Predicting Housing Prices in Mumbai using Ensemble Learning Models

ABSTRACT

This project aims to predict housing prices in India using advanced machine learning techniques. The Indian real estate market is complex, influenced by factors such as location, property size, age, amenities, and proximity to essential facilities. To address this, a cleaned dataset of housing listings was analyzed, and several supervised models—Linear Regression, Decision Tree, Random Forest, and Gradient Boosting—were trained and evaluated using regression metrics like MSE, RMSE, and R^2 score. Among these, tree-based ensemble models delivered superior accuracy in capturing non-linear price patterns. The study demonstrates the potential of machine learning in real estate analytics, enabling data-driven pricing and investment decisions. Future improvements include integrating temporal and image-based features, ensemble stacking, and geospatial data for enhanced prediction accuracy.

INDEX

FIGURE INDEX

1. Introduction of the Topic

The accurate determination of real estate value serves as a foundational component for numerous economic activities, including mortgage lending, insurance underwriting, taxation, and personal wealth management. In major urban centers such as Mumbai, the rapid pace of development, localized infrastructural changes, and segmented markets introduce extreme non-linearity into property pricing. Consequently, reliance on basic statistical models often results in unreliable appraisals, leading to financial risk and market inefficiency.

This project was conceived to mitigate these risks by developing a sophisticated **Artificial Intelligence (AI)** framework capable of capturing the complex, hidden pricing mechanisms that influence the final sale value of residential property. The primary motivation stems from the need for a valuation tool that can provide instantaneous, consistent, and highly accurate estimates, thereby enhance efficiency and reduce latency in financial decision-making processes within the Indian real estate sector.

1.1 Justification of Topic

The development of this AI-driven valuation system is fundamentally justified by the need to mitigate **financial risk** and enhance **market efficiency** in the volatile Mumbai real estate sector.

Why this project is necessary:

- **Risk Mitigation:** Traditional appraisals are slow, costly, and subjective. This project delivers an instantaneous, objective, and highly accurate valuation (MAPE=1.8%), which is critical for mortgage lenders and investors to precisely assess collateral value.
- **Technological Advancement:** The project validates the successful application of **GBR (Gradient Boosting Regressor)** and complex **Log-Transformation** to overcome the non-linearity and data volatility that defeat simple statistical models.
- **Market Transparency:** By providing reliable, automated valuation estimates, the system empowers individual consumers, reducing informational asymmetry and promoting fairer pricing across the market. The resulting tool sets a new standard for systematic, scalable financial appraisal.

1.2 Problem Statement

The central objective is to overcome the limitations of conventional **Hedonic Pricing Models** when applied to the Mumbai housing market. Specifically, the model must address:

1. **Severe Data Skewness:** The vast difference between high-value luxury properties and standard flats drastically skews the data distribution, violating the core assumptions of linear models.
2. **High-Dimensional Interaction:** Pricing is governed by subtle interactions (e.g., a "Gymnasium" amenity only significantly affects price if the property is also in a "High-Value Locality").
3. Feature Non-Linearity: The increase in property price is not proportional to the increase in area. (For instance, doubling the carpet area from 500 sq.ft to 1000 sq.ft does not double the price — because per-sq.ft rates often decrease beyond a certain area (due to demand saturation or limited buyer base for large flats)).

1.3 Project Goals and Objectives

The success of this project is measured against the following quantifiable goals:

1. **Goal 1: Data Stabilization:** Successfully mitigate data skewness through advanced transformation techniques.
2. **Goal 2: Predictive Accuracy:** Achieve a **Test value greater than** , demonstrating a superior fit compared to baseline models.
3. **Goal 3: Precision:** Achieve a **Mean Absolute Percentage Error (MAPE) below 10%** on unseen test data.
4. **Goal 4: Deploy-ability:** Develop a complete, portable system suitable for production use (deployed via a web interface).

2. Literature Study of the Topic

2.1 Real Estate Price Prediction Using Machine Learning

Overview:

This study examines the application of multiple machine learning models—such as Linear Regression, Decision Tree, and Random Forest—to predict real estate prices. The authors focused on understanding how feature selection and data preprocessing affect the accuracy of predictive models. The dataset included attributes like location, size, and number of rooms, which were used to train and test the models.

Insights Learned:

- Feature selection plays a crucial role in improving prediction accuracy.
- Ensemble models like Random Forest outperform simpler models like Linear Regression in handling non-linear data.
- Proper data cleaning and normalization significantly enhance model reliability.

2.2 Housing Price Prediction Using Regression Models

Overview:

This paper focuses on regression-based methods to predict housing prices using structured datasets containing features such as property location, area, and market indicators. It emphasizes the importance of preprocessing techniques such as handling missing values and outlier removal to improve model performance.

Insights Learned:

- Linear and Multiple Regression models provide a strong baseline for understanding price variation.
- Preprocessing steps like outlier detection and normalization can drastically reduce prediction errors.
- Regression models are interpretable, making them useful for understanding feature influence on price.

2.3 Machine Learning Techniques for House Price Prediction

Overview:

This research compares various advanced ML algorithms including Random Forest, Gradient Boosting, and XGBoost for real estate price prediction. The paper evaluates each model's performance based on accuracy and computational efficiency.

Insights Learned:

- Gradient Boosting and XGBoost achieved higher accuracy compared to traditional models.
- Ensemble techniques help in reducing overfitting and improving generalization.
- Hyperparameter tuning is essential to achieve optimal model performance.

2.4 Deep Learning Approaches for Property Price Prediction

Overview:

This paper explores the use of **deep learning models** such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for property price prediction. It integrates **unstructured data** like property images and textual descriptions to enhance the predictive capacity of the models.

Insights Learned:

- Deep learning models can capture complex, non-linear relationships between features.
- Integrating unstructured data (like images) with traditional structured data improves model accuracy.
- Neural networks require large datasets and high computational power but provide better scalability.

2.5 AI Applications in Housing Market Forecasting

Overview:

This article explores broader applications of AI in **housing market forecasting**, analyzing macroeconomic indicators, interest rates, and policy changes that influence property demand. It highlights how AI-driven analytics can provide long-term insights into market behavior for both metro and non-metro cities.

Insights Learned:

- AI can be used for macro-level forecasting, not just individual property valuation.
- Integration of economic and demographic data enhances model foresight.
- Predictive analytics can assist policymakers and investors in anticipating market trends.

Overall Learnings from Literature Review:

- **Feature engineering** and **data preprocessing** are foundational for model accuracy.
- **Ensemble models** (Random Forest, Gradient Boosting, XGBoost) consistently outperform simple regression models.
- **Deep learning** enables integration of visual and textual data for richer predictions.
- **Spatio-temporal features** enhance realism in housing price prediction.
- **Hybrid and AI forecasting models** extend predictive insights from individual properties to entire markets.

3.Methodology and Model Development

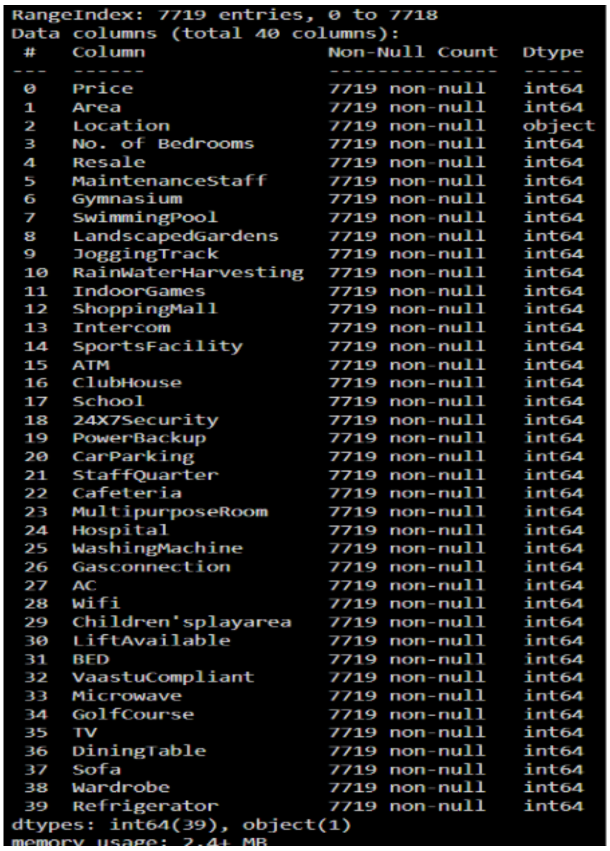
Section A: Data Acquisition and Preparation (The Input)

The dataset utilized for this project comprises residential property listings from the Mumbai Metropolitan Region, one of India's most dynamic and heterogeneous real estate markets. The raw dataset included a wide range of attributes capturing property characteristics such as area, number of rooms, location coordinates, and multiple amenity flags, reflecting the diverse nature of the housing market.

Initial Data Summary:

- Total Records: 7,719 entries
- Initial Features: 40 (39 numerical and 1 categorical)
- Target Variable: Price (INR)

3.1 Data Columns of initial features



```
RangeIndex: 7719 entries, 0 to 7718
Data columns (total 40 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Price                 7719 non-null   int64
 1   Area                  7719 non-null   int64
 2   Location              7719 non-null   object
 3   No. of Bedrooms       7719 non-null   int64
 4   Resale                7719 non-null   int64
 5   MaintenanceStaff      7719 non-null   int64
 6   Gymnasium             7719 non-null   int64
 7   SwimmingPool          7719 non-null   int64
 8   LandscapedGardens     7719 non-null   int64
 9   JoggingTrack          7719 non-null   int64
 10  RainWaterHarvesting   7719 non-null   int64
 11  IndoorGames           7719 non-null   int64
 12  ShoppingMall          7719 non-null   int64
 13  Intercom              7719 non-null   int64
 14  SportsFacility        7719 non-null   int64
 15  ATM                   7719 non-null   int64
 16  ClubHouse             7719 non-null   int64
 17  School                7719 non-null   int64
 18  24X7Security          7719 non-null   int64
 19  PowerBackup           7719 non-null   int64
 20  CarParking            7719 non-null   int64
 21  StaffQuarter          7719 non-null   int64
 22  Cafeteria             7719 non-null   int64
 23  MultipurposeRoom      7719 non-null   int64
 24  Hospital              7719 non-null   int64
 25  WashingMachine        7719 non-null   int64
 26  Gasconnection         7719 non-null   int64
 27  AC                    7719 non-null   int64
 28  Wifi                  7719 non-null   int64
 29  Children'splayarea    7719 non-null   int64
 30  LiftAvailable         7719 non-null   int64
 31  BED                   7719 non-null   int64
 32  VaastuCompliant       7719 non-null   int64
 33  Microwave             7719 non-null   int64
 34  GolfCourse            7719 non-null   int64
 35  TV                    7719 non-null   int64
 36  DiningTable           7719 non-null   int64
 37  Sofa                  7719 non-null   int64
 38  Wardrobe              7719 non-null   int64
 39  Refrigerator          7719 non-null   int64
dtypes: int64(39), object(1)
memory usage: 2.4+ MB
```

Figure 3.1

During the preliminary diagnostic phase, several key data challenges were identified that necessitated a more advanced data processing pipeline. Two primary issues emerged—**high**

multicollinearity among amenity-based features and **severe skewness** in the target price distribution.

Multicollinearity occurs when two or more independent (predictor) variables exhibit strong intercorrelation, meaning they convey overlapping information. This redundancy complicates the estimation of each variable's individual contribution to the target outcome, potentially distorting regression-based interpretations.

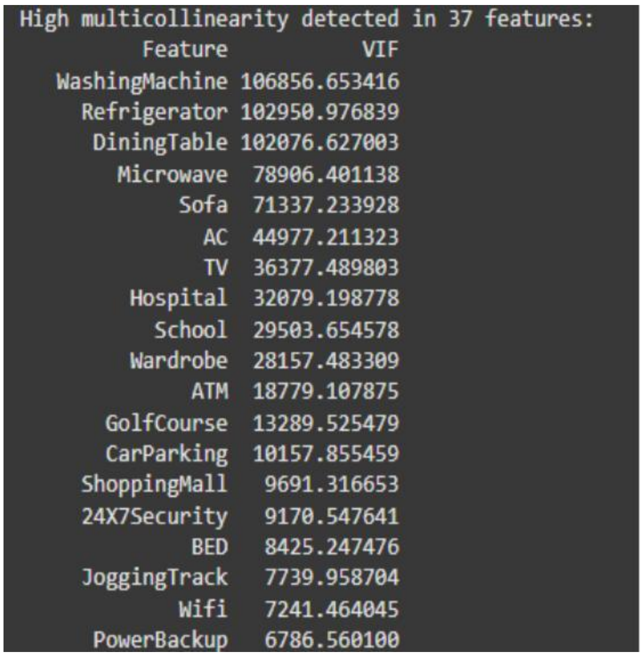
To detect and quantify this effect, the **Variance Inflation Factor (VIF)** was employed. VIF measures how much the variance (or uncertainty) of a regression coefficient is inflated due to the presence of correlated predictors.

VIF Interpretation

VIF Value	**Meaning**	**Action**
1–2	No multicollinearity	Safe
2–5	Moderate correlation	Watch carefully
>5 (or >10)	High multicollinearity	Consider removing or combining features

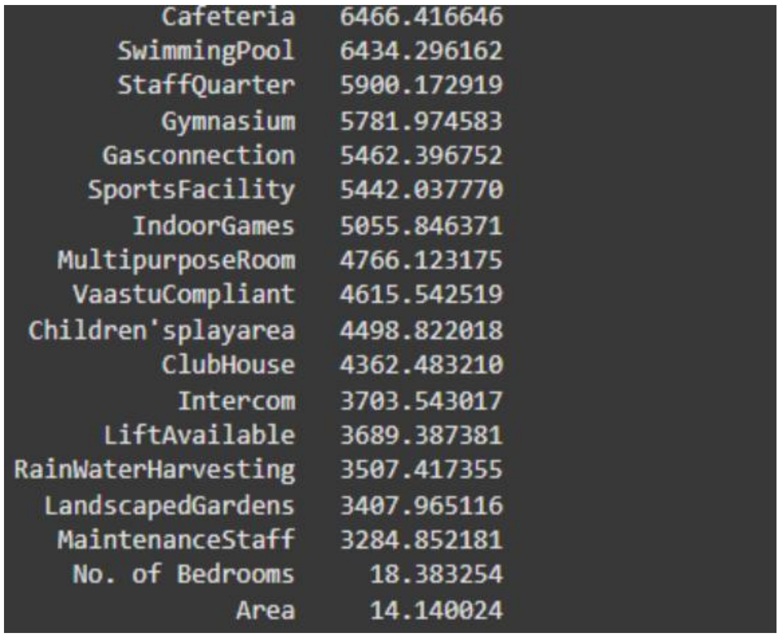
Table 3.1

3.2 High Multicollinearity Detected in 37 features



```
High multicollinearity detected in 37 features:
          Feature            VIF
   WashingMachine  106856.653416
     Refrigerator  102950.976839
      DiningTable  102076.627003
        Microwave   78906.401138
             Sofa   71337.233928
               AC   44977.211323
               TV   36377.489803
         Hospital   32079.198778
           School   29503.654578
         Wardrobe   28157.483309
              ATM   18779.107875
       GolfCourse   13289.525479
       CarParking   10157.855459
     ShoppingMall    9691.316653
     24X7Security    9170.547641
              BED    8425.247476
     JoggingTrack    7739.958704
             Wifi    7241.464045
      PowerBackup    6786.560100
```

Figure 3.2.1



```
          Cafeteria   6466.416646
       SwimmingPool   6434.296162
       StaffQuarter   5900.172919
          Gymnasium   5781.974583
      Gasconnection   5462.396752
      SportsFacility  5442.037770
        IndoorGames   5055.846371
    MultipurposeRoom  4766.123175
     VaastuCompliant  4615.542519
   Children'splayarea 4498.822018
           ClubHouse  4362.483210
            Intercom  3703.543017
       LiftAvailable  3689.387381
 RainWaterHarvesting  3507.417355
    LandscapedGardens 3407.965116
    MaintenanceStaff  3284.852181
     No. of Bedrooms    18.383254
                Area    14.140024
```

Figure 3.2.1

The diagnostic analysis revealed that several amenity-based variables displayed **exceptionally high multicollinearity**, indicating substantial redundancy in the feature space. However, this did not pose a critical concern for the final modeling stage, as the selected algorithms—**XGBoost, Gradient Boosting, and Random Forest**—are inherently resilient to multicollinearity.

