Project Report

On

TerraTrend : House Price Prediction System

Submitted in the partial fulfillment for the award of Post Graduate

Diploma in Big Data Analytics (PG-DBDA)

From Know-IT ATC, CDAC ACTS, Pune

Guided by: Mr. Milind Kapse

Submitted By:

Siddhant Annadate (250243025004)

Sakshee Kandarkar (250243025028)

Santosh Kriplani (250243025033)

Siddhesh Suryavanshi (250243025043)

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Siddhant Annadate (250243025004)
Sakshee Kandarkar (250243025028)
Santosh Kriplani (250243025033)
Siddhesh Suryavanshi (250243025043)

have successfully completed their project on

TerraTrend : House Price Prediction

Under the guidance of **Mr Millind Kapse**

ACKNOWLEDGEMENT

TerraTrend: House Price Prediction System is designed to predict house prices based on various paramenters. It helps various stakeholders to get an overall estimation of prevailing house prices in various locations. Multiple parameters that have a significant impact on the final price of the house have been taken into consideration and accordingly a machine learning model has been trained in order to have an approximate prediction of house prices so that it enables the various stakeholders such as customers, sellers, and other various stakeholders in the market to make prudent decisions regarding the purchase of real estate.

We are all grateful to Milind Kapase Sir and for their invaluable help while working on this project. Their advice and support enabled us to overcome a variety of challenges and complexities during the course of the project.
We are grateful to Mr. Shrinivas Jadhav (Vice-President, Know-it, Pune) for his assistance and support throughout the Post Graduate Diploma in Big Data Analytics (PGDBDA) course at CDAC ACTS, Pune.

Our heartfelt gratitude goes to Mrs. Dhanashree Rangole ma'am(Course Coordinator, PG-DBDA), who provided all of the necessary support and kind coordination to provide all of the necessities, such as required hardware, internet access, and extra lab hours, to complete the project and throughout the course up to the last day here at CDAC KnowIT, Pune.

Siddhant Annadate (250243025004)

Sakshee Kandarkar (250243025028)

Santosh Kriplani (250243025033)

Siddhesh Suryavanshi (250243025043)

TABLE OF CONTENTS

ABSTRACT

House price prediction is crucial for bringing transparency to the real estate market and enabling stakeholders to make prudent financial decisions. This project addresses this by developing a **regression model** to estimate house prices in **[Your City/Region, e.g., Mumbai]**.

Using a dataset of over **[Number, e.g., 10,000]** property listings, the model was trained on parameters including BHK, bathrooms, carpet area, location, and furnishing status. Key preprocessing steps involved **one-hot encoding** for categorical data and handling missing values.

Several machine learning algorithms, such as Linear Regression and Gradient Boosting, were evaluated. The **Random Forest Regressor** proved to be the most effective, achieving an **R2 score of [Your Score, e.g., 0.88]** and a **Mean Absolute Error of [Your MAE, e.g., ₹4.2 Lakhs]**.

This solution empowers customers, sellers, and brokers to make informed decisions with greater confidence, reducing price uncertainty and promoting fair transactions in the real estate market. Future work could involve deploying this model into a user-friendly web application.

Introduction : -

The real estate market is a cornerstone of any economy, representing a significant investment for individuals and a key indicator of economic health. In a rapidly urbanizing country like India, the property market is particularly dynamic, characterized by fluctuating prices and a vast array of options. However, this dynamism often leads to a lack of price transparency and information asymmetry, creating significant challenges for buyers, sellers, and real estate professionals. Stakeholders often struggle to determine a fair and accurate market value for a property, leading to uncertainty, prolonged negotiations, and potentially inequitable transactions. This ambiguity highlights the critical need for objective, data-driven tools to navigate the complexities of the real estate landscape.

Problem Statement :-

The core problem is the difficulty in accurately estimating house prices due to the multitude of factors that influence them. Without a reliable valuation method, stakeholders are prone to making uninformed decisions based on incomplete information or biased advice. This can result in buyers overpaying for properties, sellers undervaluing their assets, and a general lack of trust and efficiency in the market.

