GODREJ PROPERTIES

HOUSE PRICE PREDICTION

Created by:
Walter Edward

Introduce About the Company

Godrej Properties, part of the prestigious Godrej group in India, develops high-end residential and commercial real estate with a strong commitment to sustainability and green technology. Praised for its quality and innovative designs, the company has won many prestigious awards. Through a collaborative business model, Godrej Properties expands its influence across major Indian cities, emphasizing the creation of a quality living environment for customers.

TABLE OF CONTENTS

01 Preparation

"A machine learning model is to be proposed to predict a house price based on data related to the house i.e., its area type, availability, location, size, society, total square feet, bath and balcony using Regression."

Problem Statement

Goals of the Study

To apply data preprocessing and preparation techniques in order to obtain clean data (EDA).

To build machine learning models able to predict house price based on house features.

To analyze and compare models performance in order to choose the best model.

Data Understand

Data Information

Data contains:

- 9 Features
- 13320 rows
- Columns Location, Size, Society, Bath, Balcony have Null values.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   area_type     13320 non-null  object
 1   availability  13320 non-null  object
 2   location      13319 non-null  object
 3   size          13304 non-null  object
 4   society       7818 non-null   object
 5   total_sqft    13320 non-null  object
 6   bath          13247 non-null  float64
 7   balcony       12711 non-null  float64
 8   price         13320 non-null  float64
dtypes: float64(3), object(6)
memory usage: 936.7+ KB
```

Data Processing

'Society' column

Consider that the society column has too many null values, with 5502 values, about 41.3%.

=> Remove the society column from the dataframe

```
area_type          0
availability       0
location           1
size              16
society         5502
total_sqft         0
bath              73
balcony          609
price              0
dtype: int64
```

```
area_type        0.00
availability     0.00
location         0.01
size             0.12
society         41.31
total_sqft       0.00
bath             0.55
balcony          4.57
price            0.00
dtype: float64
```

Data Processing

Dupplicated Check

When checking Duplicate, the result returned 568 values.

=> Proceed to remove duplicate values

	area_type	availability	location	size	total_sqft	bath	balcony	price
115	Super built-up Area	Ready To Move	Marsur	2 BHK	497	1.0	1.0	20.00
145	Super built-up Area	18-Jun	Ananth Nagar	1 BHK	500	1.0	1.0	14.00
165	Super built-up Area	19-Sep	Chandapura	1 BHK	520	1.0	1.0	14.04
178	Super built-up Area	21-Jan	Kanakpura Road	1 BHK	525	1.0	1.0	30.00
184	Super built-up Area	21-Dec	Kanakpura Road	1 BHK	525	1.0	1.0	26.00
...	...	...	...	...	...	...	...	...
12364	Super built-up Area	Ready To Move	Thigalarapalya	4 BHK	3122	6.0	2.0	235.00
12365	Super built-up Area	Ready To Move	Thigalarapalya	4 BHK	3122	6.0	2.0	245.00
12744	Super built-up Area	Ready To Move	Marathahalli	4 BHK	4000	5.0	NaN	212.00
12854	Super built-up Area	Ready To Move	Sarjapur Road	4 BHK	4395	4.0	2.0	242.00
13209	Super built-up Area	18-Dec	Whitefield	4 BHK	2830 - 2882	5.0	0.0	154.50

568 rows × 8 columns

Data Understand

Data Information

Data contains:

- 8 Features
- 12752 rows
- Columns Location, Size, Bath, Balcony have Null values.

```
<class 'pandas.core.frame.DataFrame'>
Index: 12752 entries, 0 to 13319
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   area_type     12752 non-null  object
 1   availability  12752 non-null  object
 2   location      12751 non-null  object
 3   size          12736 non-null  object
 4   total_sqft    12752 non-null  object
 5   bath          12679 non-null  float64
 6   balcony       12147 non-null  float64
 7   price         12752 non-null  float64
dtypes: float64(3), object(5)
memory usage: 896.6+ KB
```

Data Processing

'availability' column

Availability is a categorical column, it will need to be converted to numeric form to train the model.

Considering, the main value is 'Ready To Move', and a specific date.

=> convert date-month values into a new representative value called 'Specific Date'.

```
availability
Ready To Move    10140
18-May             290
18-Dec             283
18-Apr             269
18-Aug             187
                  ...
16-Oct               1
16-Jan               1
15-Aug               1
15-Dec               1
17-Jan               1
Name: count, Length: 81, dtype: int64
```

```
availability
Ready To Move    10140
Specific Date     2612
Name: count, dtype: int64
```

Data Processing

'location' column

At Index=9328, location has a null value. Search based on remaining information fields that are not null, such as:

- area_type = 'Super built-up Area'
- size = '3 BHK'
- bath = 3.0
- balcony = 2.0
- price around [84.88]

We get the results as shown in the table.

After comparing and contrasting, decided to choose "Yelahanka" to replace the null value.

	area_type	availability	location	size	total_sqft	bath	balcony	price
9328	Super built-up Area	Ready To Move	NaN	3 BHK	1600	3.0	2.0	86.0

	area_type	availability	location	size	total_sqft	bath	balcony	price
8960	Super built-up Area	Ready To Move	1st Block BEL Layout	3 BHK	1540	3.0	2.0	85.0
11045	Super built-up Area	Ready To Move	5th Stage BEML Layout	3 BHK	2000	3.0	2.0	85.0
10353	Super built-up Area	The exact day	9th Phase JP Nagar	3 BHK	1780	3.0	2.0	88.0
9891	Super built-up Area	Ready To Move	Akshaya Nagar	3 BHK	1690	3.0	2.0	85.0
11041	Super built-up Area	Ready To Move	BEML Layout	3 BHK	2000	3.0	2.0	85.0
...	...	...	...	...	...	...	...	...
9247	Super built-up Area	Ready To Move	Whitefield	3 BHK	1585	3.0	2.0	88.0
8518	Super built-up Area	Ready To Move	Yelahanka	3 BHK	1491	3.0	2.0	85.0
10021	Super built-up Area	Ready To Move	Yelahanka	3 BHK	1705	3.0	2.0	85.0
9078	Super built-up Area	Ready To Move	Yelahanka	3 BHK	1556	3.0	2.0	86.0
9328	Super built-up Area	Ready To Move	NaN	3 BHK	1600	3.0	2.0	86.0

74 rows × 8 columns

Data Processing

'Location' column

Checking the unique value in the location column, discovered there are many values (1305).
Preliminarily filtering with the condition that the values are less than 10 occurrences, up to 1057 results are found.
Because these values do not repeat often (less than 10 times), but account for nearly 77% of the data, it will affect the analysis.
=> decide to convert these values into others values (representing minority values).

