HOUSING RENT ANALYSIS AND PREDICTION

Contents:

Overview:

This project utilizes a comprehensive dataset of residential rental properties from major urban cities in India, sourced from Kaggle. The dataset incorporates a wide array of property attributes, including locality, size, amenities, and other relevant features that influence rental prices. Rigorous exploratory data analysis was conducted to uncover key trends, distribution patterns, and the principal factors driving rent values within urban markets.

Building on these insights, advanced machine learning regression techniques such as XGBoost and Random Forest were implemented to develop predictive models for estimating rental prices. Comparative evaluation of model accuracy and residual patterns provided a data-driven foundation for understanding the dynamics of rental pricing and optimizing predictions.

Dataset properties and cleaning:

The dataset offers a comprehensive perspective on the numerous factors shaping rental prices in urban India. It consists of approximately 4k records featuring multiple data types.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Posted On          4746 non-null   object
 1   BHK                4746 non-null   int64
 2   Rent               4746 non-null   int64
 3   Size               4746 non-null   int64
 4   Floor              4746 non-null   object
 5   Area Type          4746 non-null   object
 6   Area Locality      4746 non-null   object
 7   City               4746 non-null   object
 8   Furnishing Status  4746 non-null   object
 9   Tenant Preferred   4746 non-null   object
 10  Bathroom           4746 non-null   int64
 11  Point of Contact   4746 non-null   object
dtypes: int64(4), object(8)
memory usage: 445.1+ KB
```

The “Posted On” column was converted to a datetime datatype instead of remaining as an object. Additionally, the column names were updated to a more standardized format.

	posted_on	bhk	rent	size	floor	area_type	area_locality	city	furnishing_status	tenant_preferred	bathroom	point_of_contact
0	2022-05-18	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
1	2022-05-13	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
2	2022-05-16	2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
3	2022-07-04	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
4	2022-05-09	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished	Bachelors	1	Contact Owner

No null/blank values were found in the dataset. Two duplicates were dropped form the dataset

