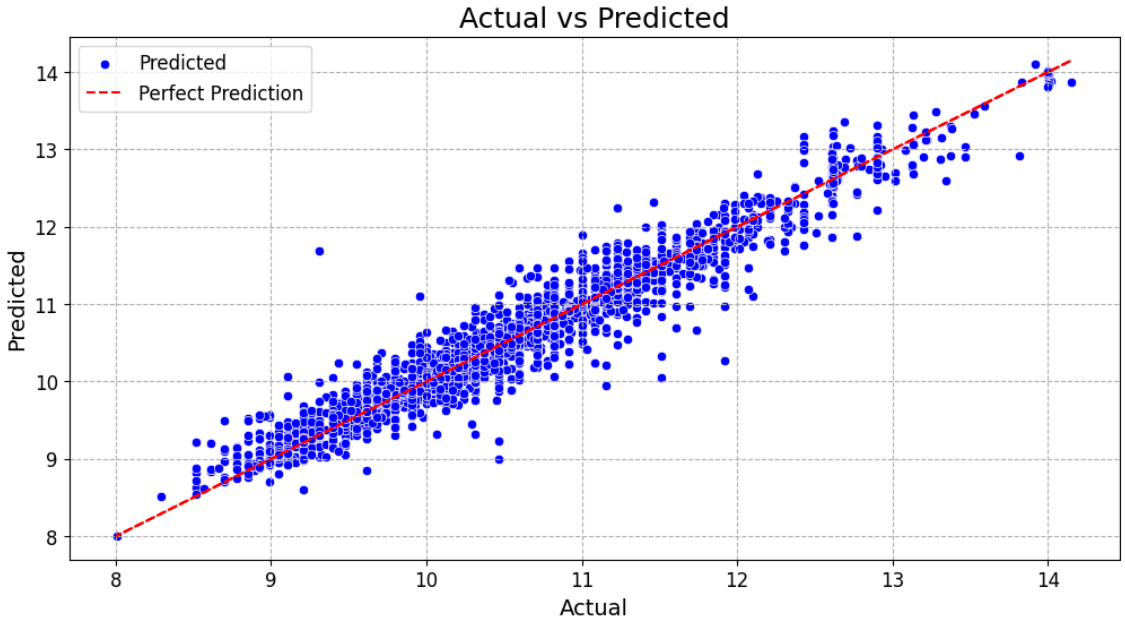
```
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
```

```
rf_model = RandomForestRegressor(random_state=42,n_estimators=100)
rf_model.fit(X_train, y_train)
```

```
# Predict on the test set
y_pred = rf_model.predict(X_test)
```

```
comparison = pd.DataFrame({'actual': y_test, 'pred': y_pred})
comparison.head(10)
```

	actual	pred
5339	12.206073	12.206982
2140	11.512925	11.533596
895	10.668955	10.623854
3950	9.680344	9.734989
12716	10.373491	10.052532
6480	9.104980	9.481137
5318	11.736069	11.754750
6298	11.156251	11.167784
9971	8.853665	8.945959
4629	10.130623	10.147331



Observations :

- This graph shows how well our model's predictions match the actual values. Each blue dot represents a pair of actual and predicted values.
- The closer the dots are to the red dashed line, the more accurate our predictions are.
- "Here, you can observe that the random forest model performs better in comparison to linear regression."

Evaluation :

```
Mean Absolute Error (MAE): 0.15274607939255413
Root Mean Squared Error (RMSE): 0.2305372724401857
R2_Score: 0.941736110303633
```

- The model seems to perform well in predicting housing rent prices, as indicated by the low MAE and RMSE values and the high R2 score.

Chapter 6. Conclusion & future scope

Metrics/Model	Linear Regression	Random Forest
MAE	0.2962	0.1527
RMSE	0.3856	0.2305
R2-Score	0.8369	0.9417

Table 6.1: Evaluation Of Linear Regression and Random Forest

- Based on the evaluation metrics, the Random Forest model performs better than the Linear Regression model in predicting rental prices of Metropolitan cities. Therefore, the Random Forest model would be preferred for making accurate predictions in the rental housing market analysis.