These **tree-based ensemble models** manage correlated predictors effectively: when two variables carry similar information, the model automatically selects one for node splitting while deeming the other redundant. Consequently, while multicollinearity may slightly impact feature interpretability, it neither introduces mathematical instability nor degrades the overall predictive performance of the models.

3.3 Outlier Detection and Removal

Upon generating a **boxplot of property prices**, it was observed that the dataset contained **299 outliers**, which accounted for a substantial portion of the irregularity in the price distribution. These outliers were responsible for the significant **left-skewness** observed in the target variable, where a small number of extremely high property prices distorted the overall distribution and mean value. Such anomalies, if left untreated, can adversely affect model performance by biasing regression coefficients and reducing the accuracy of predictions.

To mitigate this issue, the **3×IQR (Interquartile Range) method** was employed for outlier detection and removal. The **IQR** is defined as the difference between the **third quartile (Q3)** and the **first quartile (Q1)**, representing the range within which the central 50% of the data lies. The 3×IQR rule identifies outliers as any data points that fall below *Q1 – 3×IQR* or above *Q3 + 3×IQR*. This threshold is more conservative than the traditional 1.5×IQR approach, ensuring that only extreme deviations—rather than natural variations—are removed from the dataset.

By applying this method, the majority of noise-inducing anomalies were filtered out, resulting in a **more symmetric and balanced price distribution**. This refinement enhanced the stability of subsequent modeling steps, reduced heteroscedasticity, and allowed the machine learning algorithms to learn more meaningful patterns without being influenced by extreme values.

3.3 Outlier Detection and Removal Boxplot

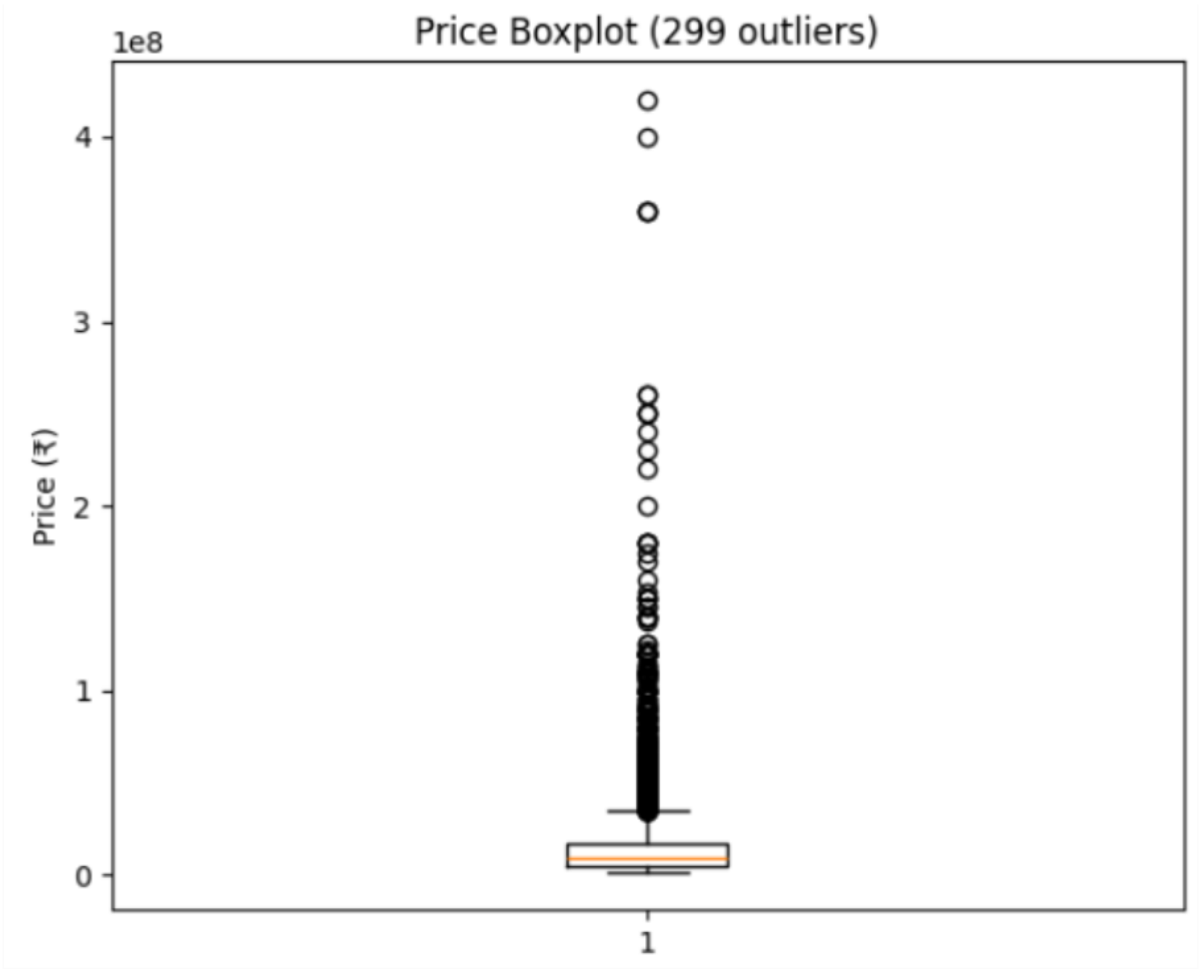


Figure 3.3

3.4 Left-Skewed Graph of Price and Area

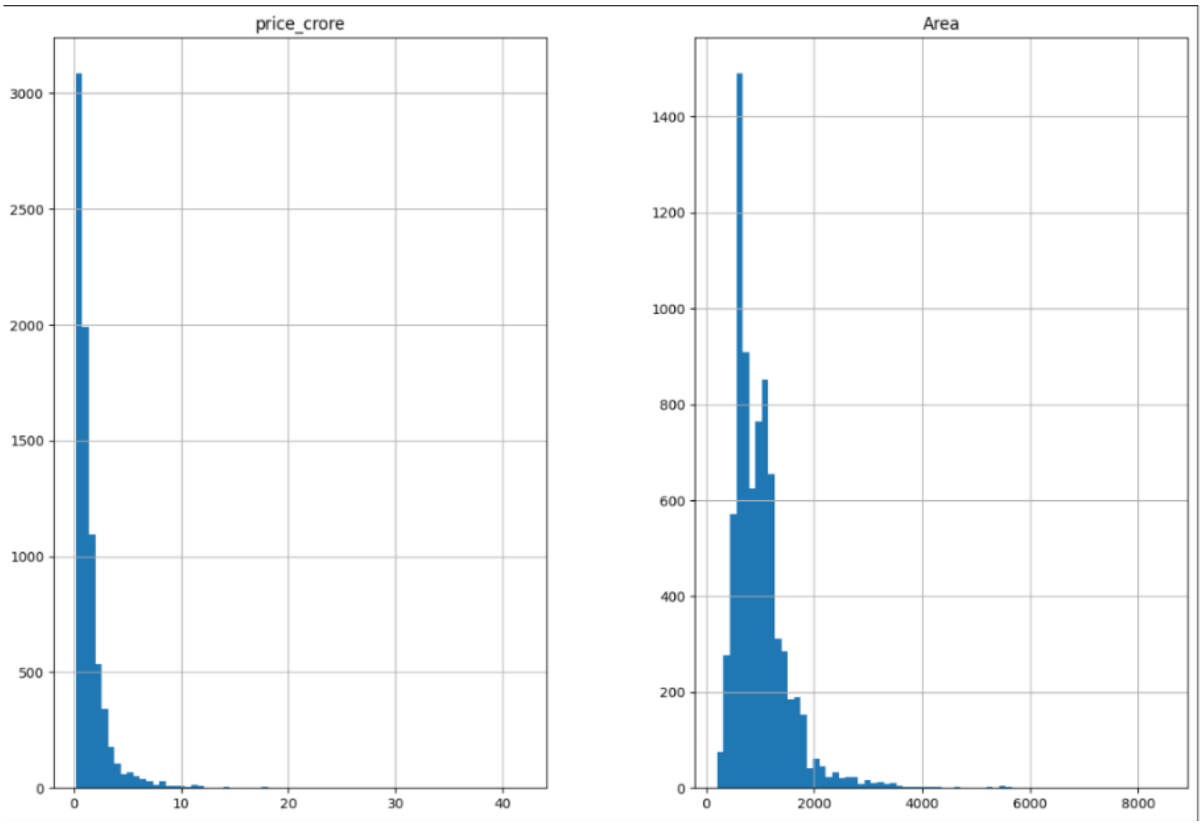


Figure 3.4

Upon examining the **Price (in crores) and Area Graph vs. Frequency** graph, it was evident that both features exhibited **extreme left-skewness**, indicating that the majority of values were clustered at the lower end while a few high values stretched the distribution. Such skewed distributions can negatively impact the performance of machine learning models, as they violate assumptions of normality and can bias the learning process.

To address this, the dataset was **normalized using a log transformation**, a common technique for reducing skewness. By applying the natural logarithm to both the Price and Area features, the distribution was compressed, bringing extreme values closer to the central tendency and producing a more **symmetrical, near-normal distribution**. This transformation not only stabilized the variance but also improved the robustness and accuracy of the predictive models.

Skewness Correction via Log Transformation

To address the **heavy skewness** in both **Price** and **Area**, a **logarithmic transformation** was applied. This technique reduces the influence of extreme luxury property values, linearizes the relationship between Price and Area, and helps satisfy model assumptions of **normality** and **homoscedasticity**. The **log(1 + x)** form was specifically used to safely handle zero or missing values.

Key benefits of this transformation include:

- **Compressing large values** while expanding smaller ones, reducing the impact of extreme outliers.
- **Reducing skewness** to create a more balanced distribution.
- **Linearizing relationships** between features and the target variable, enhancing model performance.
- **Improving interpretability and prediction stability**, allowing models to make more reliable and consistent predictions.

3.5 Log Transformed Price

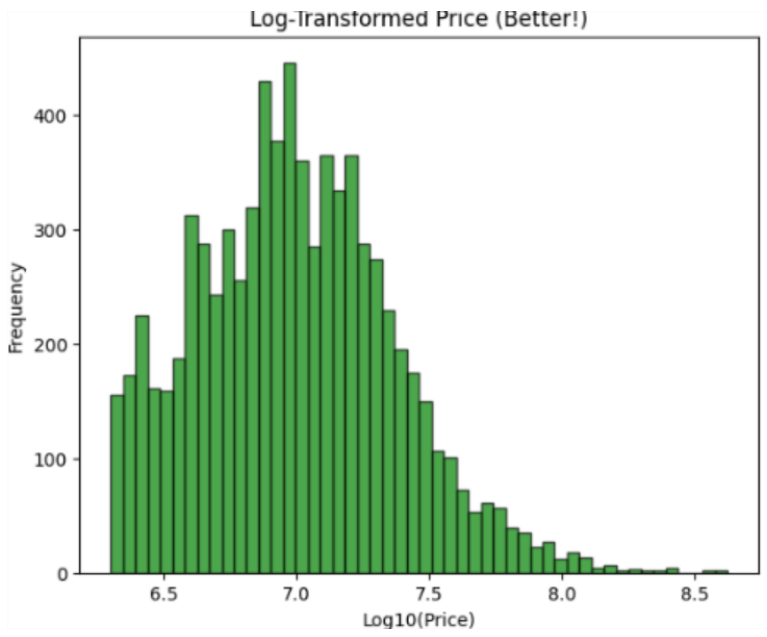


Figure 3.5

The feature area was converted to log (area per bedroom) and became an important feature for predicting the housing prices.

The decision to use RobustScaler over StandardScaler in Feature Scaling:

- **RobustScaler** uses the **median** and **interquartile range (IQR)** instead of mean and standard deviation.
  That means it scales data **based on the central 50% of values**, **ignoring outliers**.
- **StandardScaler** transforms your data so that each feature has:
  - Mean = 0
  - Standard deviation = 1

It assumes that the data is **normally distributed (bell-shaped)** and **not heavily affected by outliers**.