```
location
Whitefield               523
Sarjapur Road            379
Electronic City          286
Kanakpura Road           242
Thanisandra              229
                        ...
singapura paradise         1
CQAL LAYOUT C BLOCK        1
West of Chord Road         1
Prasanth Extension         1
arudi                      1
Name: count, Length: 1305, dtype: int64
```

```
location
Jakkur Plantation          9
Banagiri Nagar             9
Lingarajapuram             9
Volagerekallahalli         9
Vishwanatha Nagenahalli    9
                          ..
singapura paradise         1
CQAL LAYOUT C BLOCK        1
West of Chord Road         1
Prasanth Extension         1
arudi                      1
Name: count, Length: 1057, dtype: int64
```

```
location
Others                2795
Whitefield             523
Sarjapur Road          379
Electronic City        286
Kanakpura Road         242
                      ...
Gunjur Palya            10
Vasanthapura            10
BTM 1st Stage           10
Pattandur Agrahara      10
Sadashiva Nagar         10
Name: count, Length: 249, dtype: int64
```

Data Processing

'size' column

The column has 16 null values, but at the same time these values in other information fields such as bath and balcony are also null. So it will be difficult to fill all 3 columns. Due to time constraints, it was decided to remove these null values.

	area_type	availability	location	size	total_sqft	bath	balcony	price
8264	Plot Area	The exact day	Anekal	NaN	1453	NaN	NaN	16.500
11625	Plot Area	The exact day	Banashankari	NaN	2400	NaN	NaN	460.000
13148	Plot Area	The exact day	Devanahalli	NaN	1500 - 2400	NaN	NaN	46.800
13181	Plot Area	The exact day	Devanahalli	NaN	2100 - 5405	NaN	NaN	177.115
13304	Plot Area	The exact day	Hoskote	NaN	800 - 2660	NaN	NaN	28.545
7421	Plot Area	The exact day	Hosur Road	NaN	1350	NaN	NaN	8.440
8663	Plot Area	The exact day	Jigani	NaN	1500	NaN	NaN	25.490
12940	Plot Area	The exact day	Kasavanhalli	NaN	5000	NaN	NaN	400.000
13107	Plot Area	The exact day	Mysore Road	NaN	1200 - 2400	NaN	NaN	42.300
9206	Plot Area	The exact day	Others	NaN	1575	NaN	NaN	31.110
11090	Plot Area	The exact day	Others	NaN	2000	NaN	NaN	120.000
13177	Plot Area	The exact day	Others	NaN	2000 - 5634	NaN	NaN	124.000
13105	Plot Area	The exact day	Sarjapur Road	NaN	1200 - 2400	NaN	NaN	34.185
13106	Plot Area	The exact day	Sarjapur Road	NaN	1200 - 2400	NaN	NaN	28.785
11541	Plot Area	The exact day	Whitefield	NaN	2324	NaN	NaN	26.730
13104	Plot Area	The exact day	Yelahanka	NaN	1200 - 1800	NaN	NaN	12.750

Data Processing

'size' column

Columns containing complex and inconsistent values (including numbers and letters)
=> split the column into 2 new feature columns:

- 1 is no_rooms, with data type integer, containing the number of rooms for each property.
- 2 is room_types, the data type is text, representing the room type of the property.

```
array(['4 Bedroom', '7 BHK', '3 Bedroom', '1 BHK', '5 Bedroom',
       '1 Bedroom', '2 Bedroom', '1 RK', '2 BHK', '6 BHK', '6 Bedroom',
       '8 Bedroom', '9 Bedroom', '7 Bedroom', '4 BHK', '5 BHK', '3 BHK',
       '10 Bedroom', '8 BHK', '11 Bedroom', '18 Bedroom', '14 BHK', nan,
       '19 BHK', '12 Bedroom', '43 Bedroom', '9 BHK', '10 BHK', '11 BHK',
       '13 BHK', '27 BHK', '16 BHK'], dtype=object)
```

	area_type	availability	location	total_sqft	bath	balcony	price	no_rooms	room_types
0	Plot Area	Ready To Move	Sarjapur Road	1	4.0	NaN	120.0	4	Bedroom
1	Built-up Area	Ready To Move	Others	5	7.0	3.0	115.0	7	BHK
2	Plot Area	The exact day	Others	11	3.0	2.0	74.0	3	Bedroom
3	Carpet Area	Ready To Move	Others	15	1.0	0.0	30.0	1	BHK
4	Built-up Area	Ready To Move	Others	24	2.0	2.0	150.0	5	Bedroom

Data Processing

‘total_sqft’ column

The data is numeric but the data type is object because it has some interval values, and contains units of measurement such as Grounds, Guntha, Cents, Acres, Perch,...

=> Convert them to the standard unit Square Feet, and take the average value for the interval values, then force their data type to float, to match the data.