Aim and Objectives :-

The primary aim of this project is to develop a robust machine learning model that can accurately predict house prices based on their key attributes. The main objectives are:

1. To perform a comprehensive exploratory data analysis (EDA) to understand the relationships between various property features and their impact on price.
2. To preprocess and engineer features from a raw dataset to prepare it for machine learning.
3. To train, evaluate, and compare several regression models (such as Linear Regression, Random Forest, and Gradient Boosting) to identify the best-performing algorithm.
4. To develop a final predictive model that provides reliable price estimates, thereby empowering stakeholders to make informed decisions.

Data Collection:

The project utilizes a publicly available dataset sourced from Kaggle. The dataset contains thousands of property listings from a major metropolitan area and includes a wide range of features, such as **BHK (Bedrooms, Hall, Kitchen)**, **number of bathrooms**, **carpet area**, **super area**, **location**, **furnishing status**, and **construction status**. This rich dataset provides a solid foundation for training a model that can learn the intricate patterns connecting property attributes to their market value.

Predictive Features (Independent Variables) :

- **Location**: A **categorical feature** that specifies the city or locality where the property is situated.
- **Carpet Area in sqft**: A **numerical feature** representing the net usable floor area of the property, measured in square feet.
- **Super Area in sqft**: A **numerical feature** representing the total built-up area, which includes the carpet area plus the thickness of walls and common spaces (e.g., lobby, staircase).
- **BHK**: A **numerical feature** indicating the number of rooms, specifically **B**edrooms, **H**all, and **K**itchen.
- **Bathroom**: A **numerical feature** for the total number of bathrooms in the property.
- **Balcony**: A **numerical feature** for the total number of balconies.
- **Furnishing**: A **categorical feature** that describes the furnishing status of the property (e.g., Unfurnished, Semi-Furnished, or Fully-Furnished).
- **Facing**: A **categorical feature** that indicates the direction the main entrance of the house faces (e.g., North, East, South, West).
- **Status**: A **categorical feature** indicating the construction status of the property (e.g., Ready to Move or Under Construction).

- **Transaction**: A **categorical feature** specifying the type of sale (e.g., New Property or Resale).
- **Ownership**: A **categorical feature** indicating the type of legal ownership (e.g., Freehold, Leasehold, or Power of Attorney).
- **Price (in rupees)**: A **numerical feature** that likely represents the price per square foot of the property.

Target Variable (Dependent Variable)

- **Final Amount**: A **numerical feature** representing the overall transaction value or final listed price of the property. **This is the target variable that the model aims to predict.**

SYSTEM REQUIREMENTS

Hardware Requirements:

- Platform – Windows 7 or above
- RAM – Recommended 8 GB of RAM
- Peripheral Devices – Keyboard, Monitor, Mouse
- WiFi connection with minimum 2 Mbps speed

Software Requirements:

- Language: Python 3
- Machine Learning
- Tableau
- OS – Windows
- Pyspark

FUNCTIONAL REQUIREMENTS

1) Python 3:

- Python is a high-level programming language that is easy to learn and use.
- Python is an interpreted language, which means that code can be executed on the fly, without the need for compilation.
- Python is open source and free to use, with a large and active community of developers contributing to its development and maintenance.
- Python has a vast collection of third-party libraries and packages, such as NumPy, Pandas, Matplotlib, and Scikit-learn, among others, that make it easy to perform data analysis.

2) Tableau:

- Tableau is a data visualization and business intelligence software that allows users to connect, analyse, and share data in a visual and interactive way.
- It offers a user-friendly drag-and-drop interface that enables users to create interactive dashboards, reports, and charts without the need for complex coding or programming.
- Tableau supports various data sources, including spreadsheets, databases, cloud services, and bigdata platforms, such as Hadoop and Spark.

3) Data Cleaning:

- Data cleaning is a crucial step in the data mining process and significantly impacts model development. Despite its importance, it is often overlooked.
- Ensuring data quality is essential for effective information management, and data cleaning addresses various quality issues.
- Without proper data cleaning, analysis and modeling can produce erroneous or biased results, leading to serious consequences for businesses and organizations. Thus, it is a vital part of data preparation that influences the accuracy and reliability of insights and decisions.
- By improving data quality, organizations can gain a clearer understanding of their operations, customers, and market trends, enabling more informed and effective decision-making.