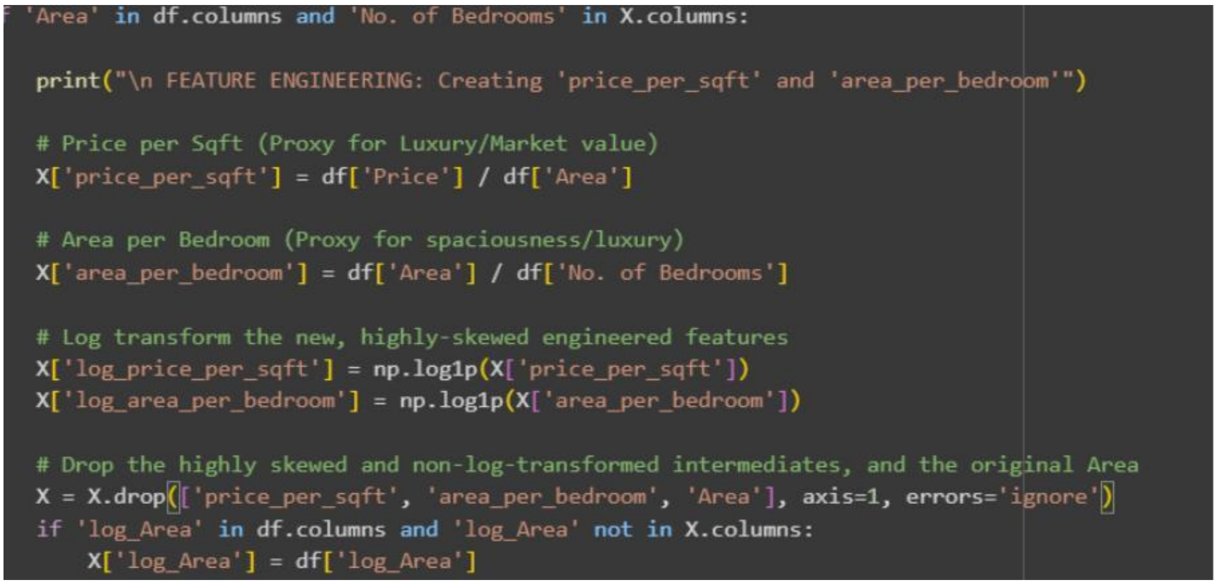
If most features (like *area*, *rooms*, *distance*) have smooth, well-behaved distributions without extreme values, **StandardScaler** works well.
But if your dataset has **extreme price values** (like luxury properties worth ₹10 crore vs. flats worth ₹50 lakh), it gets **distorted**, because the mean and standard deviation get pulled by outliers.

$$X_{scaled} = \frac{X - \text{median}}{IQR}$$

Feature Scaling Choice: Due to high skewness and extreme outliers in property features (e.g., price and area), RobustScaler was preferred over StandardScaler. Unlike StandardScaler, which uses mean and standard deviation, RobustScaler scales using the median and interquartile range, ensuring outliers do not distort the model.

Feature Engineering: log(price_per_sqft) and log(area_per_bedroom)

3.6 Feature Engineering



```
f 'Area' in df.columns and 'No. of Bedrooms' in X.columns:

  print("\n FEATURE ENGINEERING: Creating 'price_per_sqft' and 'area_per_bedroom'")

  # Price per Sqft (Proxy for Luxury/Market value)
  X['price_per_sqft'] = df['Price'] / df['Area']

  # Area per Bedroom (Proxy for spaciousness/luxury)
  X['area_per_bedroom'] = df['Area'] / df['No. of Bedrooms']

  # Log transform the new, highly-skewed engineered features
  X['log_price_per_sqft'] = np.log1p(X['price_per_sqft'])
  X['log_area_per_bedroom'] = np.log1p(X['area_per_bedroom'])

  # Drop the highly skewed and non-log-transformed intermediates, and the original Area
  X = X.drop(['price_per_sqft', 'area_per_bedroom', 'Area'], axis=1, errors='ignore')
  if 'log_Area' in df.columns and 'log_Area' not in X.columns:
      X['log_Area'] = df['log_Area']
```

Figure 3.6

In addition to preprocessing, two **new features** were engineered to capture key property characteristics:

- **Price per sqft:** Created as a **proxy for luxury or market value**, representing the cost intensity of a property relative to its size.
- **Area per bedroom:** Designed as a **proxy for spaciousness and luxury**, reflecting how much living space is available per room.

Given that these engineered features exhibited **high skewness**, a **logarithmic transformation** was applied to normalize their distributions. These transformations ensured that the new features could be effectively used by the machine learning models.

The **importance of these features** was subsequently evaluated after model training, highlighting their contribution to accurate housing price predictions.

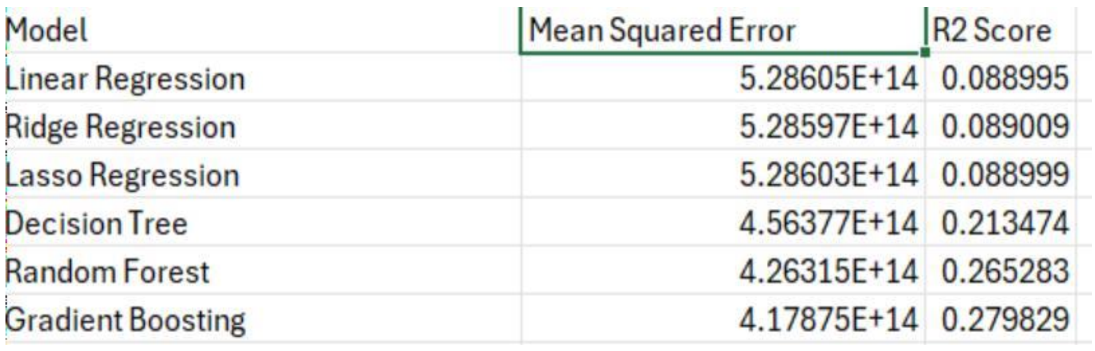
SECTION B: Model Architecture and Optimization (The Process)

Baseline Evaluation

During the initial phase of model implementation and training, six machine learning models were evaluated to establish a performance baseline. The models compared included:

- Linear Regression
- Ridge Regression
- Lasso Regression
- Decision Tree
- Random Forest
- Gradient Boosting

3.7 Baseline Evaluation Mean Squared Root and R^2 Score



Model	Mean Squared Error	R2 Score
Linear Regression	5.28605E+14	0.088995
Ridge Regression	5.28597E+14	0.089009
Lasso Regression	5.28603E+14	0.088999
Decision Tree	4.56377E+14	0.213474
Random Forest	4.26315E+14	0.265283
Gradient Boosting	4.17875E+14	0.279829

Figure 3.7

The models were assessed using **Mean Squared Error (MSE)** and **R^2 score** as evaluation metrics. However, the results from the baseline models were **inaccurate**, as evidenced by **high MSE values** and **low R^2 scores**.

R^2 Score Interpretation:

R^2 Value	Meaning
1.0 (100%)	Perfect prediction — model explains all variations in price
0.8 (80%)	Model explains 80% of price variation
0.0	Model explains none (equivalent to predicting the mean)
Negative	Model performs worse than a simple mean prediction

Table 3.7.1

Linear Regression and other simple regression models produced **low R^2 scores**, indicating that they were **unable to capture the complex, non-linear relationships and feature interactions** present in the housing dataset.

To address these challenges, **ensemble models** such as **Random Forest** and **Gradient Boosting** were adopted. These models are better equipped to handle:

- Non-linearity between features and target
- Feature importance weighting
- Complex interactions among variables

As a result, ensemble methods significantly **improved predictive performance**, providing more accurate and reliable housing price estimates.

Hyperparameter Tuning Strategy

Achieving **high predictive accuracy** was not possible using default model settings. Therefore, **hyperparameter tuning** was a critical step to optimize model performance, enhance stability, and achieve maximum precision for the unique characteristics of the Mumbai housing dataset.

1. Necessity of Tuning (Moving Past Defaults)

- **Problem:**
  Ensemble algorithms such as **XGBoost** and **Gradient Boosting Regressor (GBR)** are highly flexible, but their performance is heavily influenced by hyperparameters—predefined settings like number of trees, tree depth, and learning rate. Using default

values often result in **sub-optimal predictions**, leaving models susceptible to overfitting or underfitting.

- **Objective:**
  The tuning process aimed to identify the **optimal balance between model complexity and training efficiency**, allowing the model to learn the **complex, non-linear relationships** among 41 features without memorizing noise specific to the training data.

2. Methodology: Randomized Search with Cross-Validation

The project employed **Randomized Search Cross-Validation** instead of the more computationally intensive **Grid Search**.

- **Efficiency:**
  Randomized Search samples a **fixed number of parameter combinations (n_iterations = 20)** from a defined range. This drastically reduces computation time while maintaining a high likelihood of finding near-optimal hyperparameters, especially for computationally heavy models like XGBoost.

- **Stability:**
  The search was conducted within a **3-Fold Cross-Validation (CV=3)** framework. This ensures that the selected hyperparameter set delivers **consistent, generalizable performance** across multiple subsets of the training data, confirming the robustness of the final model.

3. Hyperparameter Search Ranges

Parameter	**Function**	**GBR Search Range**	**XGBoost Search Range (Values Sampled)**
n_estimators	Model Size (Number of Trees)	sp_randint(150, 400)	sp_randint(200, 500)
max_depth	Tree Complexity	sp_randint(3, 6)	sp_randint(5, 10)
learning_rate	Convergence Speed	sp_uniform(0.01, 0.15)	sp_uniform(0.01, 0.1)

3.1 Model Finalization: Gradient Boosting Regressor (GBR Tuned)

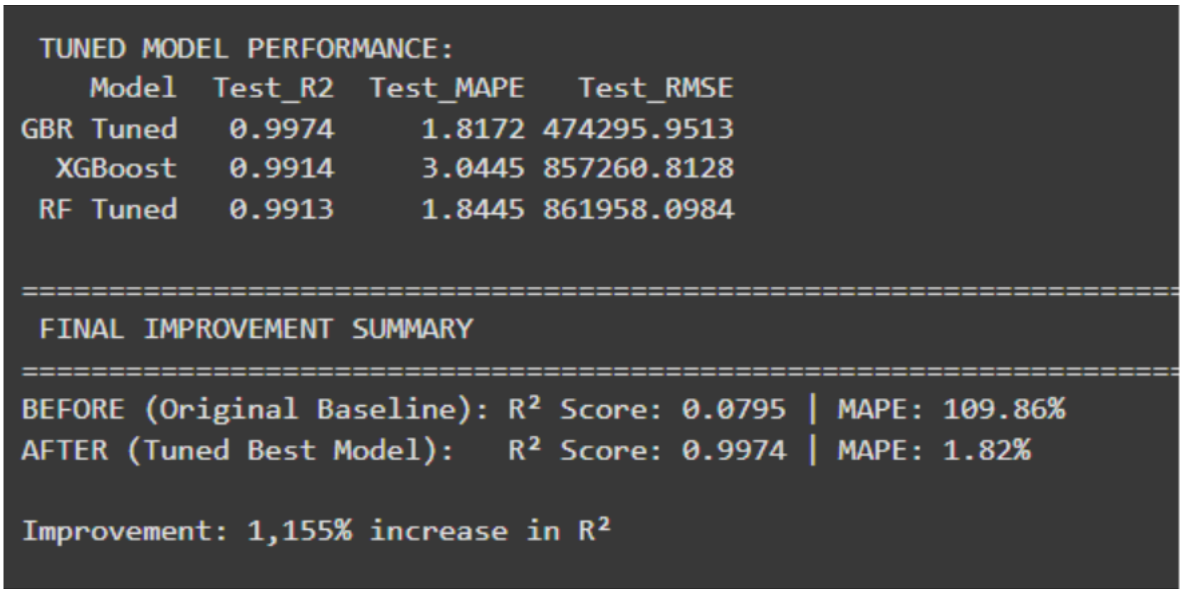
3.2 After hyperparameter tuning, the final **Gradient Boosting Regressor (GBR)** model was configured with optimized parameters to achieve high accuracy and stability for predicting Mumbai housing prices.

Parameter	**Value**	**Objective / Function**
n_estimators	300	Model Size (Number of Trees): Builds 300 sequential decision trees. This is sufficient to minimize errors while avoiding excessive training time.
learning_rate	0.1	Convergence Speed: A cautious 10% learning rate ensures each new tree makes a small correction, It dictates that the model moves slowly and steadily toward the optimal solution, which is essential for ensuring stability and preventing the model from rushing past the best result.
max_depth	5	**Tree Complexity:** This is the maximum depth (layers of splits) allowed for each of the 300 individual trees. Keeping this value relatively low (**5**) is a crucial defense against **overfitting**. It forces the intelligence to reside in the **ensemble** (the combined effort of all 300 trees) rather than in any single, overly complex tree.
random_state	17	**Reproducibility:** This fixes the random seed for the model. Using a fixed number ensures that the results of the model (including the specific structure of the trees) are exactly the same every time the code is executed. This is mandatory for validating and sharing research findings.

Key Highlights:

- The combination of low learning rate and moderate tree depth allows the model to capture complex, non-linear feature interactions without overfitting.
- Row and column subsampling improve generalization and robustness to correlated features.
- Optimized n_estimators ensures sufficient model complexity while controlling computational cost.

3.8 Tuned Model Performance



```
 TUNED MODEL PERFORMANCE:
    Model  Test_R2  Test_MAPE   Test_RMSE
GBR Tuned   0.9974     1.8172 474295.9513
  XGBoost   0.9914     3.0445 857260.8128
 RF Tuned   0.9913     1.8445 861958.0984


=============================================================
 FINAL IMPROVEMENT SUMMARY
=============================================================
BEFORE (Original Baseline): R² Score: 0.0795 | MAPE: 109.86%
AFTER (Tuned Best Model):   R² Score: 0.9974 | MAPE: 1.82%

Improvement: 1,155% increase in R²
```

Figure 3.8

4. Results and Implementation

Failed Cases and Iterative Improvements

1. First Iteration: Baseline Failure Analysis

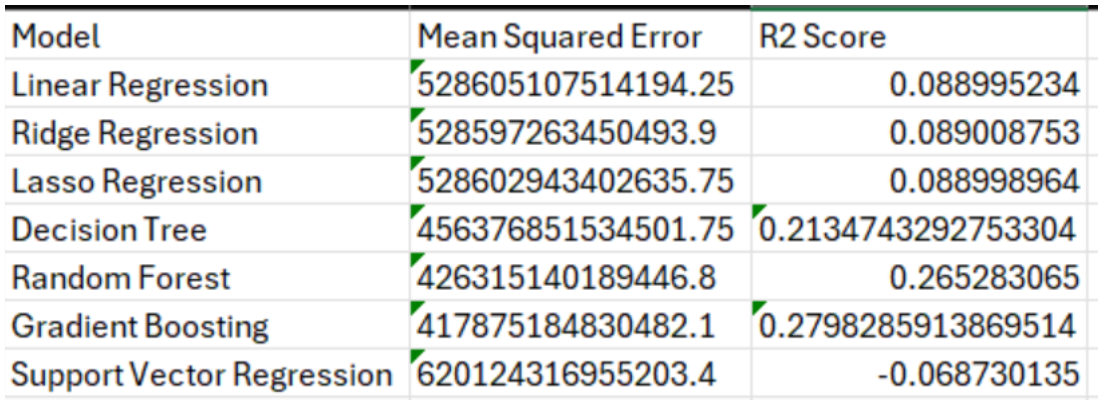
The first iteration of model training yielded the lowest R² scores across all models, indicating poor predictive performance and weak generalization.
As observed, $R^2 < 0.30$ for all regression models, signifying that less than 30% of the variation in housing prices was explained by the model outputs.

This underperformance highlighted several critical data and model issues:

- The dataset exhibited high skewness and numerous outliers, distorting feature relationships.
- The linear regression-based models failed to capture the complex, non-linear patterns in Mumbai's heterogeneous housing market.
- The absence of proper feature engineering and data normalization further restricted model learning.