```
rent_ds[rent_ds.duplicated()]
```

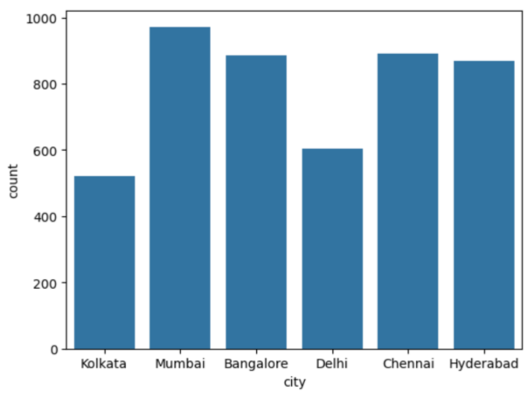
	posted_on	bhk	rent	size	area_type	area_locality	city	furnishing_status	tenant_preferred	bathroom	point_of_contact	mapped_floor
72	2022-06-26	2	16000	850	Carpet Area	Salt Lake City Sector 1	Kolkata	Semi-Furnished	Bachelors	1	Contact Agent	1
429	2022-06-03	2	5500	450	Carpet Area	Bisharpara	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner	1

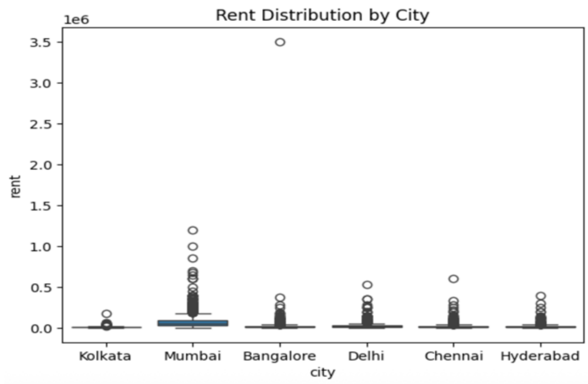
```
def get_missing_report(df):
    missing_df = (df.isnull().sum()/len(df)).rename_axis('columns').to_frame('missing_perc').reset_index()
    missing_df['missing_perc'] = missing_df['missing_perc'] * 100
    missing_df['type'] = missing_df['columns'].apply(lambda col: str(df[col].dtypes))
    return missing_df.sort_values(by = 'missing_perc', ascending=False)
get_missing_report(rent_ds)
```

	columns	missing_perc	type
0	posted_on	0.0	datetime64[ns]
1	bhk	0.0	int64
2	rent	0.0	int64
3	size	0.0	int64
4	floor	0.0	object
5	area_type	0.0	object
6	area_locality	0.0	object
7	city	0.0	object
8	furnishing_status	0.0	object
9	tenant_preferred	0.0	object
10	bathroom	0.0	int64
11	point_of_contact	0.0	object

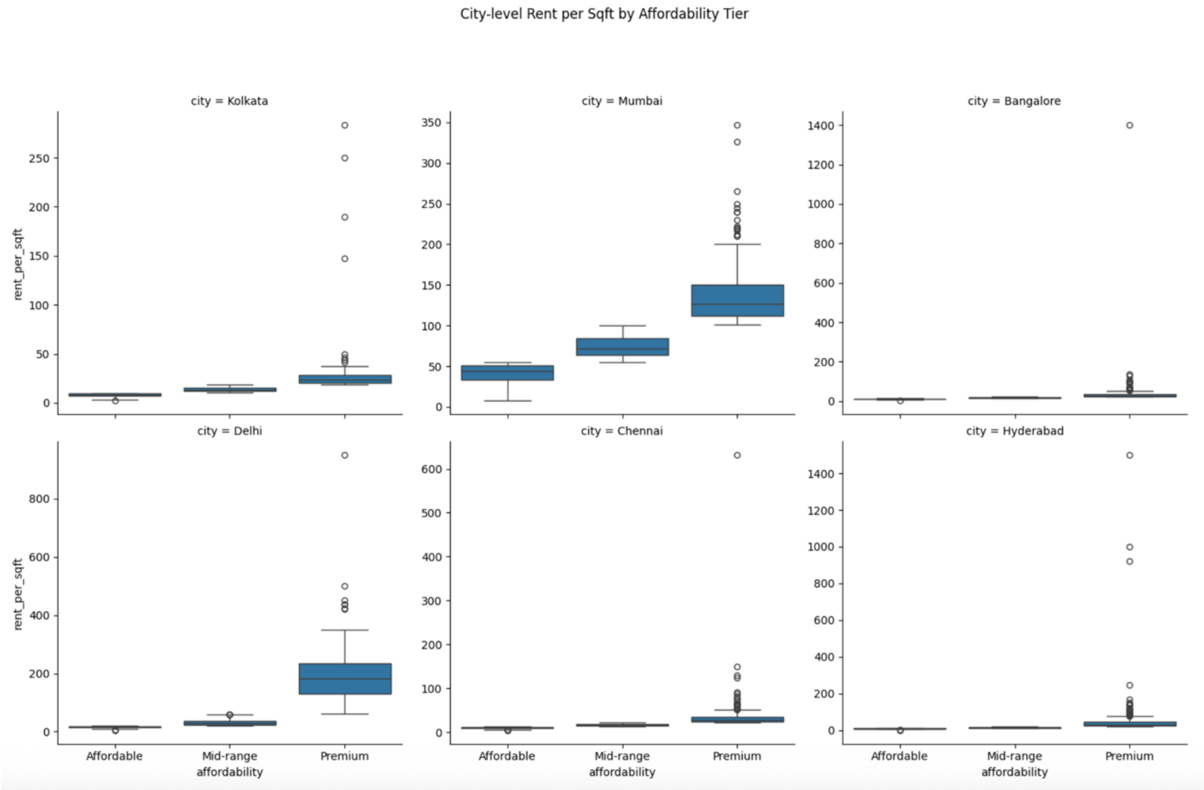
Exploratory Data Analysis:

Distribution of records across cities is uneven, with Mumbai dominating and Kolkata having the least. Other cities like Bangalore, Chennai, and Hyderabad are fairly balanced in representation. Outliers are more prominent in Mumbai than any other city.





To better analyse rental trends, a new column “rent_per_sqft” was derived for each property. Based on this metric, localities listings were segmented into three categories , Affordable, Mid-range, and Premium. The segregation was done using the 25th and 75th quantiles of rent per sqft distribution for each city.



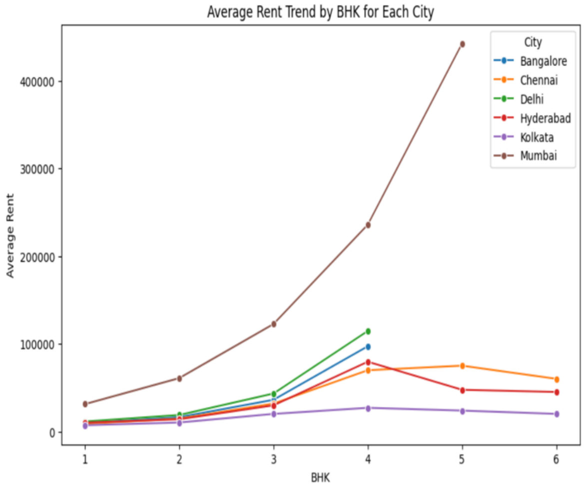
Outlier detection was performed for each city using a modified version of the Interquartile Range (IQR) method. The approach was designed to be dynamic, applying a more lenient threshold for lower values since lower rent per sqft is more common and often reflects genuinely affordable properties. In contrast, a stricter threshold was applied to higher values to eliminate inconsistent data points caused by abnormally high rents or anomalously small house sizes . This adjustment aligns with the box-plot observations, where upper outliers are noticeably more frequent than lower ones.

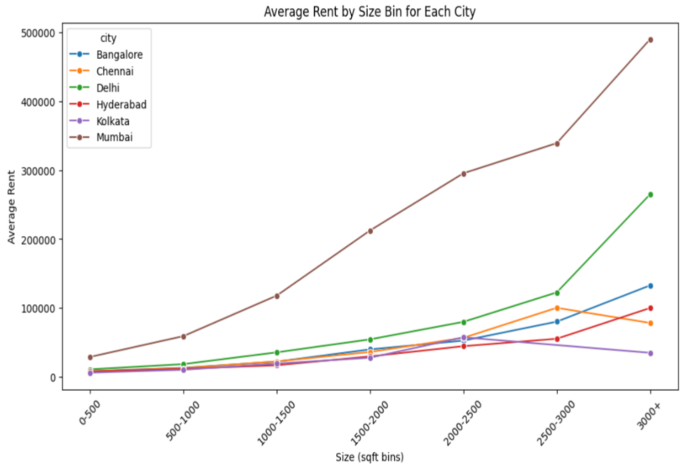
```python
for city, group in df.groupby('city'):
    q1 = group['rent_per_sqft'].quantile(0.25)
    q3 = group['rent_per_sqft'].quantile(0.75)
    iqr = q3 - q1
    lower = q1 - 0.5 * iqr
    upper = q3 + 3 * iqr