```
array(['1.25Acres', '1.26Acres', '1000 - 1285', '1000Sq. Meter',
       '1004 - 1204', '1005.03 - 1252.49', '1010 - 1300', '1015 - 1540',
       '1020 - 1130', '1042 - 1105', '1052 - 1322', '1070 - 1315',
       '1076 - 1199', '1079 - 1183', '1100 - 1225', '1100Sq. Meter',
       '1100Sq. Yards', '1115 - 1130', '1120 - 1145', '1125 - 1500',
       '1133 - 1384', '1140 - 1250', '1145 - 1340', '1150 - 1194',
       '1160 - 1195', '1160 - 1315', '117Sq. Yards', '1180 - 1630',
       '1195 - 1440', '1200 - 1470', '120Sq. Yards', '1210 - 1477',
       '1215 - 1495', '122Sq. Yards', '1230 - 1290', '1230 - 1490',
       '1235 - 1410', '1250 - 1305', '1255 - 1350', '1255 - 1375',
       '1255 - 1863', '1270 - 1275', '1300 - 1405', '1310 - 1615',
       '132Sq. Yards', '133.3Sq. Yards', '1349 - 3324', '1360 - 1890',
       '1365 - 1700', '1390 - 1600', '1400 - 1421', '1408 - 1455',
       '1410 - 1710', '142.61Sq. Meter', '142.84Sq. Meter', '1430 - 1630',
       '1437 - 1629', '1440 - 1884', '1445 - 1455', '1446 - 1506',
       '1450 - 1595', '1450 - 1950', '1469 - 1766', '1482 - 1684',
       '1482 - 1846', '1500Cents', '1500Sq. Meter', '151.11Sq. Yards',
       '1510 - 1670', '1520 - 1740', '1520 - 1759', '1550 - 1590',
       '1564 - 1850', '1565 - 1595', '1574Sq. Yards', '15Acres',
       '1610 - 1880', '1618 - 1929', '1650 - 2538', '1660 - 1805',
       '167Sq. Meter', '1750 - 2640', '1782 - 2000', '1783 - 1878',
       '1791 - 4000', '1804 - 2273', '188.89Sq. Yards', '1925 - 2680',
       '1974 - 2171', '1Grounds', '2.09Acres', '2041 - 2090',
       '204Sq. Meter', '2100 - 2850', '2150 - 2225', '2204 - 2362',
       '2215 - 2475', '2249.81 - 4112.19', '2400 - 2600', '2462 - 2467',
...
       '799 - 803', '84.53Sq. Meter', '840 - 1010', '850 - 1060',
       '850 - 1093', '854 - 960', '86.72Sq. Meter', '870 - 1080',
       '884 - 1116', '888 - 1290', '929 - 1078', '934 - 1437',
       '942 - 1117', '943 - 1220', '980 - 1030', '981 - 1249'],
      dtype=object)
```

Unit conversion table to Square Feet

Unit Name	Square Feet
Perch	272.25
Square Meter	10.764
Square Yards	9
Acres	43560
Cents	435.56
Guntha	1089
Ground	2400.35

Data Processing

'bath' column

From the data we can see that the value in the no_rooms column is similar to the bath column. This means that with the number of rooms in each property, there will be a corresponding number of bathrooms, in the sense that if the property has 3 rooms, there will be 3 bathrooms.

Since the two values are the same, the bath column no longer seems necessary, and it can add complexity to the model.
=> decided to exclude it from the analysis.

	no_rooms	bath
0	4	4.0
1	7	7.0
2	3	3.0
3	1	1.0
4	5	2.0
...	...	...
13315	2	2.0
13316	2	2.0
13317	2	2.0
13318	2	2.0
13319	2	2.0

Data Processing

'balcony' column

Because there is no exact information about null data in balcony.

So we will assume that the properties with values in the balcony column are null, meaning those properties do not have a balcony.

=> We fill null value in balcony column as 0.

	area_type	availability	location	total_sqft	balcony	price	no_rooms	room_types
11478	Built-up Area	The exact day	Hennur	2264.0	NaN	155.000	3	Bedroom
12307	Built-up Area	Ready To Move	Sarjapur Road	3004.0	NaN	158.000	3	Bedroom
12297	Built-up Area	Ready To Move	Kengeri	3000.0	NaN	130.000	8	Bedroom
12230	Built-up Area	Ready To Move	Others	2990.0	NaN	324.000	4	BHK
12192	Built-up Area	Ready To Move	Bommanahalli	2875.0	NaN	85.000	7	Bedroom
...	...	...	...	...	...	...	...	...
6261	Super built-up Area	Ready To Move	Jigani	1252.0	NaN	63.000	3	BHK
6342	Super built-up Area	The exact day	Whitefield	1256.0	NaN	73.000	2	BHK
6431	Super built-up Area	Ready To Move	Dodda Nekkundi	1264.0	NaN	52.000	2	BHK
11961	Super built-up Area	Ready To Move	Kanakpura Road	2546.0	NaN	170.000	3	BHK
13292	Super built-up Area	The exact day	Kanakpura Road	800.0	NaN	41.145	2	BHK

589 rows × 8 columns

Outlier Processing

Use IQR to detect and handle outliers.

We perform calculations, and find 2 points:

- lower_bound = q1 – 1.5 * iqr
- upper_bound= q3 + 1.5 * iqr

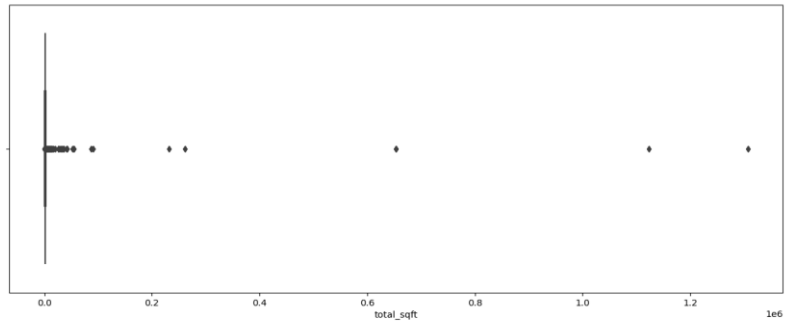
Values greater than upper_bound, or smaller than lower_bound, will be labeled as outlier, and excluded from the analysis.

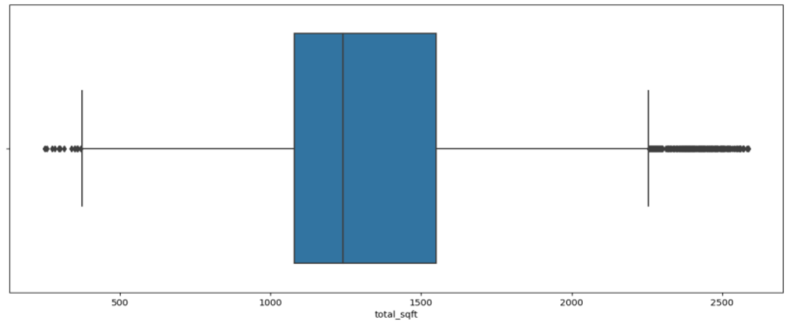
Outlier Processing

'total_sqft' column

Before processing outliers:
- The initial skew coefficient is 60.03
- Data 12736 rows

After processing outliers:
- The skew coefficient is 0.64
- Data 11577 rows



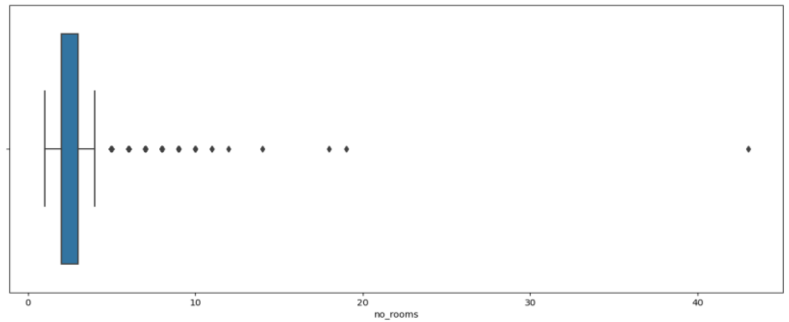


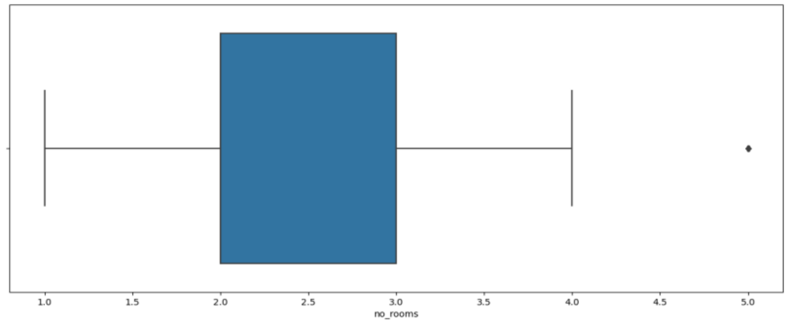
Outlier Processing

'no_rooms' column

Before processing outliers:
- The initial skew coefficient is 5.6
- Data 11577 rows

After processing outliers:
- The skew coefficient is 0.6
- Data 11216 rows



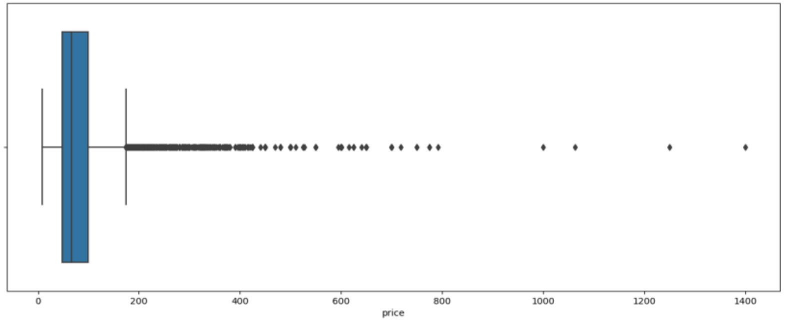


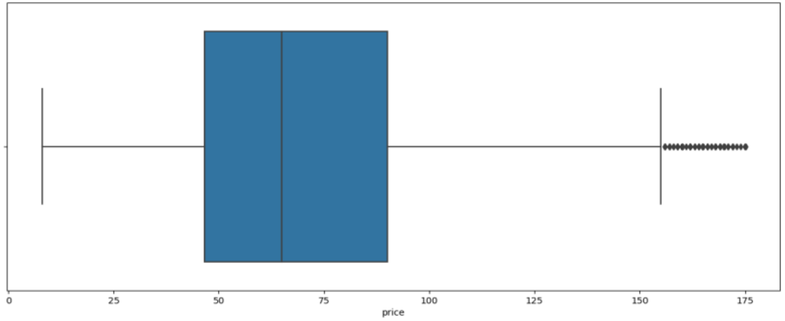
Outlier Processing

'price' column

Before processing outliers:
- The initial skew coefficient is 4.65
- Data 11216 rows

After processing outliers:
- The skew coefficient is 0.92
- Data 10470 rows





Data Understand

Data Information

Data contains:

- 8 Features
- 10470 rows
- There are no Null values.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10470 entries, 0 to 10469
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   area_type     10470 non-null  object
 1   availability  10470 non-null  object
 2   location      10470 non-null  object
 3   total_sqft    10470 non-null  float64
 4   balcony       10470 non-null  int32
 5   price         10470 non-null  float64
 6   no_rooms      10470 non-null  int32
 7   room_types    10470 non-null  object
dtypes: float64(2), int32(2), object(4)
memory usage: 572.7+ KB
```