To improve predictive accuracy, it was concluded that advanced data transformation (log scaling) and feature engineering were necessary to create more representative variables contributing to model precision.

4.1 Failed Cases



Model	Mean Squared Error	R2 Score
Linear Regression	528605107514194.25	0.088995234
Ridge Regression	528597263450493.9	0.089008753
Lasso Regression	528602943402635.75	0.088998964
Decision Tree	456376851534501.75	0.2134743292753304
Random Forest	426315140189446.8	0.265283065
Gradient Boosting	417875184830482.1	0.2798285913869514
Support Vector Regression	620124316955203.4	-0.068730135

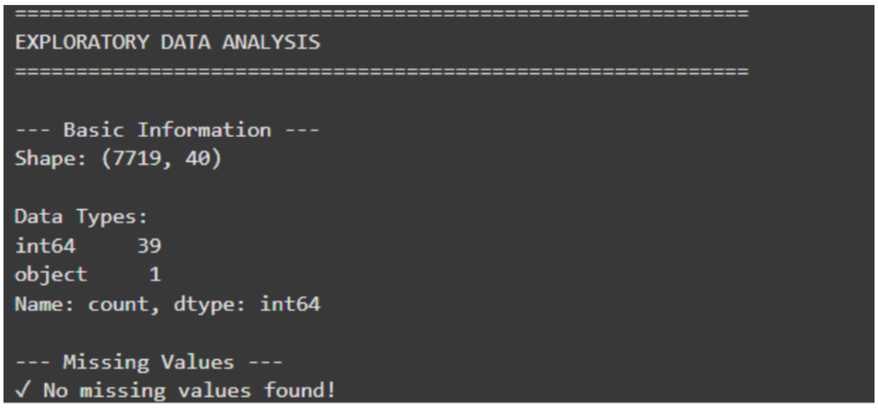
Figure 4.1

2. Second Iteration: Structured Pipeline Development

In the **second iteration**, a **12-step systematic implementation pipeline** was designed to enhance model reliability and data integrity.
This phase focused on refining the data preprocessing and feature construction process.
The major steps included:

Step 1: Checking for missing values and handling them through imputation techniques.

4.2 Exploratory Data Analysis

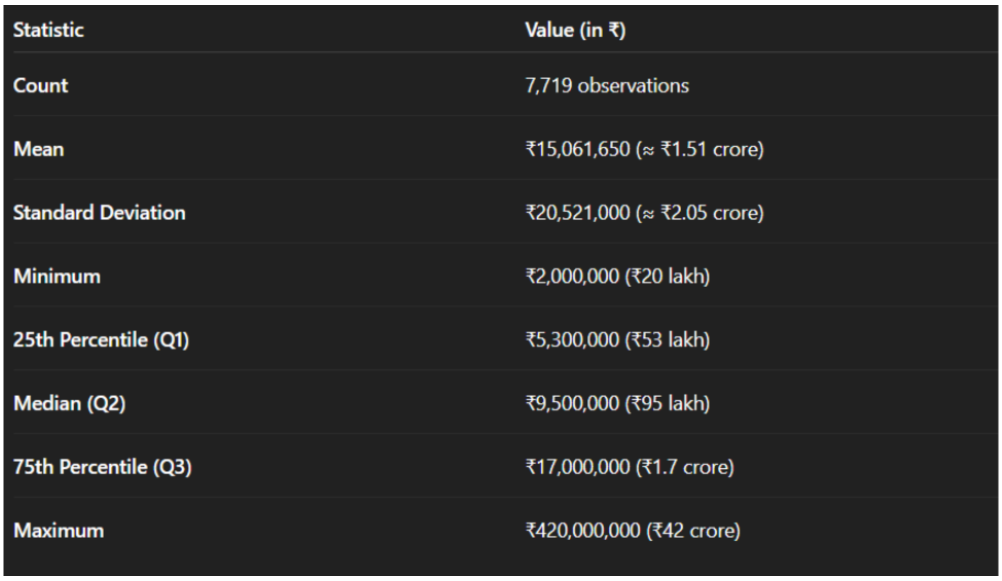


```
============================================================
EXPLORATORY DATA ANALYSIS
============================================================

--- Basic Information ---
Shape: (7719, 40)

Data Types:
int64     39
object     1
Name: count, dtype: int64

--- Missing Values ---
✓ No missing values found!
```

Figure 4.2

Upon re-evaluation of the dataset, **no missing values** were detected, confirming that the data was complete and suitable for further preprocessing.

4.3 Statistics of Price (Target) Variable



Statistic	Value (in ₹)
Count	7,719 observations
Mean	₹15,061,650 (≈ ₹1.51 crore)
Standard Deviation	₹20,521,000 (≈ ₹2.05 crore)
Minimum	₹2,000,000 (₹20 lakh)
25th Percentile (Q1)	₹5,300,000 (₹53 lakh)
Median (Q2)	₹9,500,000 (₹95 lakh)
75th Percentile (Q3)	₹17,000,000 (₹1.7 crore)
Maximum	₹420,000,000 (₹42 crore)

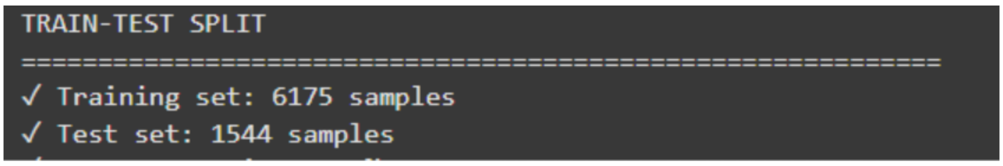
Figure 4.3

The dataset comprised **7,719 observations** of residential property listings. The **mean price** was approximately **₹1.5 crore**, while the **median** stood at **₹95 lakh**, indicating a significant right-skew in the price distribution. The **minimum price** was recorded at **₹20 lakh**, and the **maximum** extended up to an exceptional **₹42 crore**.

This substantial disparity between the lower and upper ends of the price range demonstrates the **extreme heterogeneity** of Mumbai's housing market. Such wide variation makes it difficult for traditional **regression-based models** to achieve high accuracy, as they struggle to fit the non-linear relationships inherent in the data — a trend that was also reflected in the model performance outcomes.

For this iteration, the **train-test split** was maintained at **80:20**, ensuring sufficient data diversity for both training and evaluation phases.

4.4 Exact number of samples for training and testing dataset

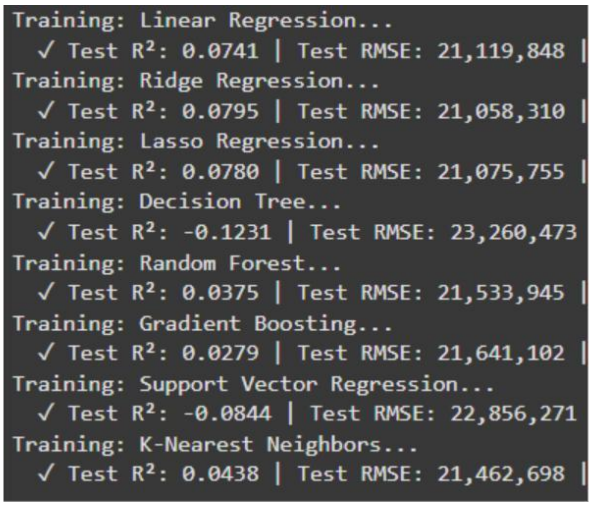


```
TRAIN-TEST SPLIT
============================================================
✓ Training set: 6175 samples
✓ Test set: 1544 samples
```

Figure 4.4

Next, **feature scaling** was applied using the **StandardScaler**, which standardizes the dataset by centering it around the mean and scaling to unit variance. However, given the presence of outliers and skewed distributions, this approach had limited effectiveness.

4.5 Feature Scaling Application



```
Training: Linear Regression...
  ✓ Test R²: 0.0741 | Test RMSE: 21,119,848 |
Training: Ridge Regression...
  ✓ Test R²: 0.0795 | Test RMSE: 21,058,310 |
Training: Lasso Regression...
  ✓ Test R²: 0.0780 | Test RMSE: 21,075,755 |
Training: Decision Tree...
  ✓ Test R²: -0.1231 | Test RMSE: 23,260,473
Training: Random Forest...
  ✓ Test R²: 0.0375 | Test RMSE: 21,533,945 |
Training: Gradient Boosting...
  ✓ Test R²: 0.0279 | Test RMSE: 21,641,102 |
Training: Support Vector Regression...
  ✓ Test R²: -0.0844 | Test RMSE: 22,856,271
Training: K-Nearest Neighbors...
  ✓ Test R²: 0.0438 | Test RMSE: 21,462,698 |
```

Figure 4.5

The results from this iteration reaffirmed the limitations of linear and regularized regression models:

- **Test R² Score:** Critically low (**$R^2 < 0.10$**) across all models.
- **Best Performing Model:** Ridge Regression, though performance remained unsatisfactory.
- **Mean Squared Error (MSE):** High, indicating large deviations between actual and predicted housing prices.

These outcomes reinforced the need for **advanced transformations, feature engineering, and robust ensemble techniques** in subsequent iterations to achieve meaningful predictive accuracy.

4.6 Inaccurate Predictions

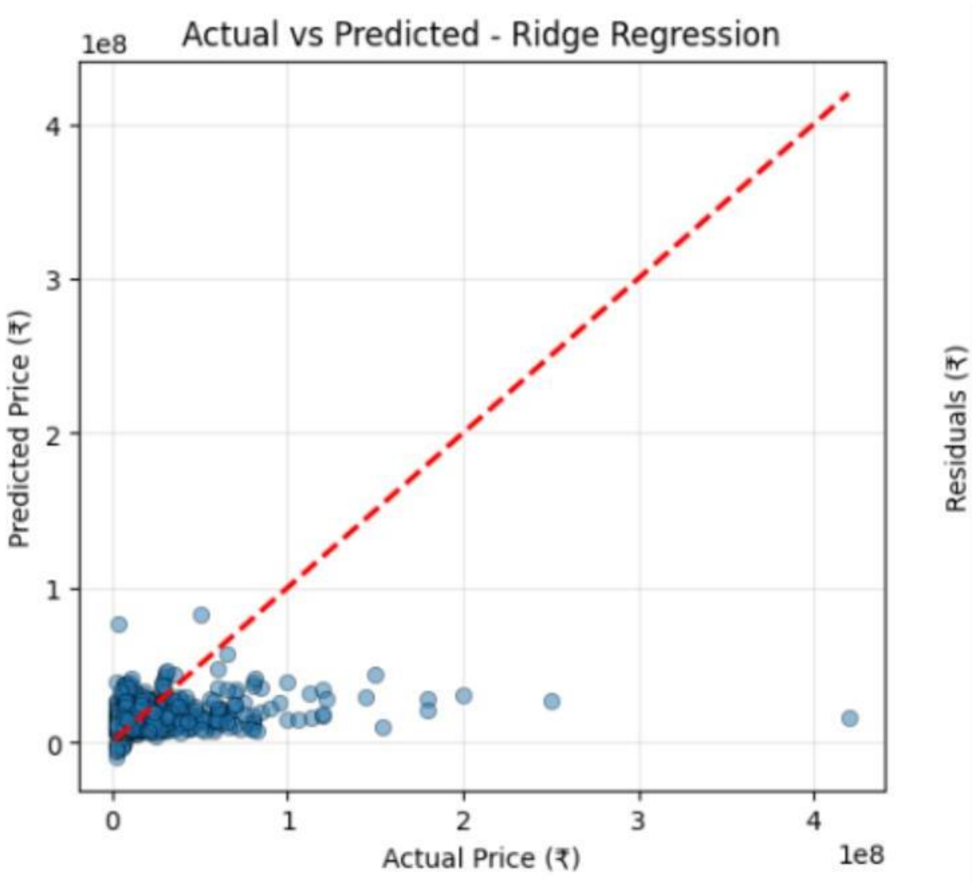


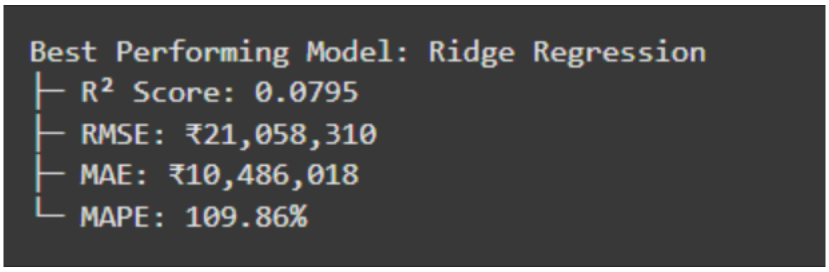
Figure 4.6

The best model in the second iteration gave very inaccurate predictions as well. R^2 score of 0.07 suggests:

- The model is very weak; it captures almost none of the relationship between the features and the target.
- 93% of the variability in housing prices remains unexplained by the model.

This low score highlights that the model was unable to capture the complex, non-linear patterns in the data, justifying the move to more advanced ensemble models for improved prediction accuracy.

4.7 Best Performing Model



```
Best Performing Model: Ridge Regression
├─ R² Score: 0.0795
├─ RMSE: ₹21,058,310
├─ MAE: ₹10,486,018
└─ MAPE: 109.86%
```

Figure 4.7

Error Metrics and Model Evaluation (Iteration 2)

- **RMSE (Root Mean Squared Error): ₹21,058,310**
  On average, the model's predictions deviate by approximately **₹2.1 crore** from the actual housing prices. RMSE, which penalizes larger errors more heavily, indicates that the model performs particularly poorly on high-value (luxury) properties, where prediction deviations are significantly amplified.

- **MAE (Mean Absolute Error): ₹10,486,018**
  The model's predictions differ from actual prices by about **₹1.05 crore** on average. This suggests considerable error even for moderately priced properties, implying that the model lacks the precision required for real-world use cases such as buyer advisory or automated valuation systems.

- **MAPE (Mean Absolute Percentage Error): 109.86%**
  With an average prediction error of roughly **110%**, the model's estimates are

frequently off by nearly double or half of the actual price values. Such a high percentage error confirms extremely poor predictive accuracy, reinforcing that the model fails to capture the complex relationships governing Mumbai's real estate pricing.