    # Mark rows that are outliers for this city
    city_outliers = group[(group['rent_per_sqft'] < lower) | (group['rent_per_sqft'] > upper)]
    outlier_rows.append(city_outliers)
```

-Trend of rent in each city :

The average rent trend by bhk shows a significant exponential increase in Mumbai, with similar upward trends observed in Delhi and Bangalore. In contrast, Hyderabad and Chennai exhibit a decline in average rent beyond 4 BHK, while Kolkata's rent remains relatively stable across bhk categories.

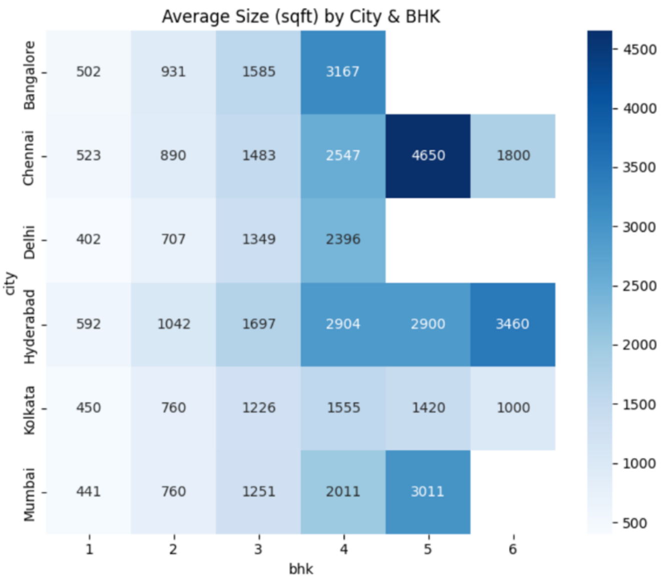
When examining average rent by property size, most cities show a consistent upward trajectory with increasing size, except Chennai which reflects a decline after 3000sqft. This is likely because houses larger than 3000 sqft are located in more affordable areas within the city.This indicates that BHK count and property size do not have a directly proportional relationship at the city level.

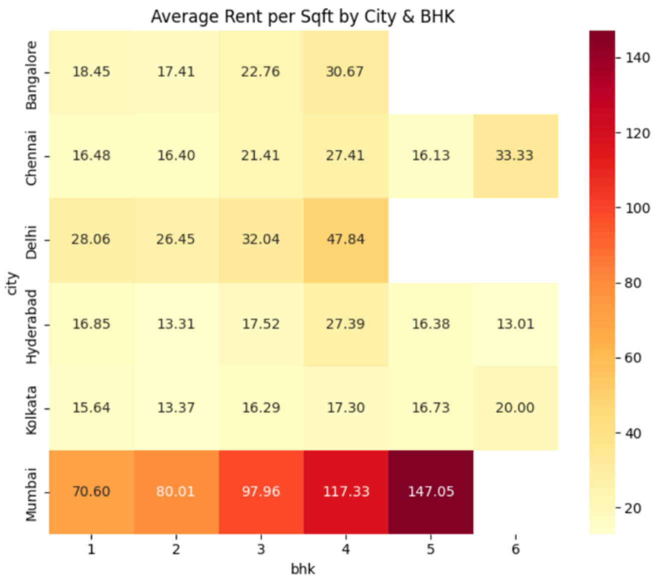




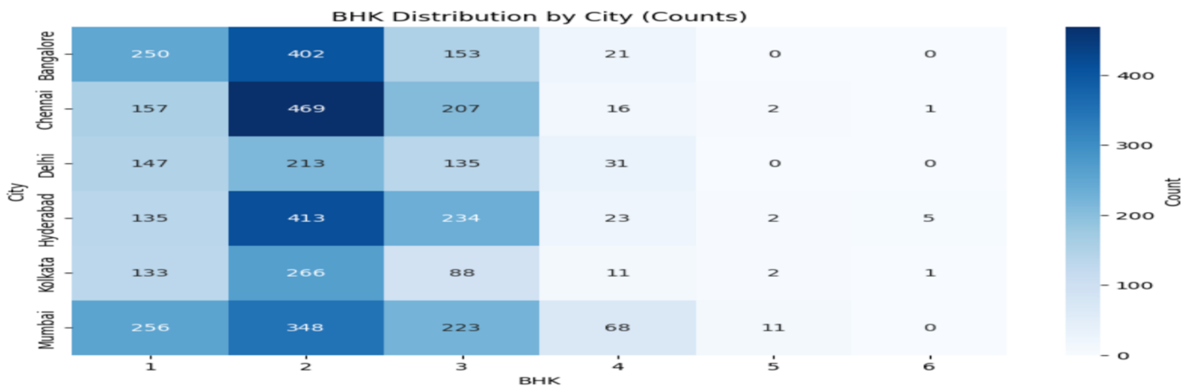
Across most cities, property size increases significantly with higher bhk counts, though this trend shows exceptions in Chennai and Kolkata, where average sizes decline beyond certain bhk levels with Chennai experiencing a notable decrease for 6 BHK properties.

Despite larger sizes, average rent per sqft does not always follow the same upward pattern. In some cases, it declines. Mumbai records the highest average rent per sqft overall. In Chennai and Kolkata, a decrease in rent per sqft is observed for 5 BHK homes. Notably, in Chennai, the largest sized properties have the lowest rent per sqft, suggesting that these homes are located in more affordable areas.





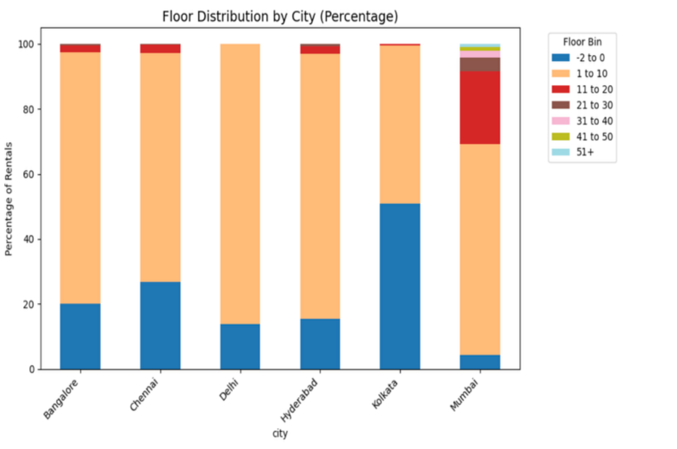
The majority of properties in each city are 2 bhks

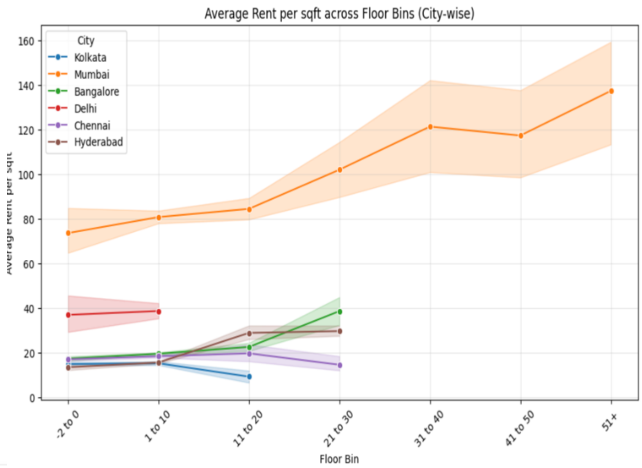


-Trend related to floors in cities:

In Kolkata, approximately half of the properties are located either in basements or on the ground floor, whereas Mumbai's dataset includes a wider range of floor levels with a lot more higher floors.

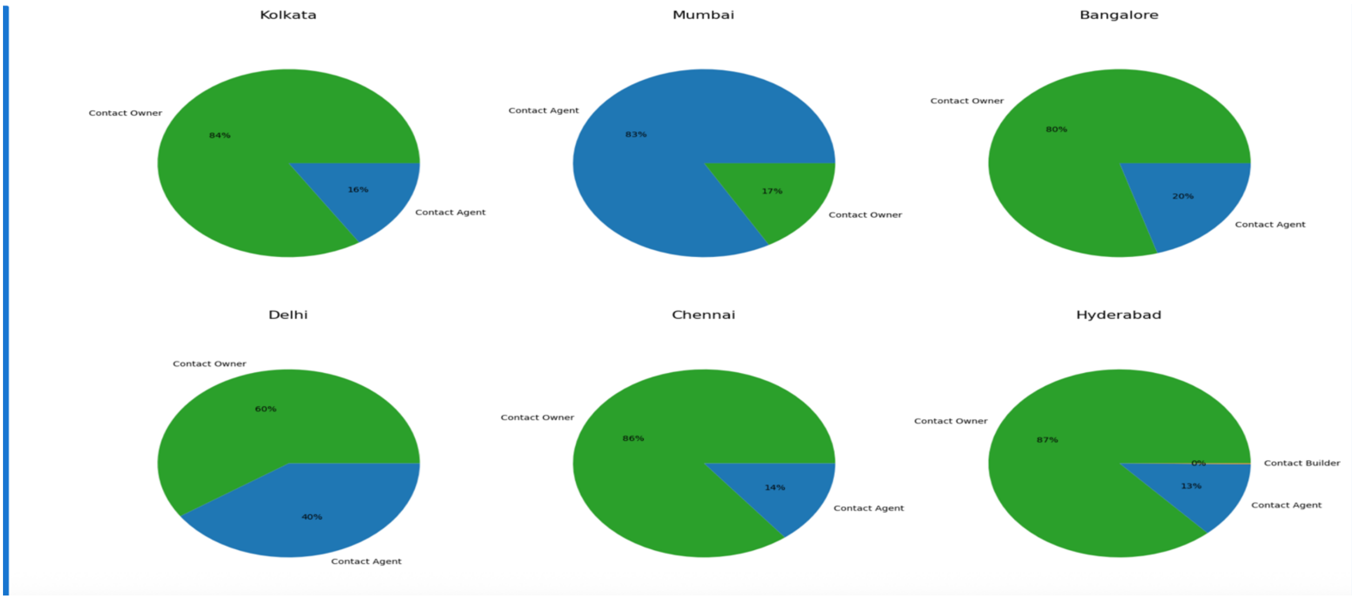
Rent per sqft exhibits an upward trend across most cities, except Kolkata, where the presence of properties above the 10th floor is minimal.