MACHINE LEARNING ALGORITHMS

- Machine learning is a subfield of artificial intelligence that involves developing algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed. The goal of machine learning is to enable computers to improve their performance over time by learning from experience and feedback.
- In our project, we applied various Algorithms such as Decision Tree, Random Forest, CatBoost, XGBoost, Adaboost, Support Vector Machine (SVM), K – Nearest Neighbour (KNN). The Linear Regression is also used for base evaluation. After the implementation, we analyzed the accuracy of all the algorithms on our data.

Linear Regression

Linear Regression is a fundamental statistical algorithm that models the linear relationship between a dependent variable and one or more independent variables. It works by fitting a linear equation (a line or a hyperplane) to the observed data, aiming to minimize the sum of the squared differences between the actual and predicted values.

- **Pros:**
  - **Highly Interpretable:** It is very easy to understand the relationship between the input features and the output. The coefficients of the model represent the impact of each feature.
  - **Computationally Fast:** It is very fast to train and does not require much computational power.
  - **Good Baseline Model:** It serves as an excellent baseline to measure the performance of more complex models.
- **Cons:**
  - **Assumes Linearity:** Its primary limitation is that it assumes a linear relationship between the features and the target variable, which is often not the case in real-world scenarios.
  - **Sensitive to Outliers:** The model can be heavily influenced by outliers in the data.
  - **Limited Complexity:** It often fails to capture the more complex, non-linear patterns present in a dataset.

Decision Tree:

A Decision Tree is a fundamental supervised machine learning algorithm used for both classification and regression tasks. It works by recursively splitting the data into subsets based on the most significant features, creating a tree-like model of decisions. Each internal node represents a test on a feature, each branch represents the outcome of the test, and each leaf node represents the final prediction.

- **Pros:**
  - o **Interpretability:** Decision trees are easy to understand and visualize, as they mimic human decision-making processes.
  - o **No scaling required:** They are not sensitive to the scaling of the data.
  - o **Handles both numerical and categorical data:** Can process different data types without extensive preprocessing.

- **Cons:**
  - o **Prone to overfitting:** Without proper pruning, a decision tree can become too complex and memorize the training data, performing poorly on new data.
  - o **Instability:** Small changes in the data can lead to a completely different tree structure.

Random Forest:

Random Forest is an ensemble learning method that builds multiple decision trees during training and combines their results. For regression tasks, it averages the predictions of all individual trees to produce a final, more stable, and accurate forecast. It introduces randomness by building each tree on a random subset of the data (bagging) and considering only a random subset of features at each split.

- **Pros:**
  - o **High accuracy and robustness:** Reduces the risk of overfitting by averaging many different decision trees.
  - o **Less prone to overfitting:** The randomness in both data sampling and feature selection makes the model more robust.
  - o **Handles non-linear data:** Can capture complex relationships in the data.

- **Cons:**
  - **Less interpretable:** As it's an ensemble of many trees, it's difficult to visualize and understand the entire model's decision-making process.
  - **Computationally intensive:** Training can be slower and require more memory than a single decision tree.

CatBoost

CatBoost (Categorical Boosting) is a gradient boosting framework that is particularly effective at handling categorical features. It uses a novel algorithm to handle categorical features directly, avoiding the need for extensive one-hot encoding, which can sometimes lead to an exponential increase in feature dimensionality. CatBoost builds trees sequentially, with each new tree correcting the errors of the previous ones.

- **Pros:**
  - **Handles categorical features automatically:** Reduces the need for manual preprocessing of categorical variables.
  - **Strong performance:** Often achieves state-of-the-art results on a wide range of tasks.
  - **Robustness to overfitting:** Uses ordered boosting to combat prediction shift and overfitting.