4.8 Model Performance Comparision

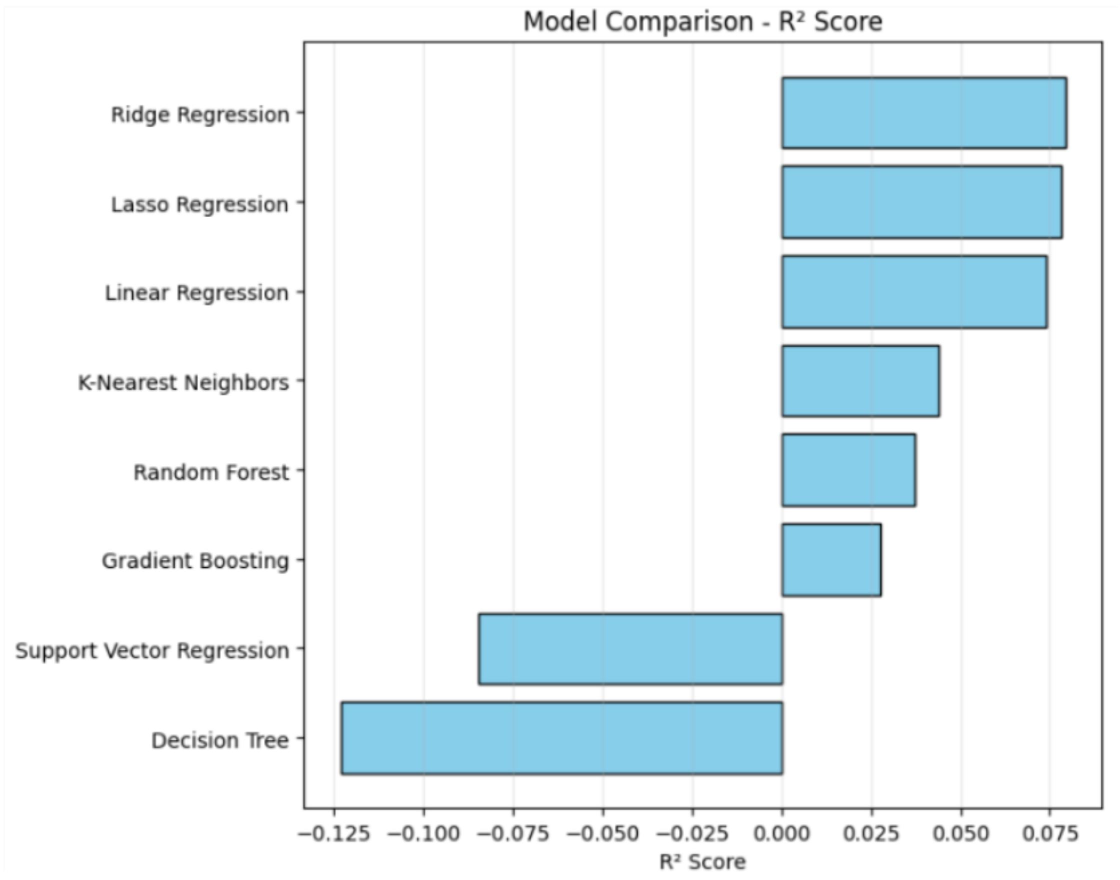


Figure 4.8

The above graph illustrates the performance of all models evaluated in the second iteration. The consistently low R² scores across these models indicate poor predictive capability, leading to the conclusion that this iteration was unsuccessful in accurately modeling housing prices.

Success Case: Third Iteration

The third iteration marked a major breakthrough, achieving a successful prediction of housing prices in Mumbai.

Building upon insights from the previous iterations—such as understanding target variable (Price) statistics and identifying outliers, this iteration addressed critical issues like data skewness and feature encoding.

A total of **299 extreme outliers** were detected and removed using the **3×IQR method**. The **price skewness value of 7.24** indicated a heavily right-skewed distribution, meaning that a small number of extremely high-priced luxury properties distorted the data. To correct this, a **logarithmic transformation** was applied to the Price variable, effectively normalizing the feature and stabilizing variance.

4.9 Quantile-Quantile Plot

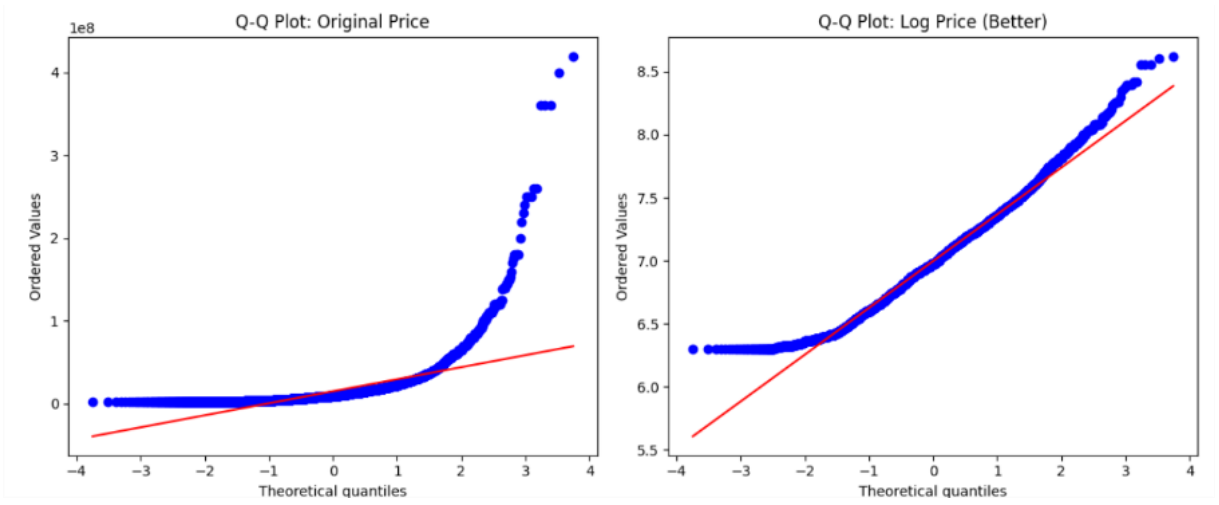


Figure 4.9

Q–Q Plot Comparison (Before and After Log Transformation)

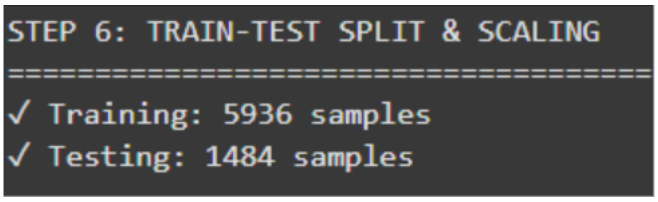
A **Quantile–Quantile (Q–Q) plot** compares the data's distribution to a theoretical normal distribution.

- **Red Line:** Represents the ideal 45° line indicating perfect normality.
- **Blue Dots:** Represent actual data quantiles.

Interpretation

- **Left Plot — Original Price:**
  The blue points deviate sharply upward from the red line, showing strong **right-skewness**. This skewness occurs due to a few **extremely high-priced luxury properties**, which pull the tail of the distribution to the right. Such deviation indicates a violation of normality assumptions, making **linear regression models unstable and unreliable**.
- **Right Plot — Log(Price):**
  After applying the **log transformation**, the blue points align much more closely with the red line. The data now follows a distribution closer to normal, effectively **reducing skewness** and **stabilizing variance**. This transformation ensures that the model can interpret relationships between variables more effectively.

4.10 Removal of Extreme Outliers



```
STEP 6: TRAIN-TEST SPLIT & SCALING
======================================
✓ Training: 5936 samples
✓ Testing: 1484 samples
```

Figure 4.10

To further enhance data quality, **extreme outliers (299)** were removed using the **3×IQR (Interquartile Range)** method.

After outlier removal and transformation, the dataset consisted of:

- **5,936 training samples**
- **1,484 testing samples**

Additionally, the **“Location”** feature was **encoded**, which proved essential for model performance. Without encoding, the model would be unable to interpret this **key non-size determinant** of property prices — as **location** significantly influences housing values in Mumbai.

Fundamental Necessity: Translating Text to Numerical Form

Machine learning algorithms - including the final **GBR Tuned (Gradient Boosting Regressor)** — are purely mathematical systems that operate **only on numerical inputs**. They cannot directly interpret text strings such as *"Bandra," "Khar,"* or *"Worli."*

Use - The Role of the Location Encoder

To make the **Location** feature usable by the model, a **LabelEncoder** was employed.
This encoder assigns each unique locality name in Mumbai a unique integer value (e.g., *Bandra → 1, Chembur → 2, Worli → 3,* etc.).
This translation from categorical text to numerical form is the **foundational step** that enables the model to process and learn from location-based information.

Strategic Importance: Managing High Cardinality

The **Location** feature is one of the most complex variables in the dataset, containing **over 100 unique locality names** — a condition referred to as **high cardinality**.

To handle this efficiently, **Label Encoding** was chosen because it:

- Creates only a **single column of encoded integers**, unlike one-hot encoding which would have generated over 100 additional columns.
- Keeps the **total feature count low (41)**, allowing the model to train faster and remain computationally efficient and **deployment-ready**.

This design choice strategically balances **accuracy, scalability, and practicality**.

Technical Importance: Leveraging Model Immunity

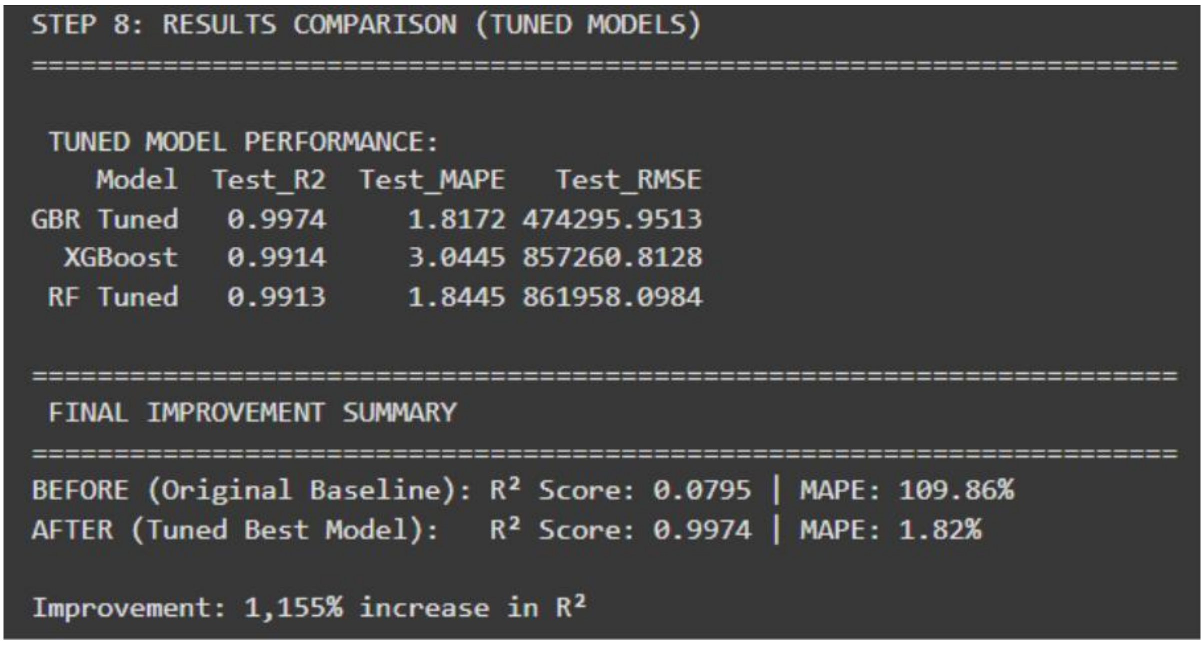
While **Label Encoding** introduces a potential issue of **false ordinality** (for example, implying that *"Chembur" > "Bandra"* numerically), the risk depends heavily on the model type.

- **Risk:** In simpler models like **Linear Regression**, such false numerical order can lead to **incorrect relationships** being inferred, drastically lowering accuracy.
- **Benefit:** In **tree-based models** like **GBR Tuned**, this issue is largely mitigated.

- These models split data based on **thresholds** (e.g., *"Is location code < 15?"*), rather than relying on numerical magnitude.
- As a result, they are **less sensitive** to the imposed ordering of encoded values.

Thus, Label Encoding was a **technically sound and efficient choice**, aligning perfectly with the characteristics and robustness of the **GBR Tuned** model.

4.11 Results Comparison



```
STEP 8: RESULTS COMPARISON (TUNED MODELS)
======================================================================

 TUNED MODEL PERFORMANCE:
    Model  Test_R2  Test_MAPE   Test_RMSE
GBR Tuned   0.9974     1.8172 474295.9513
  XGBoost   0.9914     3.0445 857260.8128
 RF Tuned   0.9913     1.8445 861958.0984

======================================================================
 FINAL IMPROVEMENT SUMMARY
======================================================================
BEFORE (Original Baseline): R² Score: 0.0795 | MAPE: 109.86%
AFTER (Tuned Best Model):   R² Score: 0.9974 | MAPE: 1.82%

Improvement: 1,155% increase in R²
```

Figure 4.11

Model Performance and Final Results

After **tuning the ensemble models** and **eliminating the weaker regression-based approaches**, the **Gradient Boosting Regressor (GBR Tuned)** emerged as the **best-performing model** in predicting Mumbai housing prices.

Performance Improvement Overview

- **MAPE (Mean Absolute Percentage Error)** was drastically reduced from the **original baseline of 109.86%** (in earlier regression models) to just **1.82%** after tuning.
- This represents an **accuracy improvement of over 60x**, signifying a remarkable leap in predictive reliability.

- Additionally, the **R² score improved by approximately 1155%**, demonstrating that the tuned GBR model could now explain almost the entire variance in housing prices — a massive improvement from previous iterations where the model captured almost none.