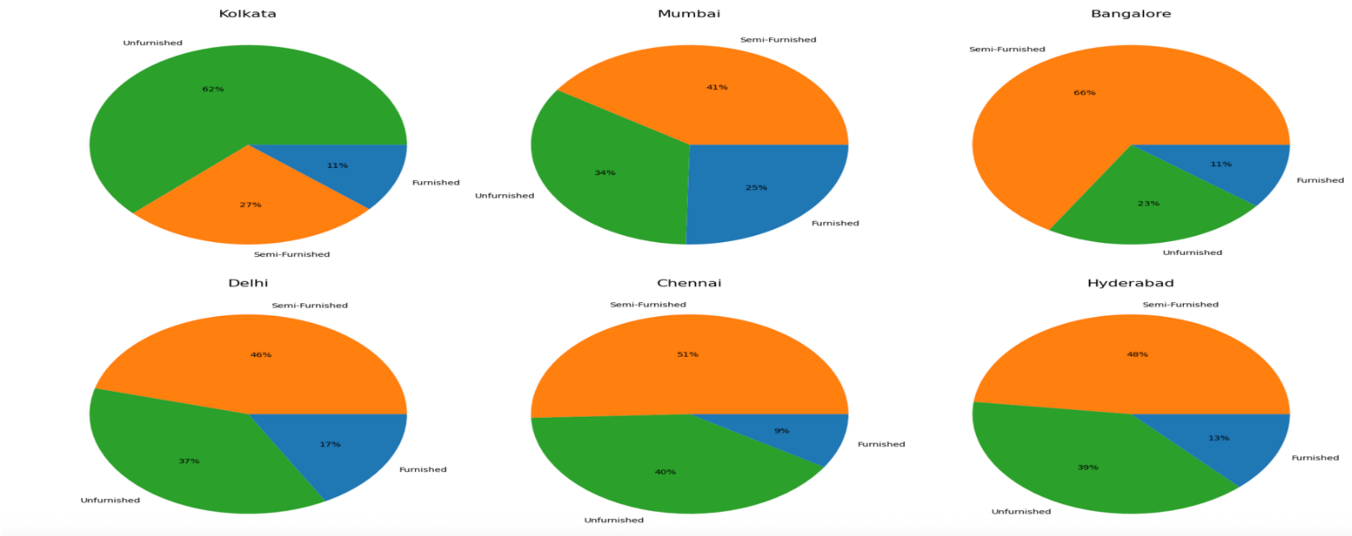
Trends related to point of contact :

In Mumbai, property rentals are primarily facilitated through agents, while in all other cities, homeowners themselves serve as the main point of contact.



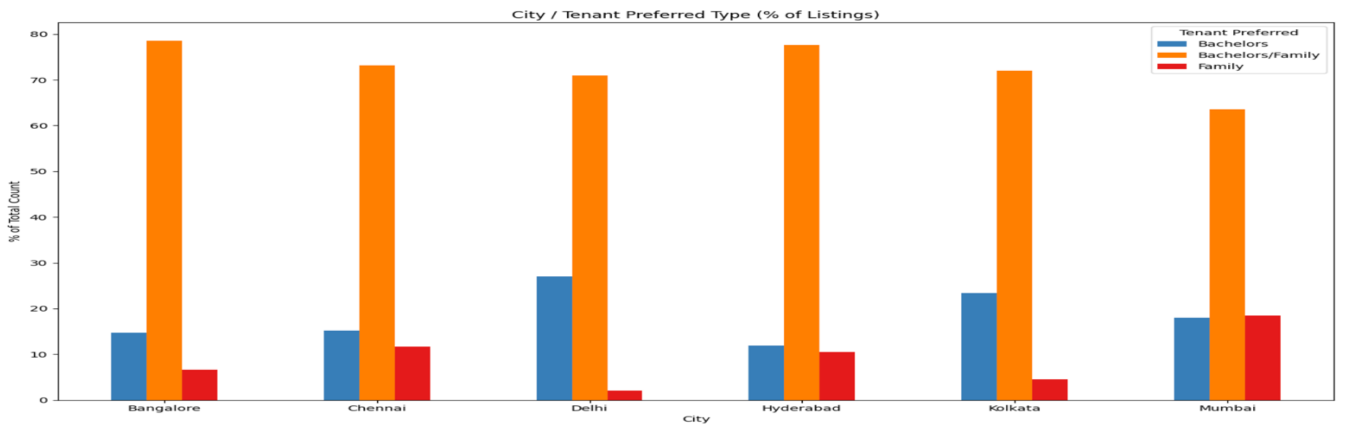
-Trends about Furnishing status:

Mumbai has nearly equal distribution across furnishing statuses, while Kolkata predominantly has unfurnished houses. In contrast, the other cities mainly consist of semi-furnished properties.



-Trends on preferred tenants:

Delhi shows the lowest preference for family tenants, whereas Mumbai is unique in exhibiting an almost equal preference between families and bachelors.



The OLS Regression model explains approximately 79% of the variation in rent per sqft, indicating a strong fit. Tenant preference, however, does not show consistent statistical significance in influencing rent. In contrast, variables such as city, furnishing status, affordability category, property size, number of bathrooms, and floor level are all significant predictors with p-values below 0.05. Additionally, area type (Carpet Area or Super Area) is not statistically significant, indicating it does not have a meaningful impact on rent per sqft.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          rent_per_sqft   R-squared:                       0.791
Model:                            OLS   Adj. R-squared:                  0.790
Method:                 Least Squares   F-statistic:                     1040.
Date:                Wed, 20 Aug 2025   Prob (F-statistic):               0.00
Time:                        01:17:09   Log-Likelihood:                -18427.
No. Observations:                4423   AIC:                         3.689e+04
Df Residuals:                    4406   BIC:                         3.700e+04
Df Model:                          16
Covariance Type:            nonrobust
=============================================================================================================
                                                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------------------------
Intercept                                       1.9123     11.124      0.172      0.864     -19.896      23.721