- **Cons:**
  - **Slower training time:** Can be slower to train compared to other boosting frameworks, especially on larger datasets.
  - **Requires more memory:** The ordered boosting algorithm can be memory-intensive.

XGBoost

XGBoost (eXtreme Gradient Boosting) is a highly optimized and scalable implementation of gradient boosting. It is known for its speed and high performance. Like other boosting algorithms, it builds trees sequentially, but it incorporates regularization techniques to prevent overfitting and provides advanced features like parallel processing and tree pruning.

- **Pros:**
  o **High performance and speed:** One of the fastest and most powerful algorithms, widely used in data science competitions.
  o **Regularization:** Built-in regularization techniques (L1 and L2) help to control model complexity and prevent overfitting.
  o **Flexibility:** Can be used for various tasks and supports different objective functions.

- **Cons:**
  o **Parameter tuning:** Has a large number of hyperparameters that can be challenging to tune for optimal performance.
  o **Less interpretable:** Similar to Random Forest, it is a complex black-box model.

AdaBoost (Adaptive Boosting)

AdaBoost (Adaptive Boosting) is an ensemble learning algorithm that builds a strong predictive model by combining multiple weak learners (typically simple decision trees). It works sequentially, where each subsequent model gives more weight to the data points that were incorrectly predicted by the previous models. The final prediction is a weighted sum of the predictions from all the weak learners.

- **Pros:**
  - **Good Performance:** It often provides high accuracy and is less prone to overfitting compared to a single complex model, especially if the weak learners are simple.
  - **Relatively Simple:** It is easier to implement and has fewer critical hyperparameters to tune compared to more advanced boosting models like XGBoost.
  - **Versatile:** It can be used with many different types of base learners.

- **Cons:**
  - **Sensitive to Noisy Data and Outliers:** The algorithm can be heavily affected by outliers because it will try hard to correctly predict these unusual data points.

- **Can be Slow:** Since it builds models sequentially, it cannot be parallelized and can be slower than algorithms like Random Forest.
- **May Be Outperformed:** It is often outperformed by more modern gradient boosting implementations like XGBoost or LightGBM on complex datasets.

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple, non-parametric algorithm used for both classification and regression. It is an instance-based learning method, meaning it does not learn a discriminative function from the training data but instead memorizes the training dataset. For regression, the prediction for a new data point is the average of the values of its 'k' nearest neighbors.

- **Pros:**
  - **Simple and Intuitive:** The logic is easy to understand and implement.
  - **No Training Phase:** The algorithm is fast during the "training" step because it only involves storing the dataset.
  - **Adapts to New Data:** It can easily incorporate new data points without needing to retrain the model.
- **Cons:**
  - **Computationally Expensive:** The prediction step can be slow and resource-intensive, especially with large datasets, as it needs to compute the distance to all training points.
  - **Sensitive to Feature Scale:** The algorithm is highly sensitive to the scale of the data and irrelevant features. Feature scaling (like normalization) is required.
  - **Requires Optimal 'k' Value:** The performance is heavily dependent on the choice of 'k' (the number of neighbors), which can be challenging to determine.

Support Vector Machine (SVM)

Support Vector Machine (SVM), when used for regression (as **Support Vector Regression or SVR**), works by finding a hyperplane that best fits the data while trying to keep as many data points as possible within a specified margin. Unlike other models that try to minimize error, SVR tries to fit the error within a certain threshold, making it robust to outliers that fall outside this margin.

- **Pros:**
  - **Effective in High-Dimensional Spaces:** It performs well even when there are more features than data points.
  - **Memory Efficient:** It uses a subset of training points (the support vectors) in the decision function, making it memory efficient.
  - **Versatile with Kernels:** The use of different kernel functions (e.g., linear, polynomial, RBF) allows it to model complex, non-linear relationships.
- **Cons:**
  - **Computationally Intensive:** Training can be slow on large datasets.
  - **Difficult Parameter Tuning:** Performance is highly dependent on the choice of the kernel and hyperparameters (like `C` and `gamma`), which can be difficult to tune optimally.
  - **Less Interpretable:** The model is considered a "black box" and is not as easy to interpret as Linear Regression.