02

Data Exploration

Univariate, and Bivariate

Data Exploration

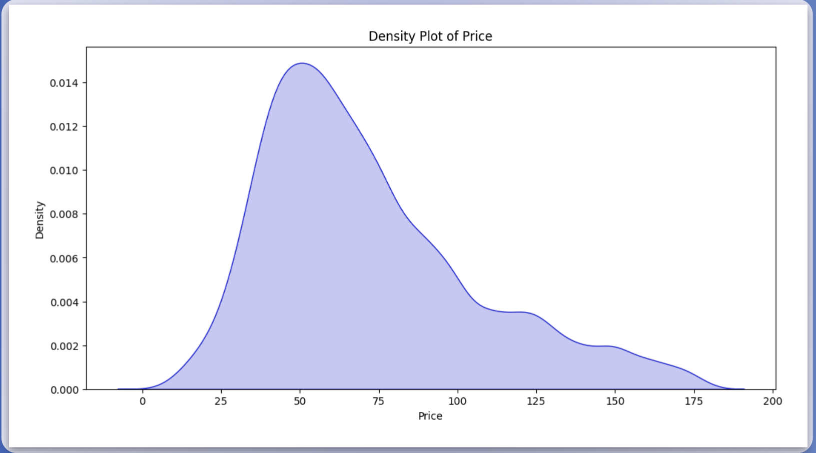
Check target column

- The average properties price is $71.826.

- The lowest priced properties is $8.

- The highest priced properties is $175

- Properties prices mainly range from $25 to $100.



	count	mean	std	min	25%	50%	75%	max
price	10470	71.826	33.93	8	46.64	65	90	175

Data Exploration

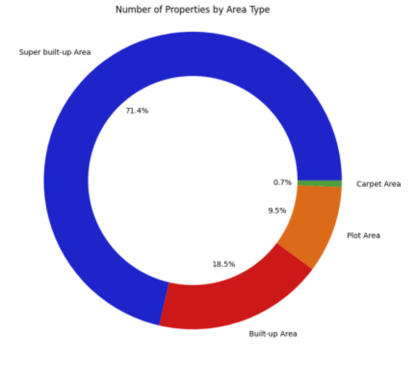
'area_type' column

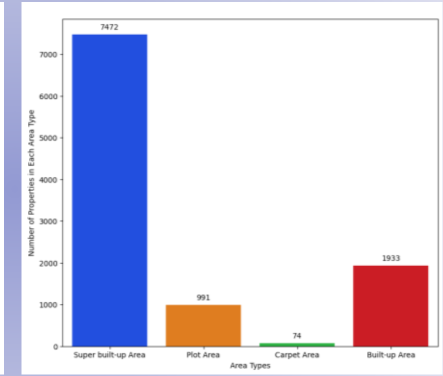
- Super built-up area: is the area with the largest number of houses (7472, accounting for 71.4%), is a densely populated area, most likely the city center.

- Built-up area: although not equal to Super built-up area, it is still an area with a high number of houses, and accounts for a significant proportion (1933, accounting for 18.5%), possibly a newly developed urban area, with Good infrastructure and services.

- Plot area: has a significant number of houses (991, accounting for 9.5%), is a moderate-sized residential area.

- Carpet area: is the area with the fewest houses (74, accounting for 0.7%), sparsely populated, may be rural, or a less developed area with limited services.



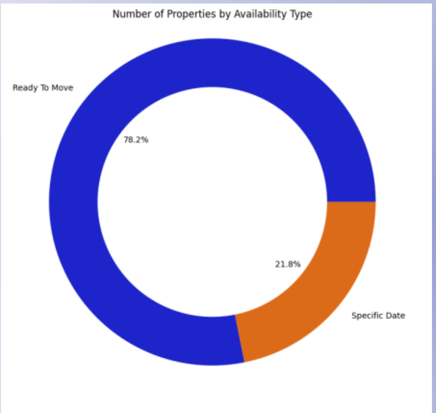


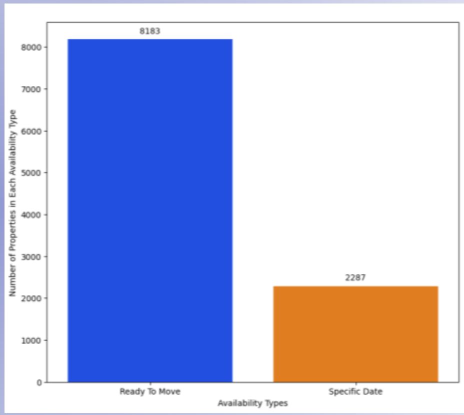
Data Exploration

'availability' column

- Ready to move: accounts for 78.2%, showing that the majority of houses are ready for buyers or renters to move in without waiting.

- Specific Date: houses where the move-in date can be fixed in advance. Only accounts for 21.8%. Suitable for those who have specific plans regarding moving time, while waiting for financial or legal procedures, or in accordance with personal needs.