C(tenant_preferred)[T.Bachelors/Family]         1.5657      0.681      2.299      0.022       0.231       2.901
C(tenant_preferred)[T.Family]                  -0.3505      0.963     -0.364      0.716      -2.238       1.537
C(city)[T.Chennai]                             -1.2151      0.774     -1.569      0.117      -2.733       0.303
C(city)[T.Delhi]                               23.1326      0.892     25.939      0.000      21.384      24.881
C(city)[T.Hyderabad]                           -2.5064      0.792     -3.165      0.002      -4.059      -0.954
C(city)[T.Kolkata]                             -2.4369      0.916     -2.659      0.008      -4.234      -0.640
C(city)[T.Mumbai]                              60.3636      0.917     65.805      0.000      58.565      62.162
C(furnishing_status)[T.Semi-Furnished]         -3.6357      0.722     -5.036      0.000      -5.051      -2.220
C(furnishing_status)[T.Unfurnished]            -4.0506      0.757     -5.349      0.000      -5.535      -2.566
C(area_type)[T.Carpet Area]                     0.1288     11.076      0.012      0.991     -21.586      21.843
C(area_type)[T.Super Area]                      1.9994     11.067      0.181      0.857     -19.698      23.697
C(affordability)[T.Mid-range]                  10.3347      0.585     17.661      0.000       9.188      11.482
C(affordability)[T.Premium]                    38.4159      0.736     52.200      0.000      36.973      39.859
size                                           -0.0032      0.001     -4.730      0.000      -0.004      -0.002
bathroom                                        3.2344      0.473      6.838      0.000       2.307       4.162
mapped_floor                                    0.3626      0.049      7.369      0.000       0.266       0.459
==============================================================================
Omnibus:                     2557.929   Durbin-Watson:                   1.951
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            36417.132
Skew:                           2.480   Prob(JB):                         0.00
Kurtosis:                      16.153   Cond. No.                     9.41e+04
==============================================================================
```