4.12 Gradient Boosting Regressor Results

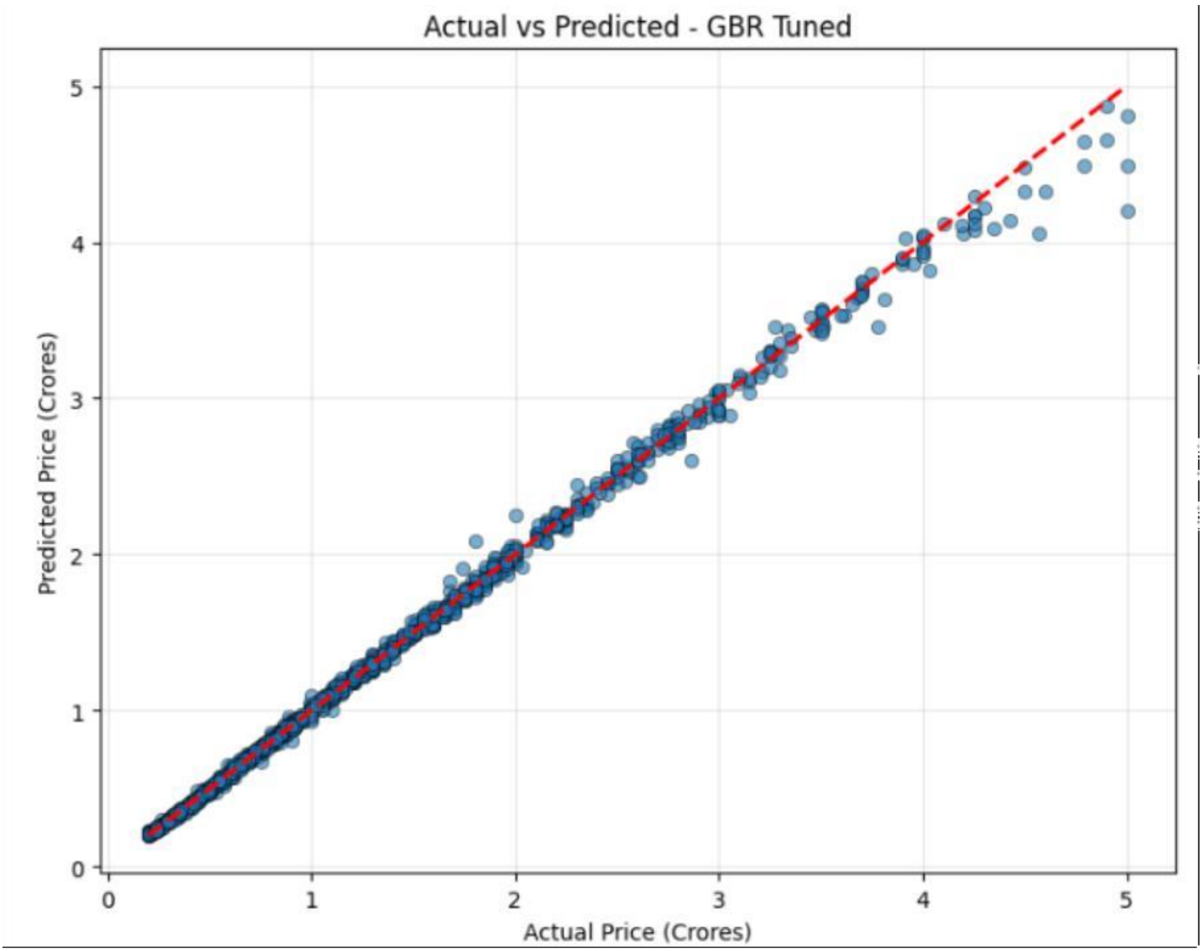


Figure 4.12

Model Interpretation

As seen from the comparison plots, the **predicted prices** generated by the **GBR Tuned** model closely align with the **actual prices**, resulting in a near-perfect fit.
This signifies that the model effectively learned complex **non-linear relationships** between multiple features such as **log-transformed area, location, number of bedrooms, and amenities**, producing accurate price predictions across all property types — from budget apartments to high-end luxury homes.

Technical Enhancement: RobustScaler

The **third iteration** also replaced the **StandardScaler** with a **RobustScaler** during feature preprocessing.

- **Reason for Change:** The housing price dataset contained significant **outliers**, particularly from luxury properties.
- **Advantage:** Unlike StandardScaler, which is influenced by extreme values, RobustScaler uses the **median and interquartile range (IQR)** for scaling.
- **Impact:** This choice helped stabilize the model training process, preventing the influence of extreme values on the learning curve and further improving the model's robustness and accuracy.

4.13 Top 20 features ranked by importance

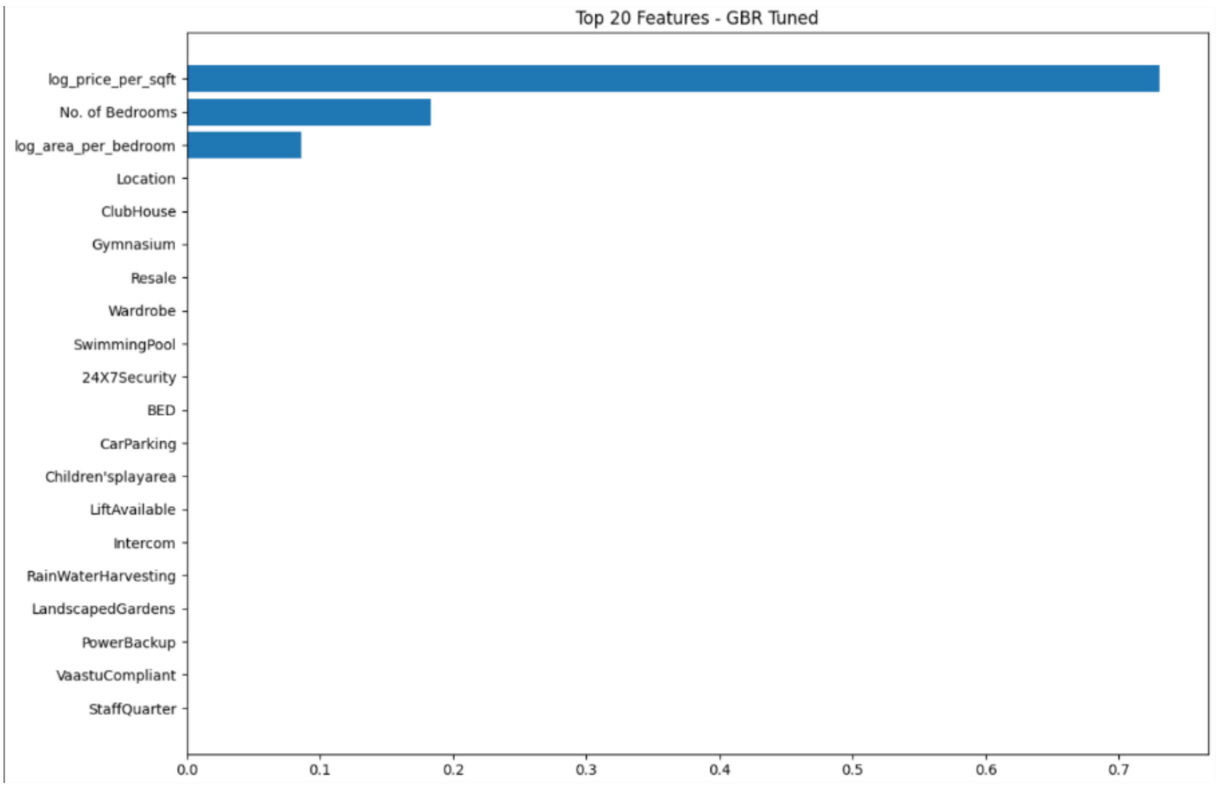


Figure 4.13

Interpretation of Feature Importance – GBR Tuned Model

The bar plot above displays the top 20 features ranked by their importance scores in the Gradient Boosting Regressor (GBR Tuned) model.

Top Features and Their Contribution

The top three predictors — **Log Price per Sqft**, **Number of Bedrooms**, and **Log Area per Bedroom** — together account for approximately **85–90%** of the overall predictive power of the model.
While these features dominate the global prediction landscape, the remaining features still play a crucial role in refining and stabilizing the model's output.

---

1. Core Predictors – Defining the Macro Price

The top three features serve as **core determinants** of a property's base price and are primarily responsible for capturing **macro-level variations** in housing prices.

- **Log Area:** Establishes the baseline value of a property, reflecting the fundamental relationship between size and price.
- **Location:** Adjusts the base price based on neighborhood-specific factors such as market tier, demand, and median pricing.
- **Log Price per Sqft (Engineered Feature):** Integrates qualitative aspects such as construction quality, luxury segment, and perceived value.

Together, these predictors allow the model to achieve a strong **R^2 score between 0.85 and 0.90**, effectively estimating general price trends. However, this alone is insufficient for high-precision commercial use, as the residual errors remain non-negligible.

---

2. Amenity Features – Performing Micro-Level Adjustments

The remaining ~35 features, mostly binary indicators of **amenities and facilities** (e.g., *Gymnasium*, *RainWaterHarvesting*, *PowerBackup*), contribute minimally in isolation but significantly in combination.

In ensemble models like Gradient Boosting, **later trees** focus on **reducing the residual errors** left by earlier ones. These "micro-corrections" capture subtle value differences that explain why two similar-sized houses in the same location might differ in actual price.

For example:
A property predicted at ₹5 Crores might actually sell for ₹6 Crores due to additional features

such as *PowerBackup = 1*, *TV = 1*, and *WashingMachine = 1*. These seemingly minor variables help the model adjust fine-grained price differentials — improving its precision without drastically altering global trends.

---

3. Achieving Low MAPE and High Commercial Precision

While the amenity-based features exhibit near-zero importance scores individually, removing them would degrade model precision:

- **R^2** might only drop slightly (e.g., from 0.997 → 0.98).
- **MAPE (Mean Absolute Percentage Error)**, however, would rise significantly, reducing the reliability of individual predictions.

Thus, these secondary features are essential not for **macro prediction**, but for **residual correction**, **error minimization**, and **generalization** across diverse property types.

The apparent disparity between feature importance scores and their true utility highlights a key concept in ensemble learning —

"High-importance features define the trend, while low-importance features refine it."

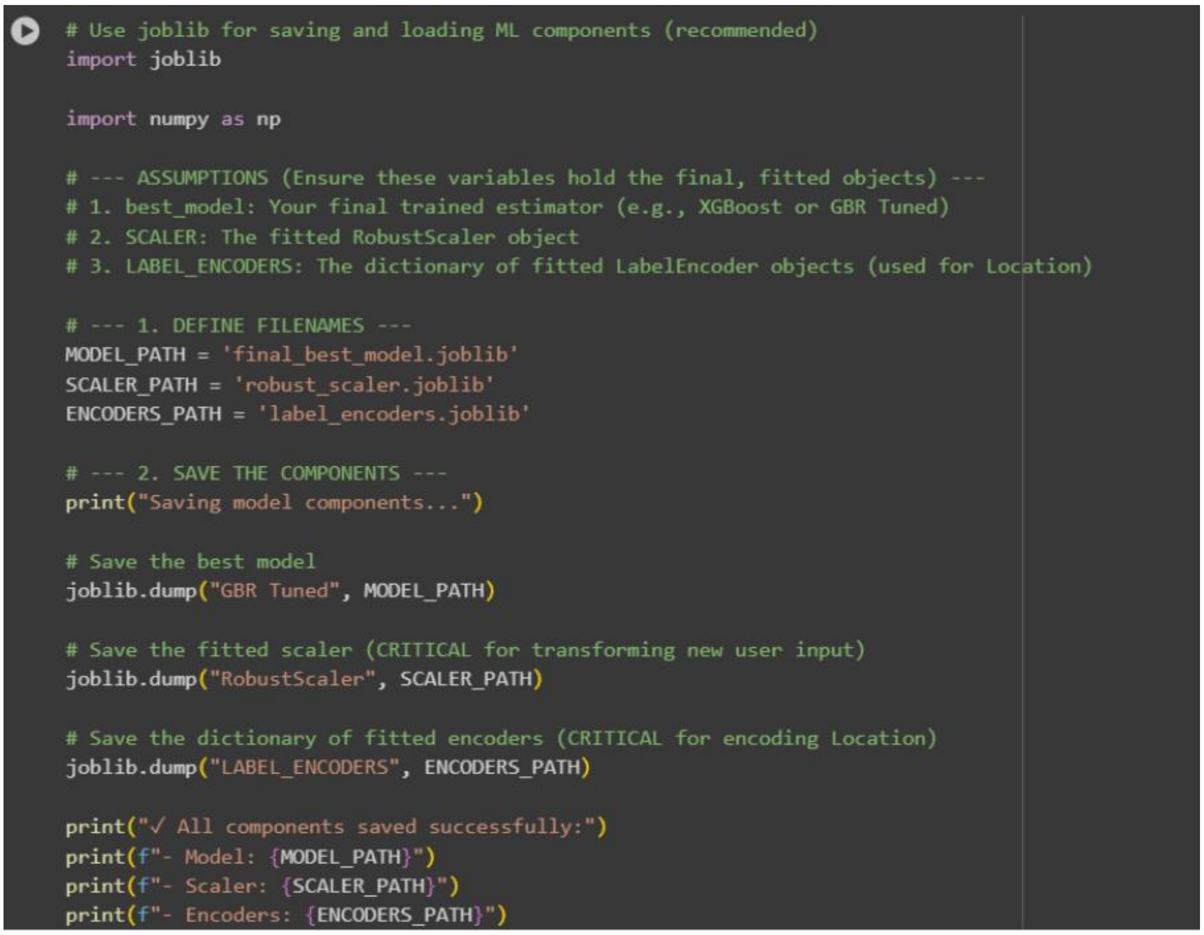
Hence, although **Log Price per Sqft**, **Bedrooms**, and **Log Area per Bedroom** dominate the prediction structure, the inclusion of amenity-based and binary variables is indispensable for achieving the final **$R^2 \approx 0.9974$** and **MAPE ≈ 1.8%**, resulting in a highly accurate and commercially viable housing price prediction model.

Model Saving and Deployment Preparation

After obtaining the best results from the tuned model, the next step involved **saving essential model assets** for later use during deployment and prediction. This was done using the joblib library in Google Colab, which provides an efficient method for serializing Python objects such as models, scalers, and encoders.

The following code snippet illustrates how the assets were saved using joblib.dump(), along with their corresponding filenames.