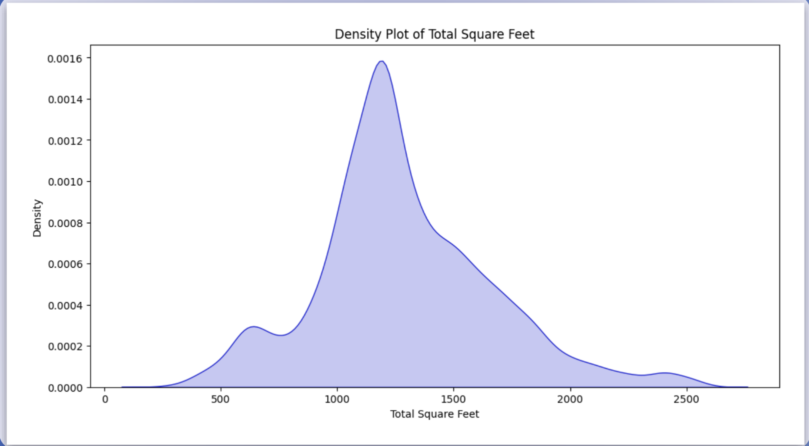




Data Exploration

'total_sqft' column

- The average house area is 1245.74 square feet.
- The smallest house is 435 square feet.
- The house has the largest area of 2130 square feet.
- House sizes mainly range from 1060 square feet to 1450 square feet.



	count	mean	std	min	25%	50%	75%	max
Total_sqft	10470	1284.95	378.56	250	1070	1221	1500	2585.5

Data Exploration

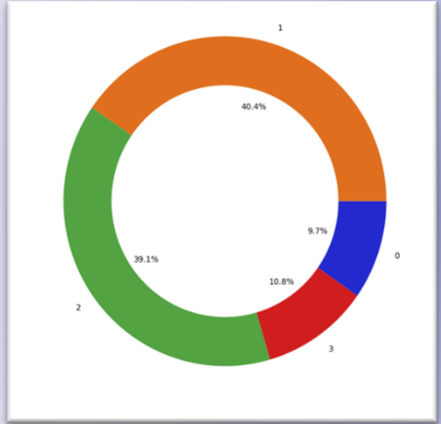
'balcony' column

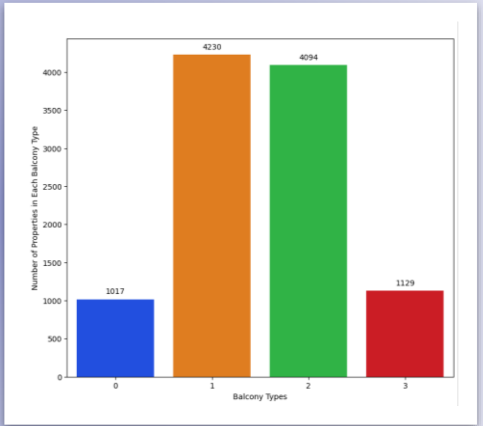
- None: accounts for about 10%, possibly due to small space, or to make the price more affordable. Suitable for people who are less interested in outdoor space, or in crowded urban areas with limited space.

- 1 balcony: accounts for the largest number, 40.4%. Shows that the majority of users prioritize having at least 1 outdoor space, used for many purposes such as growing plants, resting, or drying clothes.

- 2 balcony: very high quantity, only 1% lower than 1 balcony. Shows a significant preference for houses with 2 balconies.

- 3 balcony: accounts for a low but still significant proportion (10%). Reflecting the needs of a high-end customer segment that desires luxury and larger outdoor spaces.





Data Exploration

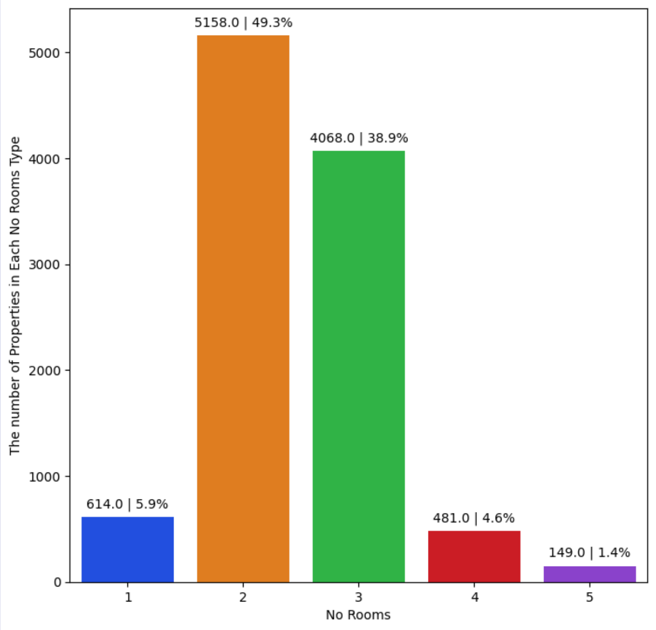
'no_rooms' column

- 1 room: not a high proportion, demand for this type of house is quite low. Can be a small apartment, suitable for individuals who do not have a need for large space, and want a more economical option.

- 2 rooms: this is the most popular type, accounting for half of the market. Reflecting that customers have high needs for space, such as 1 bedroom, and 1 other room that can be used as an office or living room.

- 3 rooms: accounts for a significant proportion of 39%. Similar to 2 rooms, it can be used as a bedroom, office, and entertainment space. But maybe because the price is higher, there is less demand for 2 rooms.

- Greater than 3 rooms: accounts for a low percentage, only about 5%, showing that the demand for houses with many rooms is quite limited.



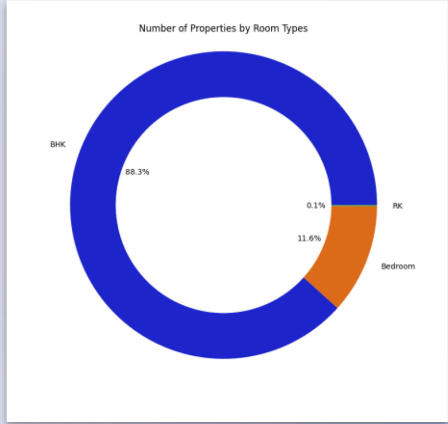
Data Exploration

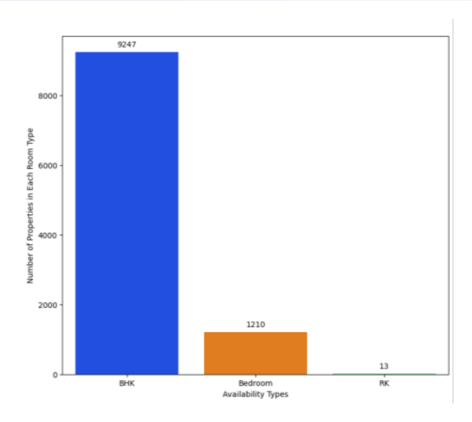
'room_types' column

- BHK: means a house with at least 1 bedroom, 1 living room, and 1 kitchen. It is a popular type of house, accounting for an overwhelming rate of 88.3%. Reflects widespread demand for fully furnished living spaces.

- Bedrooms: this category only includes bedrooms, and may not include kitchens, or living rooms. This type only accounts for a small part, about 11.6%. Suitable for students or single people who don't need a lot of common space or amenities like a kitchen.