Model Training:

Columns that do not significantly influence rent were dropped from the dataset. The remaining data was then split into training and testing sets with an 80:20 ratio to build and evaluate the predictive model.

```
rent_ds_model = rent_ds_new.drop(columns={'tenant_preferred' ,'posted_on','area_type','point_of_contact','rent_per_sqft','size_bin','floor_bi

# One-hot encode categorical columns
categorical_cols = ['city', 'furnishing_status', 'area_locality', 'affordability']
rent_ds_model = pd.get_dummies(rent_ds_model, columns=categorical_cols, drop_first=True)

# Features and target
X = rent_ds_model.drop(columns=['rent'])
y = rent_ds_model['rent']

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

-XGB boost Model:

```
XGBoost Performance on Test Set:
RMSE: 12437.89
MAE: 5611.94
R²: 0.938
```

After evaluating the initial model performance, I applied RandomizedSearchCV to optimize the hyperparameters for improved accuracy. The best parameters obtained through this tuning process are as follows:

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
Best parameters: {'subsample': 0.8, 'n_estimators': 500, 'max_depth': 5, 'learning_rate': 0.1}
```

```
Tuned XGBoost Performance on Test Set:
RMSE: 12154.82
MAE: 5471.68
R²: 0.940
```

-RandomForestRegressor:

```
Random Forest Performance on Test Set:
RMSE: 12891.08
MAE: 5553.51
R²: 0.933
```

:

Both XGBoost and Random Forest models provide strong predictive performance. XGBoost achieves slightly lower error (RMSE and MAE) and a higher R^2 score (0.94) compared to Random Forest (R^2: 0.93), making it the more accurate model for predicting rent

	model	rmse	mae	r2
0	XGBoost	12154.818633	5471.679688	0.940405
1	RandomForest	12891.083100	5553.509005	0.932966

.

Comparison of Models:

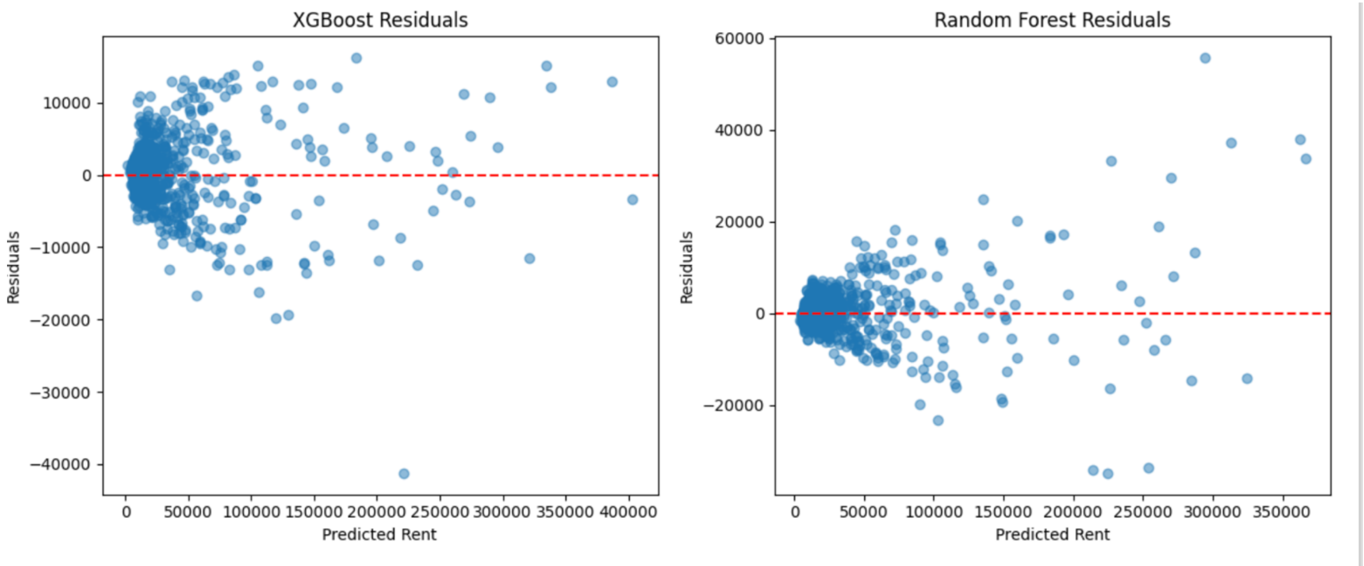
-K-fold Cross Validation:

k-fold cross-validation was implemented to rigorously assess model robustness and generalization capability. the XGBoost model consistently outperformed the Random Forest model, demonstrating lower prediction errors and higher R^2 scores across all folds.