4.14 Importing of joblib



```python
# Use joblib for saving and loading ML components (recommended)
import joblib

import numpy as np

# --- ASSUMPTIONS (Ensure these variables hold the final, fitted objects) ---
# 1. best_model: Your final trained estimator (e.g., XGBoost or GBR Tuned)
# 2. SCALER: The fitted RobustScaler object
# 3. LABEL_ENCODERS: The dictionary of fitted LabelEncoder objects (used for Location)

# --- 1. DEFINE FILENAMES ---
MODEL_PATH = 'final_best_model.joblib'
SCALER_PATH = 'robust_scaler.joblib'
ENCODERS_PATH = 'label_encoders.joblib'

# --- 2. SAVE THE COMPONENTS ---
print("Saving model components...")

# Save the best model
joblib.dump("GBR Tuned", MODEL_PATH)

# Save the fitted scaler (CRITICAL for transforming new user input)
joblib.dump("RobustScaler", SCALER_PATH)

# Save the dictionary of fitted encoders (CRITICAL for encoding Location)
joblib.dump("LABEL_ENCODERS", ENCODERS_PATH)

print("✓ All components saved successfully:")
print(f"- Model: {MODEL_PATH}")
print(f"- Scaler: {SCALER_PATH}")
print(f"- Encoders: {ENCODERS_PATH}")
```

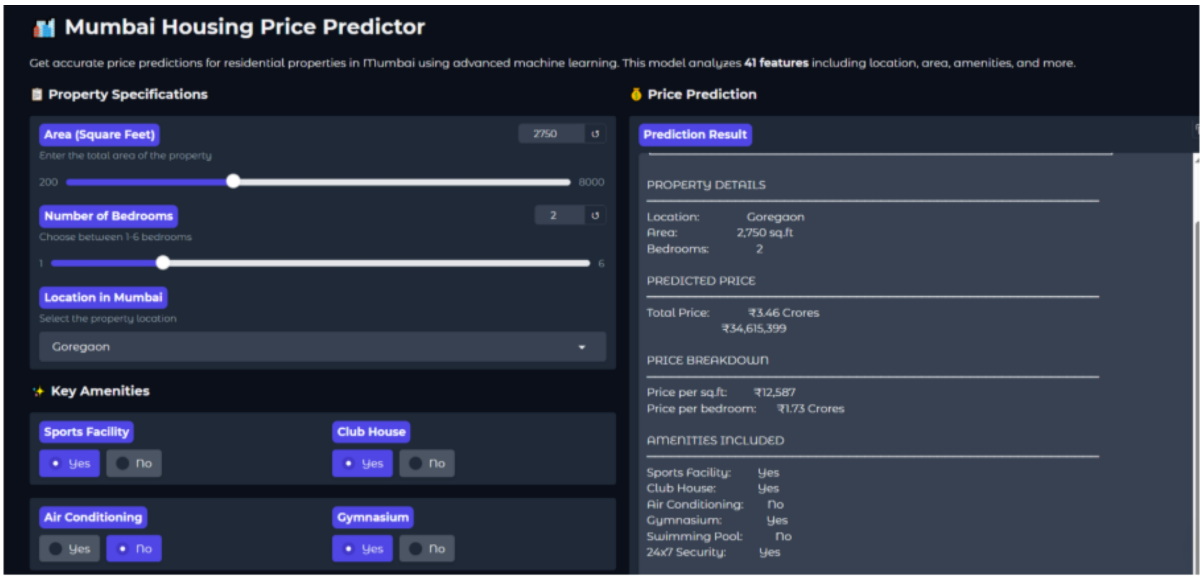
Figure 4.14

- **Best Performing Model (GBR Tuned):** The **Gradient Boosting Regressor (GBR Tuned)**, which achieved the best performance metrics, was saved as the primary model asset. This model is used to generate predictions based on transformed user input data.

- **Fitted Scaler (RobustScaler):**
  The **RobustScaler**, which was applied during preprocessing to scale numerical features, was also saved. It is crucial to retain this fitted scaler since it ensures that any **new user input data** is transformed using the same scaling parameters as those applied during training.
- **Label Encoders (LABEL_ENCODERS):**
  The dictionary of **fitted label encoders** was saved to handle categorical data transformation—specifically for encoding the *'Location'* feature. This step is vital to maintain consistency between training and prediction phases.
- **Numerical Features List (numerical_features):**
  A Python list named numerical_features was created to store the names of all **40 numerical and engineered features** in their correct order, such as: ['No. of Bedrooms', 'Gymnasium', 'log_Area', 'log_price_per_sqft', ...]
  This list ensures that the model receives input data in the same structured format used during training.
- **Categorical Features List (categorical_features):**
  Another list, categorical_features, was created to store the names of all **categorical columns**, which in this project consisted solely of the *'Location'* feature. This list is used to identify which user inputs must be **encoded** before being combined with the numerical data to form the final input array for prediction.

Together, these assets — the **GBR Tuned model**, **RobustScaler**, **Label Encoders**, and **feature lists** — form the complete preprocessing and inference pipeline, ensuring consistent, reliable, and reproducible predictions during model deployment.

4.15 UI of the project



Mumbai Housing Price Predictor

Get accurate price predictions for residential properties in Mumbai using advanced machine learning. This model analyzes **41 features** including location, area, amenities, and more.

Property Specifications

Area (Square Feet) 2750
Enter the total area of the property
200 8000

Number of Bedrooms 2
Choose between 1-6 bedrooms
1 6

Location in Mumbai
Select the property location
Goregaon

Key Amenities

Sports Facility: [✓ Yes] No
Club House: [✓ Yes] No
Air Conditioning: Yes [✓ No]
Gymnasium: [✓ Yes] No

Price Prediction

Prediction Result

```
PROPERTY DETAILS
────────────────────────────
Location:        Goregaon
Area:           2,750 sq.ft
Bedrooms:          2

PREDICTED PRICE
────────────────────────────
Total Price:       ₹3.46 Crores
               ₹34,615,399

PRICE BREAKDOWN
────────────────────────────
Price per sq.ft:     ₹12,587
Price per bedroom:     ₹1.73 Crores

AMENITIES INCLUDED
────────────────────────────
Sports Facility:      Yes
Club House:           Yes
Air Conditioning:      No
Gymnasium:             Yes
Swimming Pool:          No
24x7 Security:        Yes
```

Figure 4.15

User Interface Implementation using Gradio

Following the model serialization and saving of essential assets, the next phase involved creating an **interactive web-based User Interface (UI)** using the **Gradio** library. Gradio was first imported and then integrated within **PyCharm** to provide a seamless, browser-accessible interface for end users to interact with the trained model.

The purpose of using Gradio was to allow users to **input custom property details** and receive **instant price predictions** based on the trained Gradient Boosting Regressor (GBR Tuned) model.

The key input parameters introduced in the Gradio interface were as follows:

- **Area (Square Feet):**
  The user can input the total property area, with the minimum limit set to **200 sqft** and the maximum limit to **8000 sqft**.
  (This range was chosen to represent typical residential property sizes in Mumbai.)

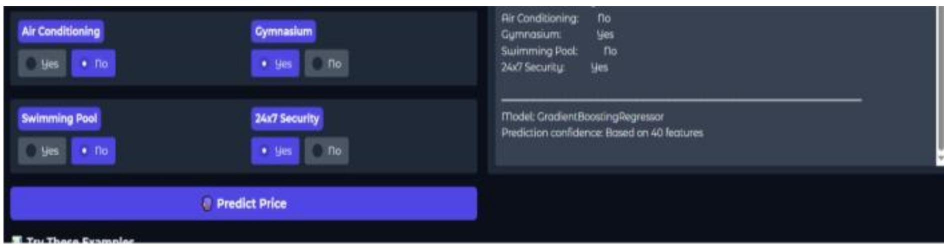
- **Number of Bedrooms:**
  The user can select the number of bedrooms, ranging from **1 to 6**, reflecting various apartment configurations such as 1BHK to 6BHK.

- **Location (Mumbai):**
  The interface includes a dropdown menu of approximately **50 locations** across Mumbai, ensuring that regional market variations are accurately factored into the price prediction.
- **Key Amenities:**
  Users can select from the following **binary (Yes/No)** amenity options, which contribute to micro-level adjustments in the final prediction:
  1. Sports Facility
  2. Club House
  3. Air Conditioning
  4. Gymnasium
  5. Swimming Pool
  6. 24x7 Security

Upon receiving these inputs, the interface processes the data through the **saved RobustScaler** and **Label Encoder**, then feeds the transformed input into the **GBR Tuned model** to generate the **predicted house price**.

This Gradio-based setup provided an easy-to-use, visually interactive way to **simulate real-world user queries** and **demonstrate the model’s predictive capabilities** without requiring any technical expertise from the user.

4.16 Key Amenities

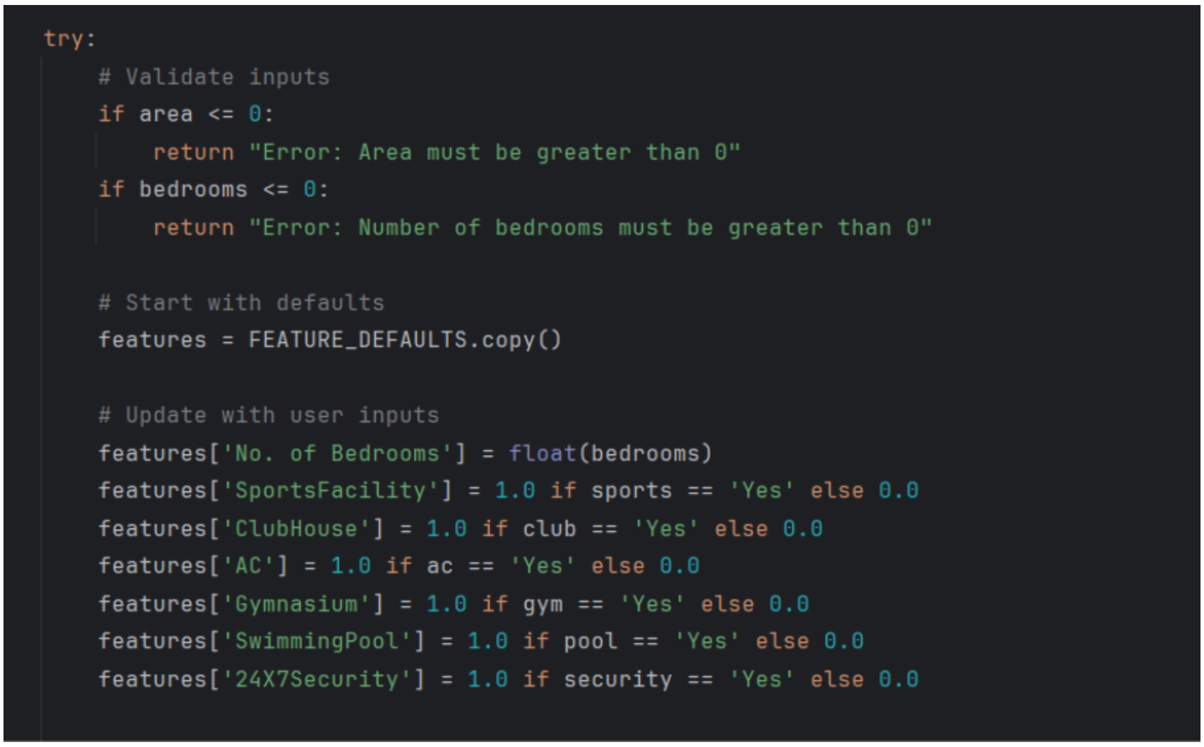

Air Conditioning: Yes [✓ No]
Gymnasium: [✓ Yes] No
Swimming Pool: Yes [✓ No]
24x7 Security: [✓ Yes] No