- RK: The number of houses of this type is very small, accounting for only 0.1% of the market. Suitable for low-income people, or individuals who live simply and do not need much private space.





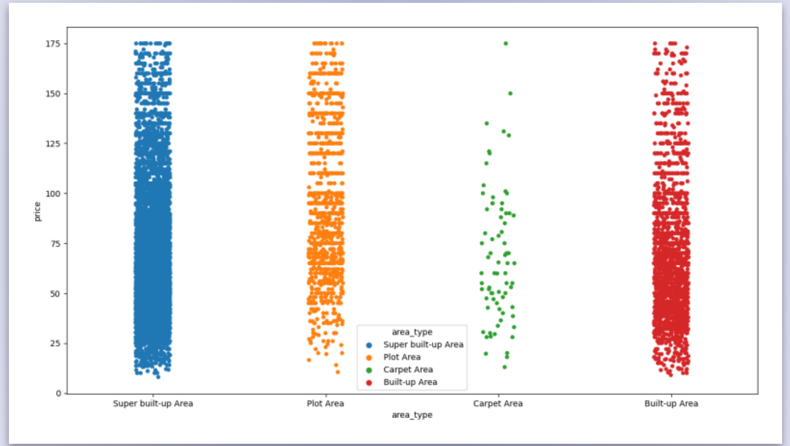
Data Exploration

‘area_type’ and ‘price’ columns

- Super built-up Area: the area with the densest data density (where there are the most houses), house prices are relatively evenly distributed, and close to each other. It shows that house prices in this area are quite stable, and have little variation, when prices range from $10 to $175. However, prices tend to focus from $15 to $130.

- Built-up Area and Plot Area have similar characteristics to Super built-up Area, however, house prices in Built-up Area are concentrated at $25 to $100, while Plot Area is from $50 to 100$.

- Carpet area: the area with the sparsest data density (the place with the fewest houses), house prices are relatively unevenly distributed, and far apart. Shows that house prices in this area are quite volatile, and less stable than the other 3 areas.



Data Exploration

'availability' and 'price' columns

In terms of home availability, both have almost the same data density. House prices are relatively evenly distributed and close together, showing that house prices are stable, and not much affected by house availability. At the same time, house prices tend to concentrate from $25 to $100.

Data Exploration

‘total_sqft’ and ‘price’ columns

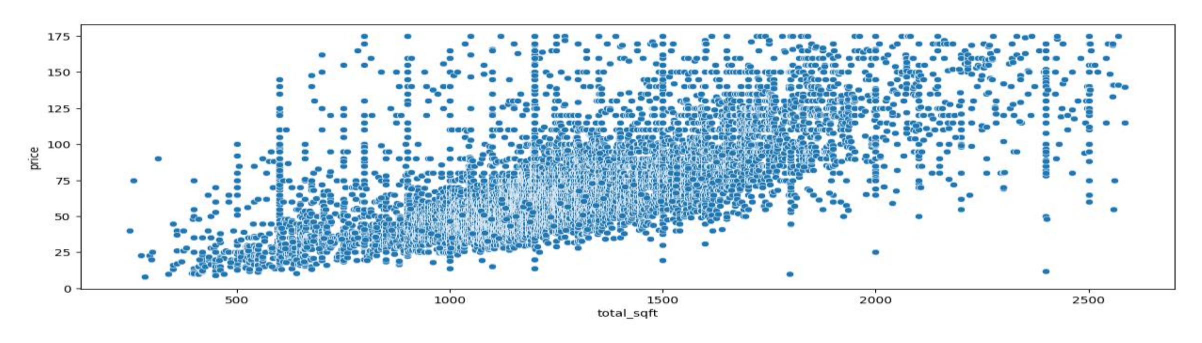
The data points are concentrated in a straight line increasing from left to right, showing a positive relationship between total_sqft and price, meaning the larger total_sqft, the higher the price will be.

- total_sqft ranges from 500 to 900, with prices ranging from $10 to $60
- total_sqft ranges from 900 to 1500, with prices ranging from $25 to $100
- total_sqft is between 1500 and 2000, with prices ranging from $50 to $130

⟹are areas with dense data density (with the most houses), are popular segments that customers are often interested in, and need special attention.

- Total_sqft from 1500 to 2500, but price only from 8 to 50 dollars.
- Or properties with total_sqft under 500, but priced at 75 to 100 dollars.

⟹Are data points that fall far from the main trend, classified as unusual cases, requiring further analysis to find the cause.

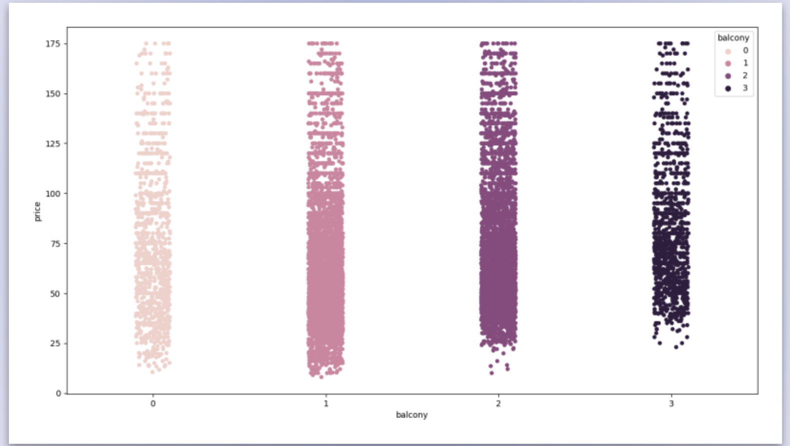


Data Exploration

‘balcony’ and ‘price’ columns

- In terms of data density, balconies with values of 1 and 2 have denser data than 0 and 3. It can be understood that properties with 1 or 2 balconies will be more popular than 0 or 3.

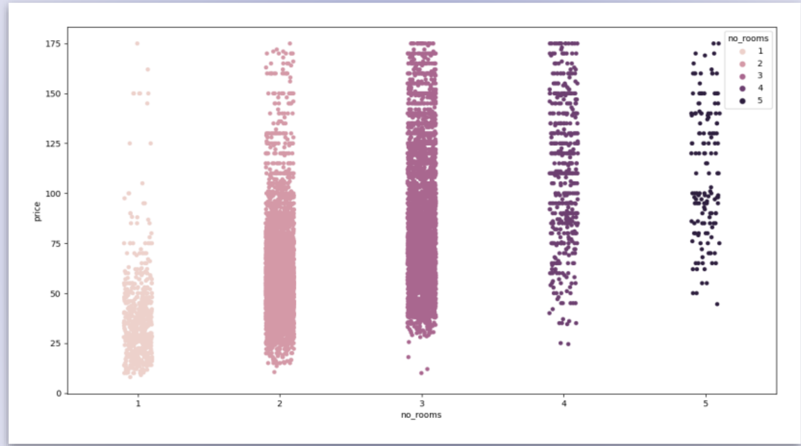
- House prices are distributed relatively evenly and close to each other, showing that no matter how many balconies a house has, the price is still relatively stable, and has little variation, ranging from $10 to $175. However, for properties with 1 or 2 balcony, prices tend to range from $25 to $100.