MODEL PERFORMANCE

Linear Regression:

- RMSE : 21.29
- MAE : 14.85
- R – Squared : 0.8699

Random Forest:

- RMSE : 2.022595
- MAE : 0.557946
- R – Squared : 0.998837

Adaboost:

- RMSE : 26.462150
- MAE : 22.494619
- R – Squared : 0.800990

XGBoost:

- RMSE : 4.948093
- MAE : 2.008280
- R – Squared : 0.993042

CatBoost:

- RMSE : 5.901226
- MAE : 2.603820
- R – Squared : 0.990103

Decision Tree:

- RMSE : 6.84
- MAE : 2.01
- R – Squared : 0.9866

K Nearest Neighbour:

- RMSE : 9.57
- MAE : 3.43
- R – Squared : 0.9737

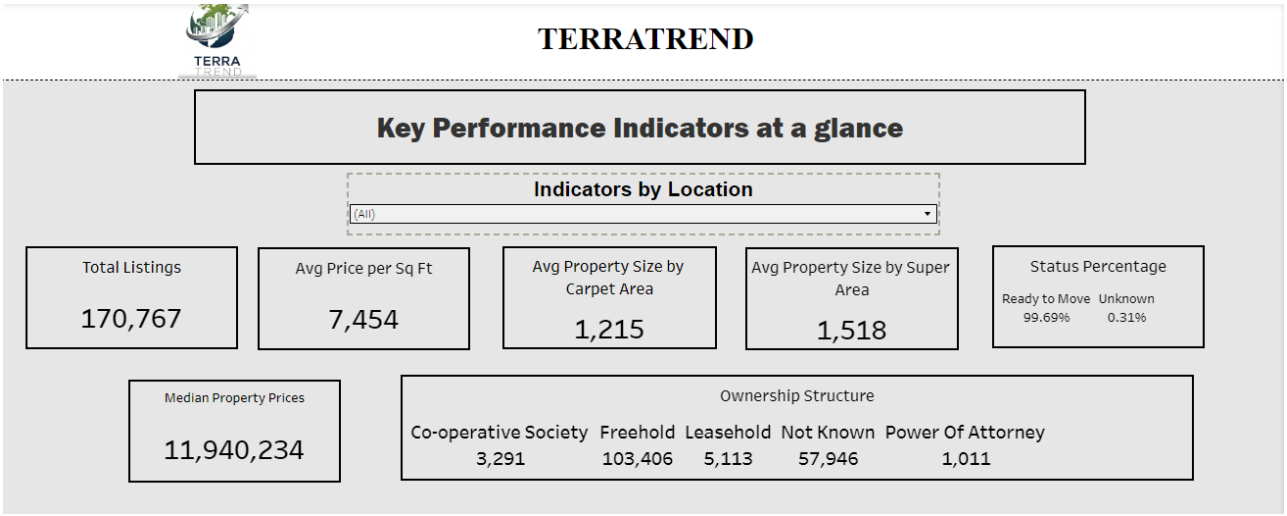
The **Random Forest** model is the clear winner. It outperformed all other models across every single evaluation metric.

Here’s a breakdown of why it's the best:

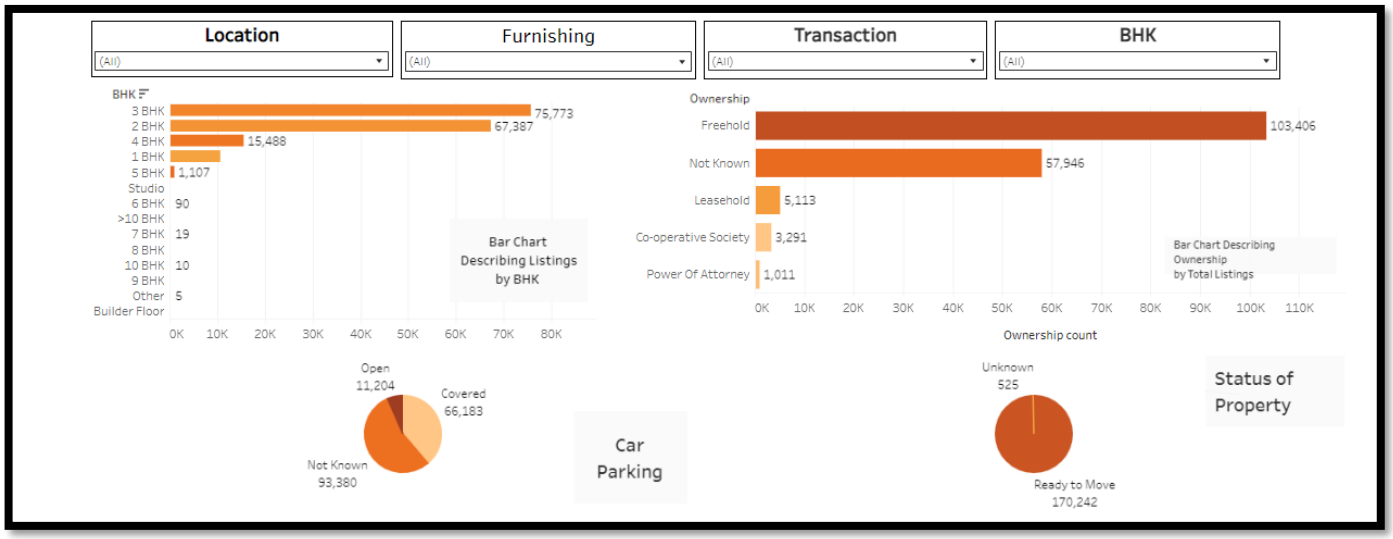
- **Highest R-Squared (0.9988):** This is the most important indicator. An R-squared value this close to 1 means your Random Forest model can explain **99.88%** of the variability in house prices. This is an exceptionally strong fit.
- **Lowest RMSE (2.02):** The Root Mean Squared Error is the lowest, indicating that the model's predictions have the smallest average error magnitude.
- **Lowest MAE (0.55):** The Mean Absolute Error is also the lowest, meaning on average, its predictions are closer to the actual house prices than any other model.