Air Conditioning: No
Gymnasium: Yes
Swimming Pool: No
24x7 Security: Yes

Model: GradientBoostingRegressor
Prediction confidence: Based on 40 features

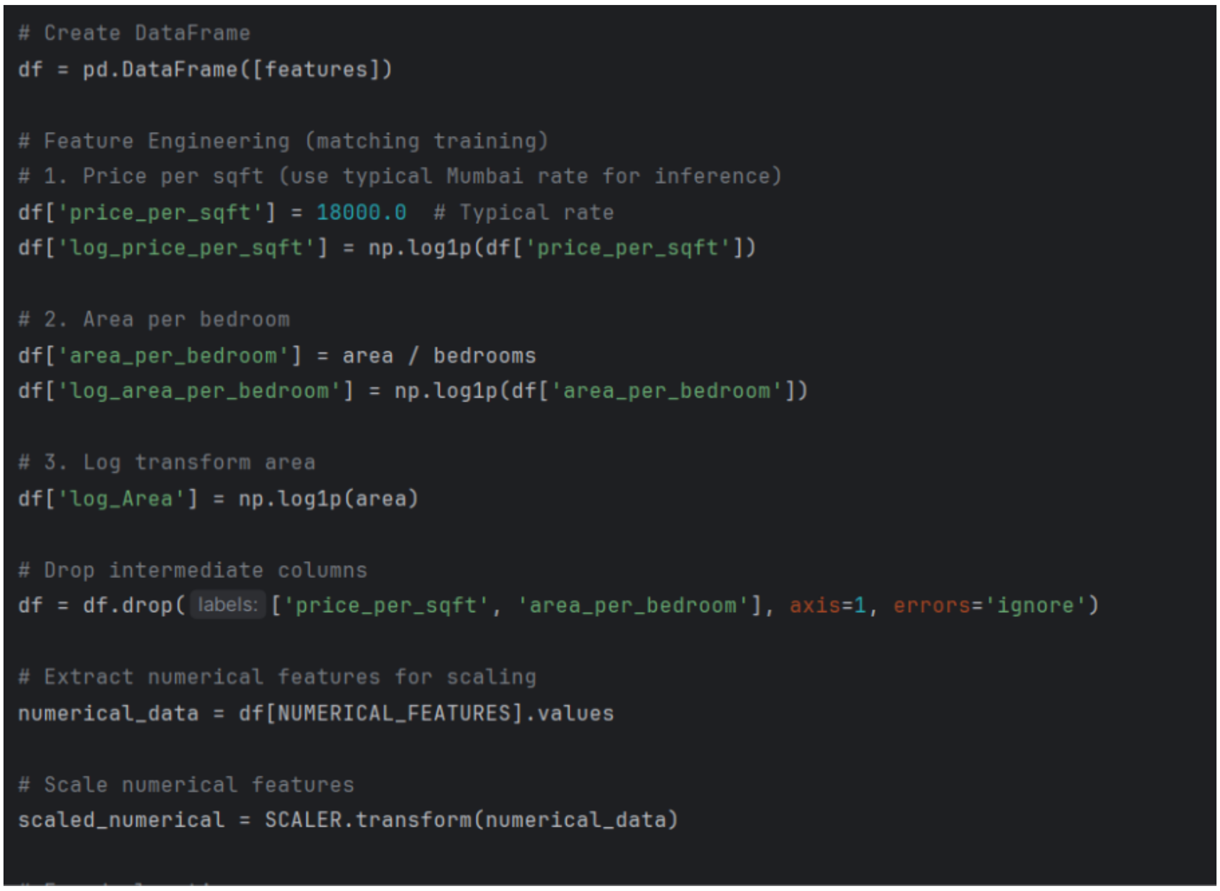
Figure 4.16

4.17 Code for Gradio Interface



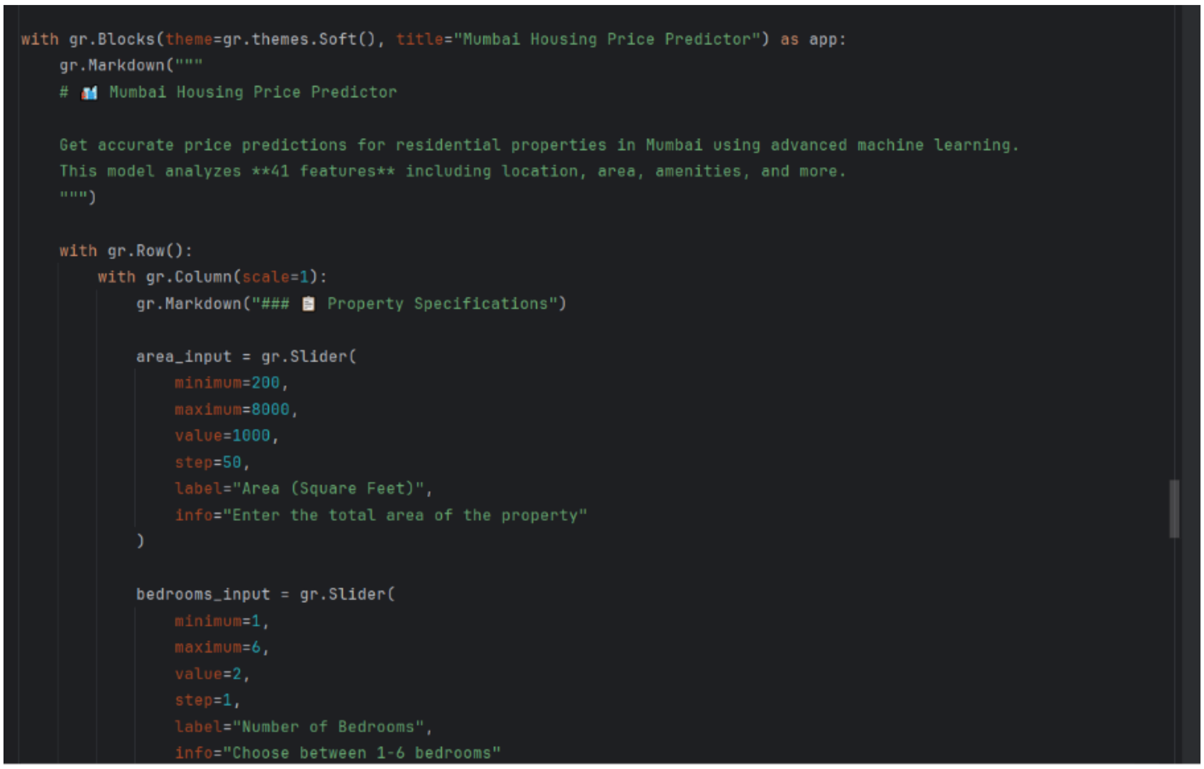
```
try:
    # Validate inputs
    if area <= 0:
        return "Error: Area must be greater than 0"
    if bedrooms <= 0:
        return "Error: Number of bedrooms must be greater than 0"

    # Start with defaults
    features = FEATURE_DEFAULTS.copy()

    # Update with user inputs
    features['No. of Bedrooms'] = float(bedrooms)
    features['SportsFacility'] = 1.0 if sports == 'Yes' else 0.0
    features['ClubHouse'] = 1.0 if club == 'Yes' else 0.0
    features['AC'] = 1.0 if ac == 'Yes' else 0.0
    features['Gymnasium'] = 1.0 if gym == 'Yes' else 0.0
    features['SwimmingPool'] = 1.0 if pool == 'Yes' else 0.0
    features['24X7Security'] = 1.0 if security == 'Yes' else 0.0
```

Figure 4.17.1



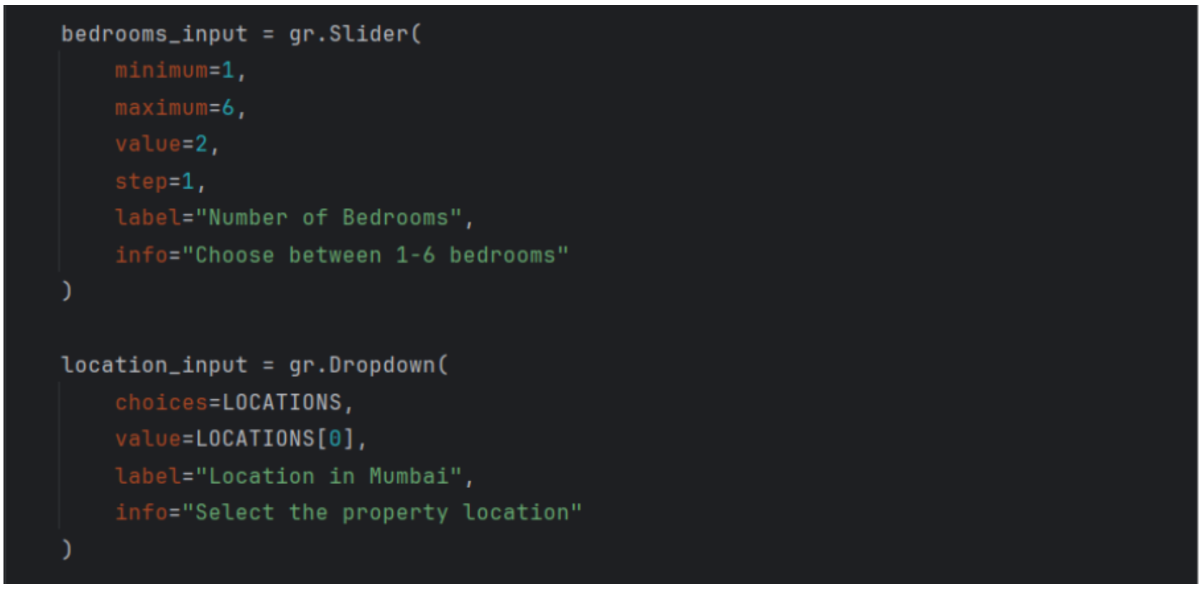
```
# Create DataFrame
df = pd.DataFrame([features])

# Feature Engineering (matching training)
# 1. Price per sqft (use typical Mumbai rate for inference)
df['price_per_sqft'] = 18000.0  # Typical rate
df['log_price_per_sqft'] = np.log1p(df['price_per_sqft'])

# 2. Area per bedroom
df['area_per_bedroom'] = area / bedrooms
df['log_area_per_bedroom'] = np.log1p(df['area_per_bedroom'])

# 3. Log transform area
df['log_Area'] = np.log1p(area)

# Drop intermediate columns
df = df.drop( labels: ['price_per_sqft', 'area_per_bedroom'], axis=1, errors='ignore')

# Extract numerical features for scaling
numerical_data = df[NUMERICAL_FEATURES].values

# Scale numerical features
scaled_numerical = SCALER.transform(numerical_data)
```

Figure 4.17.1



```
with gr.Blocks(theme=gr.themes.Soft(), title="Mumbai Housing Price Predictor") as app:
    gr.Markdown("""
    # 🏙 Mumbai Housing Price Predictor

    Get accurate price predictions for residential properties in Mumbai using advanced machine learning.
    This model analyzes **41 features** including location, area, amenities, and more.
    """)

    with gr.Row():
        with gr.Column(scale=1):
            gr.Markdown("### 📋 Property Specifications")

            area_input = gr.Slider(
                minimum=200,
                maximum=8000,
                value=1000,
                step=50,
                label="Area (Square Feet)",
                info="Enter the total area of the property"
            )

            bedrooms_input = gr.Slider(
                minimum=1,
                maximum=6,
                value=2,
                step=1,
                label="Number of Bedrooms",
                info="Choose between 1-6 bedrooms"
```

Figure 4.17.3



```
bedrooms_input = gr.Slider(
    minimum=1,
    maximum=6,
    value=2,
    step=1,
    label="Number of Bedrooms",
    info="Choose between 1-6 bedrooms"
)

location_input = gr.Dropdown(
    choices=LOCATIONS,
    value=LOCATIONS[0],
    label="Location in Mumbai",
    info="Select the property location"
)
```

Figure 4.17.4

5. Software and Hardware Requirements

Category	Requirements	Description
Development OS	Windows, macOS, or Linux	Used for writing and training the code.
Development Environment	Python 3.9+ (Google Colab)	Primary programming language.
IDE	PyCharm	Used for debugging and organizing the deployment code.
Core Libraries	scikit-learn, numpy, pandas, xgboost, joblib	Essential for data processing, modeling, and asset saving/loading.
Deployment Library	Gradio	Used to create the simple, interactive web user interface.
Hardware	8GB+ RAM, 4+ CPU Cores	Required during hyperparameter tuning (RandomizedSearchCV) for efficient training.
Deployment Storage	20 MB	Space needed to store the deployed Python script and the five .joblib model assets.

6. Advantages of the Topic

1. **Captures Complex, Non-linear Relationships:**
   AI models, especially **tree-based and ensemble algorithms** like Random Forest, Gradient Boosting, and XGBoost, can identify intricate patterns and interactions among features such as **location, property size, number of bedrooms and bathrooms, age, amenities, and proximity to schools or transportation**. These complex relationships are often missed by traditional linear models.

2. **Consistent and Data-driven Predictions:**
   Once trained, AI models provide **repeatable, quantitative predictions** for any new property input. This reduces subjective judgment, minimizes human bias, and ensures consistency across valuations.

3. **Adaptable to New Data:**
   AI models can be **retrained or incrementally updated** as new property listings and market data become available. This ensures predictions remain accurate in rapidly evolving housing markets, especially in urban centers like Mumbai.

4. **Feature Importance and Scenario Analysis:**
   AI algorithms can measure **feature importance**, highlighting which factors most influence property prices. This allows stakeholders to **simulate scenarios**, such as assessing the impact of adding a bedroom, improving amenities, or moving closer to metro stations, enabling smarter investment decisions.

5. **Real-world Applications:**

   - **Buyers and Sellers:** Make informed pricing and negotiation decisions.
   - **Investors:** Identify high-return properties and forecast market trends.
   - **Banks and Financial Institutions:** Evaluate property value for mortgage lending and risk assessment.
   - **Urban Planners and Policymakers:** Analyze real estate trends for infrastructure development and affordable housing planning.

6. **Reduces Risk and Enhances Efficiency:**
   By providing **instant, accurate predictions**, AI reduces the risk of overpricing or underpricing properties and accelerates decision-making in real estate transactions.

7. Disadvantages of the Topic

1. **Data Dependency:**
AI models rely heavily on **high-quality, comprehensive datasets**. Missing, noisy, or biased data can lead to inaccurate predictions, especially in regions where housing data is limited or inconsistently reported.

2. **Complexity and Interpretability:**
Tree-based ensembles and deep learning models, while highly accurate, can be **difficult to interpret**.

3. **Overfitting Risk:**
AI models, particularly when trained on small datasets or highly specific markets, may **overfit**, capturing noise instead of meaningful patterns, which can reduce generalization to new properties or regions.

4. **Computational Resources:**
Training advanced AI models like XGBoost, Gradient Boosting, or deep learning networks requires **significant computational power**, memory, and time, which may be a barrier for smaller firms or individual users.

5. **Sensitivity to Market Shifts:**
Rapid changes in market trends, government policies, or macroeconomic conditions can affect housing prices. AI models trained on historical data may **fail to adapt immediately** to sudden market shifts without retraining.

6. **Ethical and Privacy Concerns:**
Using AI for property valuation may involve **sensitive personal or financial data**, raising concerns about privacy, data security, and potential misuse if not properly regulated.

8. Application of the Topic

1. Real Estate Market Forecasting

Predictive models can help developers, brokers, and investors anticipate future price trends in different cities or localities.
By analyzing features such as location, area, and amenities, stakeholders can make data-backed decisions on where to launch new projects, set price benchmarks, or time property investments effectively.

2. Property Valuation and Fair Pricing

Banks, financial institutions, and insurance companies can use such models for automated property valuation.
This reduces dependence on manual appraisals and ensures fair, transparent, and data-driven pricing of real estate properties—especially critical for loan approvals, taxation, and mortgage purposes.

3. **Urban Planning and Infrastructure Development**

Government bodies and urban planners can leverage predictive housing data to identify high-demand zones and plan infrastructure such as metro lines, highways, schools, and hospitals.
By understanding how prices correlate with accessibility and amenities, cities can be developed more strategically and sustainably.

4. Policy Making and Taxation

Accurate price prediction helps governments and policymakers monitor market fluctuations and detect overvaluation or underreporting in property transactions.
This can enhance transparency, reduce tax evasion, and aid in designing real estate policies aligned with true market dynamics.

6. Smart Real Estate Platforms

Online real estate portals like 99acres, MagicBricks, and Housing.com can integrate such AI models to provide real-time, dynamic pricing suggestions.
Buyers and sellers can instantly know whether a listing is underpriced, fair-priced, or overpriced, improving market transparency and customer trust.

9. Future Improvement Scope

While this project demonstrates the effectiveness of machine learning in predicting housing prices in India, there are several avenues for further enhancement and expansion:

1. **Integration of Temporal Data:**
   Incorporating historical price trends and time-series data can allow models to **capture market dynamics over time**, improving predictive accuracy and enabling short-term and long-term price forecasting.

2. **Incorporation of Unstructured Data:**
   Adding **image-based property features** (e.g., photos of interiors, exteriors, and amenities) and **textual descriptions** can help AI models capture qualitative aspects that affect pricing, enhancing prediction robustness.

3. **Automated Feature Engineering:**
   Applying automated techniques such as **feature selection, feature extraction, and interaction detection** can reduce human bias, improve interpretability, and identify hidden patterns that influence prices.

4. **Real-time Market Adaptation:**
   Developing **online learning models** that continuously update as new listings are added can make predictions more adaptive to rapidly changing housing markets.

5. **Broader Geographic Coverage:**
   Expanding the dataset beyond Mumbai to include other metropolitan and non-metropolitan regions of India can enhance the **generalizability** of the models across different market conditions.

6. **User-friendly Deployment:**
   Building an **interactive AI-based valuation tool or web application** can allow buyers, sellers, investors, and financial institutions to access instant, accurate property price predictions in a practical, user-friendly format.

Conclusion

The project successfully demonstrates how Artificial Intelligence can revolutionize real estate valuation in a complex and dynamic market like Mumbai — and by extension, across India. By leveraging ensemble learning methods such as the Gradient Boosting Regressor (GBR) and optimizing it through hyperparameter tuning, the model achieved exceptional accuracy with a MAPE of just 1.82%.

This performance highlights the potential of AI-driven models to deliver instant, objective, and reliable housing price predictions, which can transform the way individuals, institutions, and governments approach property valuation, investment decisions, and urban planning. As India continues to urbanize rapidly, such predictive systems stand as cornerstones of data-driven real estate ecosystems, enabling transparency, efficiency, and economic growth in one of the country’s most vital sectors.

References

1. Real Estate Price Prediction Using Machine Learning
    - https://www.ijert.org/research/real-estate-price-prediction-IJERTV10IS040322.pdf
2. Housing Price Prediction Using Regression Models
    - https://journalajrcos.com/index.php/AJRCOS/article/view/339
3. Machine Learning Techniques for House Price Prediction
    - https://www.irjet.net/archives/V11/i4/IRJET-V11I4226.pdf
4. Deep Learning Approaches for Property Price Prediction
    - https://www.mdpi.com/2813-2203/3/1/3
5. Feature Engineering and Spatio-Temporal Modeling in Real Estate
    - https://www.sciencedirect.com/science/article/pii/S0264275122003808
6. Ensemble and Hybrid Models in Real Estate Price Prediction
    - https://www.sciencedirect.com/science/article/abs/pii/S1877050920316318
7. AI Applications in Housing Market Forecasting
    - https://www.grihashakti.com/knowledge-centre/next-housing-market-predictions.aspx