Data Exploration

'no_rooms' and 'price' columns

- 2 and 3 rooms: have the densest data density (most properties have 2 or 3 rooms, especially 3 rooms), house prices are relatively evenly distributed, and close to each other. House prices are quite stable, and have little variation, in the range from $25 to $175. For 2-room houses, prices tend to range from $20 to $110, while 3-room houses usually range from $30 to $130.
- 4 and 5 rooms: data density is not dense but evenly distributed (popular but not equal to 2 and 3 rooms). The fact that house prices are evenly distributed and close to each other shows price stability. Prices usually range from $40 to $175, spread evenly and do not tend to be concentrated.
- 1 room: dense data density ranges from $10 to $75 (for properties with 1 room, the price will usually range from $10 to $75), they are evenly distributed and close to each other. That means in the price range from $10 to $75, house prices are relatively stable. With house prices higher than $75, the density is sparse (there are very few 1-room houses priced over $75), unevenly distributed, and far apart (prices are less stable, and tend to fluctuate).

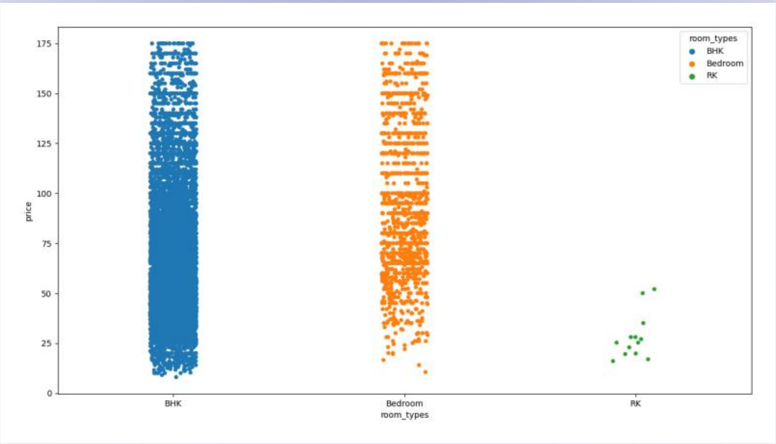


Data Exploration

‘room_types’ and ‘price’ columns

- BHK and Bedroom: 2 types of rooms with the densest data density (the 2 types with the most houses), house prices are relatively evenly distributed, and close to each other. Shows that house prices are stable, and less volatile, evenly distributed from $10 to $175. However, the price of houses where the room type is Bedroom does not tend to be concentrated, but BHK tends to be concentrated from 10$ to 130$.

- RK: room type has the sparsest data density (has the fewest houses), house prices are mainly distributed from $10 to $50, but unevenly and far apart, meaning house prices often fluctuate, and less stable.



03 Model Prediction

You can enter a subtitle here if you need it

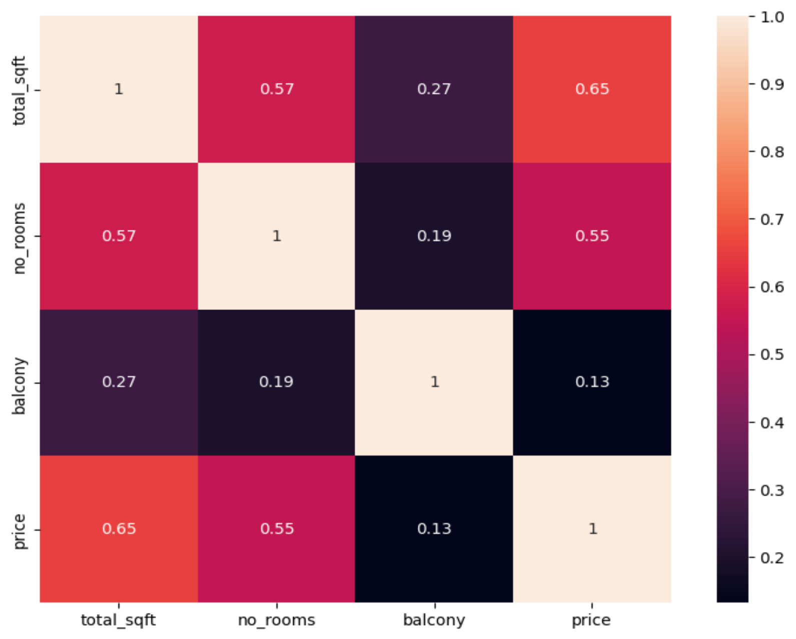
Model Prediction

Correlation

Through the process of examining the correlation of attributes, there are 3 factors that have the most impact on our target variable Price.

In particular, ranked first is total_sqft (65%), second is no_rooms (55%), and third is balcony (13%)

We will focus more on these three factors when doing the next work.



Model Prediction

Label encoding

```
area_type: ['Super built-up Area' 'Plot Area' 'Carpet Area' 'Built-up Area']
availability: ['Ready To Move' 'The exact day']
location: ['Others' 'Hennur Road' 'Yelahanka New Town' 'Nagarbhavi' 'Malleshwaram'
 'Rajaji Nagar' 'Kengeri' 'Ramamurthy Nagar' 'Electronics City Phase 1'
 'Shampura' 'BTM 1st Stage' 'Kanakpura Road' 'Attibele' 'Nagavara'
 'Yeshwanthpur' 'Anekal' 'Kengeri Satellite Town' 'Chandapura'
 'Electronic City' 'Vasanthapura' 'Magadi Road' 'Rachenahalli'
 'Kodichikkanahalli' 'BTM Layout' 'Ulsoor' 'Hulimavu' '8th Phase JP Nagar'
 'Kaval Byrasandra' 'Hosur Road' 'Vidyaranyapura' 'Rayasandra'
 'Sarjapur Road' 'Kenchenahalli' 'Amruthahalli' 'Vijayanagar' 'Marsur'
 'Bannerghatta Road' 'Ananth Nagar' 'Raja Rajeshwari Nagar' 'Indira Nagar'
 'Banashankari' 'Banashankari Stage II' 'Mysore Road' '7th Phase JP Nagar'
 'Kodigehalli' 'Shivaji Nagar' 'Chamrajpet' 'Hoskote' 'Bommanahalli'
 'Whitefield' 'Singasandra' 'Neeladri Nagar' 'Sarjapur' 'Ramagondanahalli'
 'Kogilu' 'Kammanahalli' 'Electronic City Phase II' 'Dasanapura'
 'Sultan Palaya' 'Doddathoguru' 'Kanakapura' '5th Phase JP Nagar'
 'Margondanahalli' 'Frazer Town' 'Haralur Road' 'Cooke Town' 'Yelahanka'
 'Thyagaraja Nagar' 'Thanisandra' 'Ganga Nagar' 'Kumaraswami Layout'
 'Banjara Layout' 'Kothannur' 'Gubbalala' '6th Phase JP Nagar' 'Arekere'
 'Vishveshwarya Layout' 'Basavangudi' 'KR Puram' 'Jigani'
 'Vishwapriya Layout' 'Jalahalli' '2nd Stage Nagarbhavi' 'OMBR Layout'
 'Anjanapura' '9th Phase JP Nagar' 'Mallasandra' 'TC Palaya'
 'Chikkabanavar' 'Kereguddadahalli' 'HAL 2nd Stage' 'Battarahalli'
 'Subramanyapura' 'Naganathapura' 'Banashankari Stage VI' 'Nagasandra'
 'Hosakerehalli' 'Giri Nagar' 'JP Nagar' 'Cox Town' 'Sonnenahalli'
...
 'Lakshminarayana Pura' 'HRBR Layout' 'Dasarahalli' 'Narayanapura'
 'Murugeshpalya' 'Vittasandra' 'Benson Town' 'Dodsworth Layout'
 'Ambedkar Nagar' 'Iblur Village' 'Lingadheeranahalli' 'Chikka Tirupathi']
room_types: ['BHK' 'Bedroom' 'RK']
```

```
area_type: [3 2 1 0]
availability: [0 1]
location: [190 102 243 180 168 200 143 203  80 214  25 135  23 182 245  19 144  59
  78 234 163 198 146  27 230 111   9 141 110 235 204 209 142  16 236 171
  38  18 199 115  30  31 175   8 148 215  58 109  51 240 216 185 208 202
 150 132  79  67 221  73 134   6 170  81  96  65 242 227 224  83 157  36
 153  89   7  22 237  39 122 120 238 118   4 187  20  10 166 222  62 145
  92  41 220 178  34 181 108  85 116  66 219 169 128 191 133 107  17 160
 104 154 206  75  71  43  29 183 177  69  49 231 193  56 159  91 127  76
 232  93  82 105 126  50  40 156  57 103 229 106 119 228 155  64 164  12
 158 210  13 140 138  14 125 192 139 211 194 213  99 197  26 241  52 172
  47  60  37  98  87  42 130 218  54   3  70 117 233  44 113 129   2  86
 147  35  63 131 151 212 121 176  32 195  33 223 207 189  97  88  55 196
 188   0 173 124  48 136 205 101 186 123 201 226  24 112   1 225 149  53
 100 152  11  95  72 217  28 165 167 244 137  84  90  21 179   5  46  77
 161  94  68 184 174 239  45  74  15 114 162  61]
room_types: [0 1 2]
```