6.1 Findings/observations

Below are the key findings and observations derived from the comprehensive study :

- The range of rental prices in metropolitan cities is 2000 to 1500000 rupees, with an average of 69,000 rupees.
- The average house size in metropolitan cities is 1508 square feet, while the sizes of these homes range from 150 to 16000 square feet.
- The greatest number of rental homes is found in Delhi and Mumbai, a sign of strong demand in these areas.
- Delhi has the highest average rental cost compared to other large cities, with Mumbai coming in second.
- There is a higher demand for semi-furnished houses compared to furnished and unfurnished ones.
- A strong positive correlation between price and house size is shown in the correlation matrix, indicating that house size has a significant effect on the pricing of the houses.
- The dataset contains a variety of residential property types, with villas displaying a larger interquartile range than other property types. The average price is greatest for villas as well.

- The demand for 2BHK houses is observed higher than for other types, followed by 3BHK houses.
- Metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R2 score are used to compare the performance of the random forest and linear regression models. The results show that the random forest model performs better than the linear regression model in predicting rental prices in metropolitan areas, with lower MAE and RMSE values and a higher R2 score.

6.2 Challenges

- While our model works based on the features we select, there's always a possibility that other features not included in the model significantly affect housing prices. The significance of in-depth feature engineering and domain knowledge is shown by this problem. Even with our best efforts, certain significant characteristics can escape our observation or prove challenging to quantify.
- Concerning the existing data and model assumptions, the price we predict is an approximation of the rental value. It's important to recognize, though, that owners' circumstances, the state of the property, their emotional connection, and market sentiment may all have an impact on their valuation standards. Though it is limited in its ability to take into account subjective values, our model offers insightful information on pricing and market movements.
- Fluctuations in the real estate market may impact the predicted rental price. Market factors that can have significant effects on home prices include changes in supply and demand, interest rates, the state of the economy, and governmental regulations. Even though our model takes past data and trends into account, it might not fully account for unexpected shifts or external disturbances to the market.

6.3 Future plan

- It is recommended to enhance and broaden the model's feature set to encompass a greater variety of elements that impact home prices. Utilizing other data sources, like sentiment research on social media for market trends, may be required to do this.
- Include up-to-date data streams and APIs to guarantee that the model accurately represents current market circumstances and trends. This might involve forming alliances with data suppliers or creating internal data-gathering platforms to instantly catch relevant market signals.

Group Contribution

Member 1

Name: Mitul Dudhat
ID: 202318024

Member 2

Name: Patel Ayush
ID: 202318036

Contribution Of Both Members:

Significant contributions were made by both members to the EDA study on predictions of housing rent prices. They actively participated in the following processes: data collection, data cleaning (handling outliers and missing values), data visualization (applying different analytical techniques), model fitting (using random forest and linear regression), model feature engineering (extraction and selection), evaluation (evaluating the performance of random forest and linear regression), and concluding remarks (summarizing results and outlining future goals). They both also added to the knowledge of the data by examining the structure of the dataset and identifying relevant features. Both the Members actively contributed to the report and presentation.

Short Bio

1. Mitul Dudhat is a dedicated individual with a Bachelor's degree in Information Technology from K. S. School of Business Management & Information Technology. He possesses proficient programming skills in Python and SQL, as well as in Big Data, Machine Learning, and some aspects of Recommendation Systems. He considers himself proficient in several essential skills required for a Data Scientist, including programming, Machine Learning, Big Data, data visualization, communication, and problem-solving.

2. Patel Ayush is a passionate learner with a bachelor's degree in Statistics from St. Xavier's College, Ahmedabad. With a background in Statistics and Maths, my interests involve linear algebra, calculus, and machine learning. He has experience coding in various programming languages including Python, R, SQL, and SAS. He has expertise in Statistical analysis, Data Visualization, and Machine Learning techniques.

References

[1] makaan.com is used for the data collection of house rental prices in 9 metropolitan cities *URL:* https://www.makaan.com

[2] Numpy *URL:* https://numpy.org/doc/

[3] Pandas *URL:* https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

[4] Scikit learn *URL:* https://scikit-learn.org/stable/