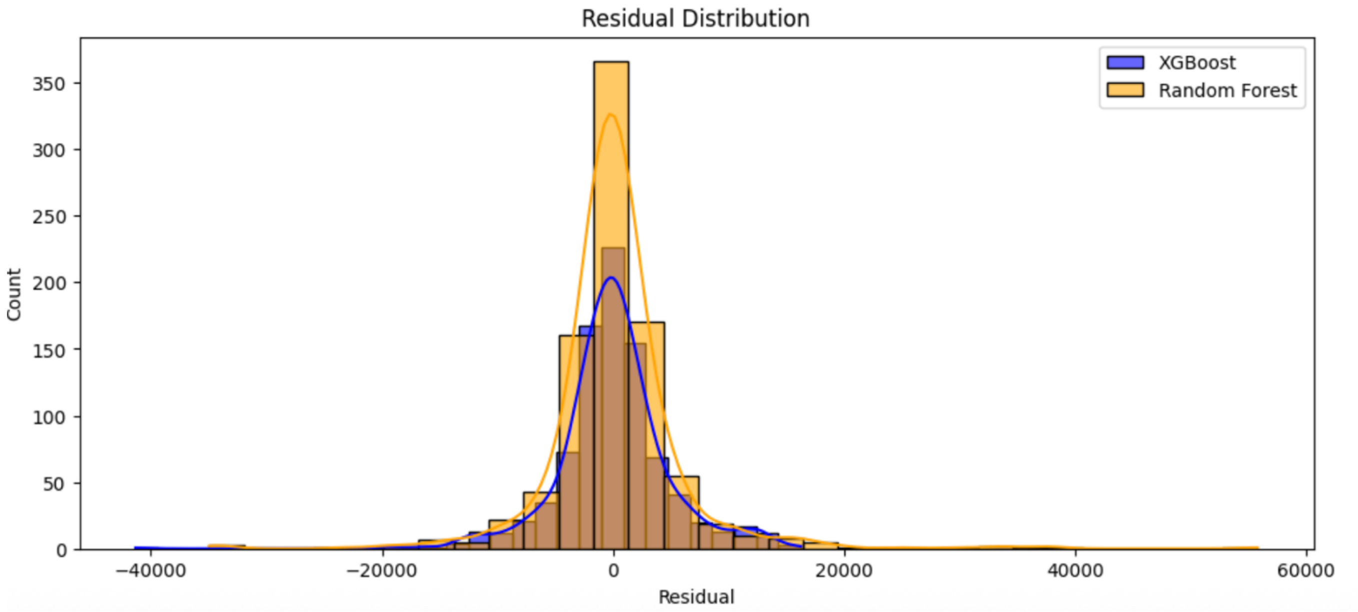
	Metric	XGBoost	Random Forest
0	RMSE Mean	14508.767840	15238.387782
1	RMSE Std	2376.904154	2619.149544
2	MAE Mean	5653.932227	5782.982600
3	MAE Std	342.805012	356.264127
4	R2 Mean	0.919416	0.911182
5	R2 Std	0.013398	0.015577

-Residuals Analysis:

Both models performed well for average rents, but error variance increased with rent value, revealing difficulty predicting high-end properties. The residuals for XGBoost are more symmetrically and tightly distributed around zero, indicating good fit and reliability. In contrast, Random Forest has larger residuals and a higher frequency of extreme errors for expensive properties, suggesting less stability in these cases.



The residual distribution analysis demonstrates that both models typically predict rent values with low error, as most residuals are close to zero and the distributions are nearly symmetric. XGBoost exhibits a more pronounced central peak and shorter tails, confirming its stronger and more consistent prediction accuracy, while Random Forest shows increased variability with a higher frequency of extreme errors. These findings validate the selection of XGBoost as the better model both in average performance and error consistency.



Conclusion:

This study provides a comprehensive analysis of housing rent trends across multiple cities, highlighting significant variations in rent per sqft relative to property size, BHK count, and location. The dynamic outlier detection method improved data quality, ensuring reliable results. Machine learning models, particularly XGBoost, demonstrated strong predictive performance for rent estimates, supported by robust evaluation including k-fold cross-validation.

Limitations:

-Several categories had minimal data points, limiting the robustness of insights for those segments and potentially reducing model accuracy in those cases.

-Incorporating additional variables such as property age, proximity to amenities, or market conditions could enhance predictive accuracy.