Use a for loop, loop through each column in the data frame whose dtype is 'object'.
Then use the sklearn.processing.LabelEncoder library to convert the column using the encoder.

Model Prediction

Train Test Splits

Set:
- The target variable is y, containing the price column value in the dataframe
- The input variable is X, containing the remaining values in the data frame, after removing the price column.

Use the sklearn.model_selection.train_test_split library to split the data into 4 parts, X_train, X_test, y_train, y_test, with test_size of 0.2, and random_state of 42.

	Rows	columns
Training feature set size	8376	7
Test feature set size	2094	7
Training variable set size	8376	,
Test variable set size	2094	,

Model Prediction

Try various ML models and train them

Use Regressor model for training, and use K-fold cross validation for evaluation.
In K-Fold cross-validation, choose cv=10, which means dividing the training data into 10 training times and evaluating the regression model 10 times, resulting in an array of 10 points.

By finding the root mean square error(rmse) between prediction and reality. It measures the standard deviation of the errors the system makes in its predictions.

RMSE (Root Mean Squared Error)

The specific formula to calculate RMSE from the predicted values and the actual values is:

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

where:

- $\hat{y}_i$ is the predicted value for the ith observation.
- y_i is the actual value for the ith observation.
- n is the total number of observations.

Standard Deviation

The specific formula to calculate the standard deviation from a data set is:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}}$$

where:

- x_i is the ith value in the data set.
- μ is the mean of the data set, calculated as $\frac{\sum_{i=1}^{n} x_i}{n}$.
- n is the total number of observations.

Model Prediction

Try various ML models and train them

- We have a prediction error of 21.0005 with +/- 0.8130

- Ensemble learning with Random Forest looks very promising and performs best among the five methods. This is a model that fits the training data. When this happens, it can mean that the features provide enough information to make accurate predictions or that the model is robust enough.

	Mean	Standard Deviation
Linear Regression	23.4680	0.9145
Lasso Regressor	23.6869	0.8781
Ridge Regressor	23.4680	0.9144
Decision Tree Regressor	26.8819	0.8294
Random Forest Regression	21.0005	0.8130

Model Prediction

Random Forest Regressor

Fine-tune the model with hyperparameters.

Hyperparameters are "knobs" that we adjust to optimize output.
This can be done manually, but we can use the GridSearchCV tool, to do this automatically.

We will apply it to our RandomForest model.

Hyperparameter	values
n_estimators	[200, 400]
max_depth	[20, 40]
min_samples_split	[2, 4]
min_samples_leaf	[1, 2]
max_features	['auto', 'sqrt']
bootstrap	[True, False]

Model Prediction

Random Forest Regressor

Use hyperparameters to train, fit the model, and predict.

Once completed, re-evaluate the model using the R-Squared Score value:

- 0.8067.

Above 80%, the model is strong enough, and can make accurate predictions.

Hyperparameter	values
n_estimators	400
max_depth	40
min_samples_split	4
min_samples_leaf	2
max_features	‘sqrt’
bootstrap	True

Model Prediction

Random Forest Regressor

- R-Squared = 0.63: means the model can explain about 63% of the variation in house price data. This is a relatively good result but still needs improvement.

- Mean Absolute Error (MAE) = 14.1: on average, the model's predicted value differs by about $14.1 from the actual value.

- Mean Squared Error (MSE) = 403.51 and Root Mean Squared Error (RMSE) = 20.09: High MSE indicates the presence of some large deviations in the predicted values. RMSE is the square root of MSE, providing an idea of the average error in the same units as the house price. This value is also quite high, suggesting that the model can be improved to minimize large errors.

evaluation index	values
R-Squared	0.63
Mean Absolute Error (MAE)	14.1
Mean Squared Error (MSE)	403.51
Root Mean Squared Error (RMSE)	20.09

04 Suggestion

Suggestion

Suggested Improvements to the Model

1. Collect more data: If possible, try to collect more data to get broader coverage of the different characteristics of different house types and locations.

2. Handle input variables: Check and handle outliers or missing data in variables such as total_sqft, no_rooms. Consider converting categorical variables like area_type, room_types using one-hot encoding to improve the model.

3. Select and refine features: Perform importance analysis of features and eliminate features with little information, which can help improve model performance. Test adding new features that may be useful, such as distance to the city center, surrounding amenities.

4. Model tuning: Experiment with different regression models (e.g. Ridge regression, Lasso, ElasticNet) and tweak hyperparameters to see if it is possible to improve R^2 and reduce MSE, RMSE.

5. Cross-validation: Use cross-validation techniques to evaluate the model more thoroughly, helping to avoid overfitting and more accurately evaluate model performance.

6. Error analysis: Analyze specific cases where the model predicts large deviations to understand the reasons and find ways to fix them.

THANKS!

Do you have any questions?
nhudaitran1510@gmail.com
+96 862 93 64
www.linkedin.com/in/ddaitran/