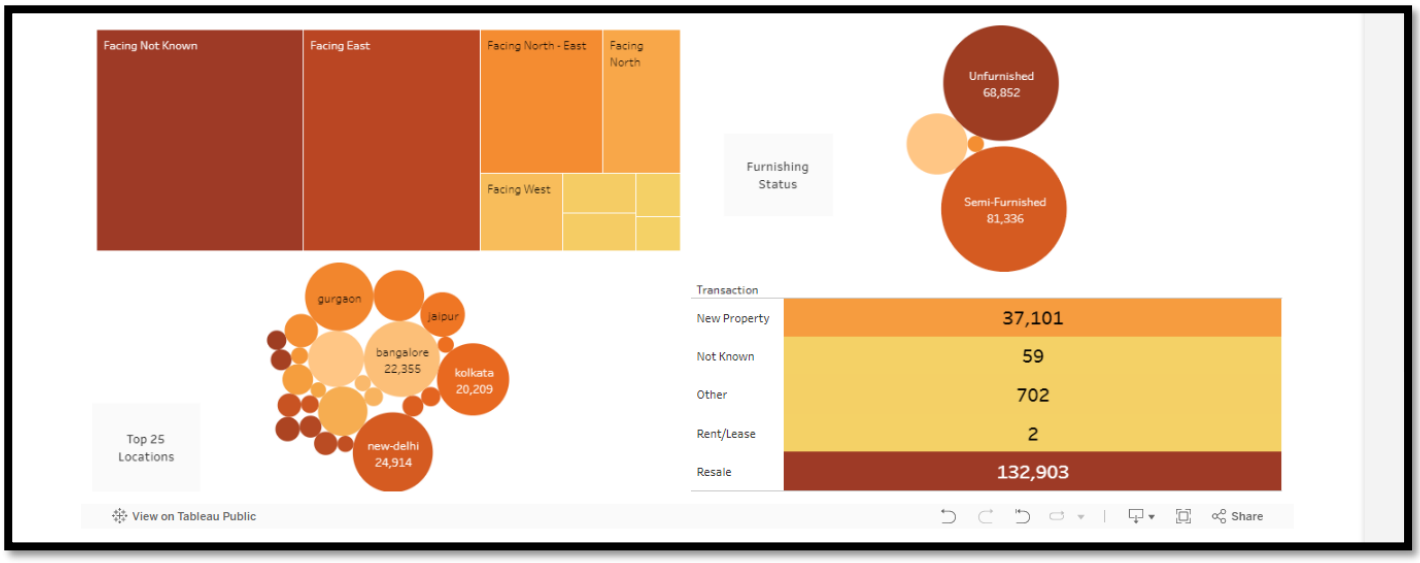
DATA VISUALIZATION AND REPRESENTATION

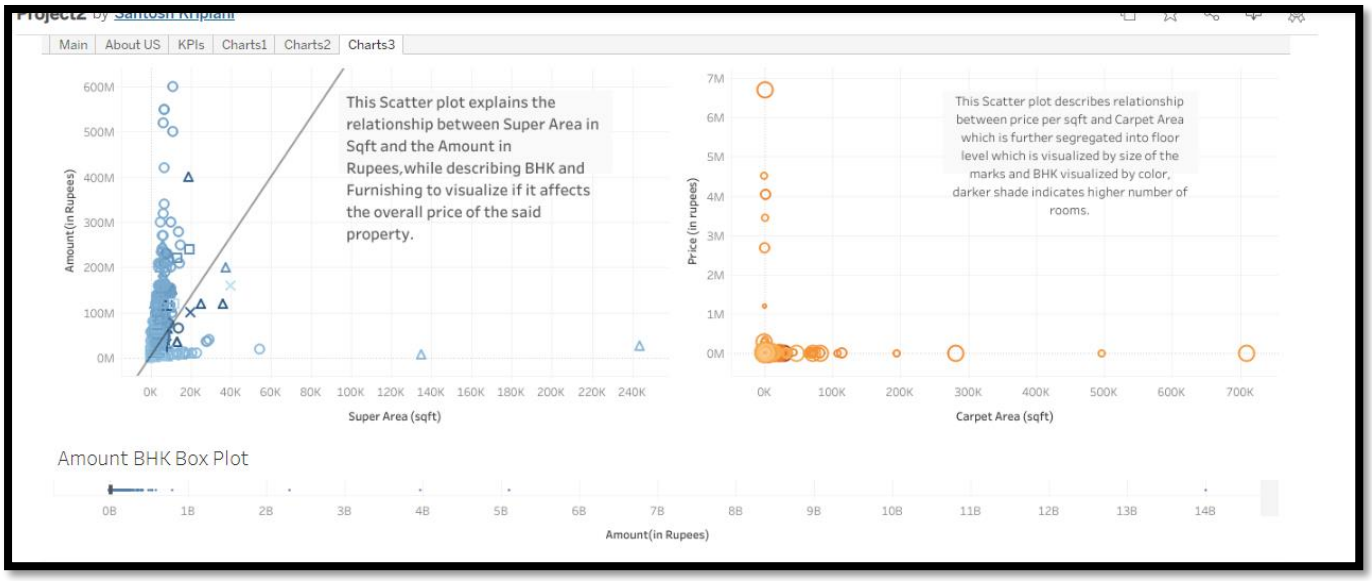
Our Locations Across India



Our Key Indicators at a Glance


Location
(All)
Furnishing
(All)
Transaction
(All)
BHK
(All)
BHK
3 BHK 75,773
2 BHK 67,387
4 BHK 15,488
1 BHK
5 BHK 1,107
Studio 90
6 BHK
>10 BHK 19
7 BHK
8 BHK
10 BHK 10
9 BHK
Other 5
Builder Floor
0K 10K 20K 30K 40K 50K 60K 70K 80K
Bar Chart Describing Listings by BHK
Ownership
Freehold 103,406
Not Known 57,946
Leasehold 5,113
Co-operative Society 3,291
Power Of Attorney 1,011
0K 10K 20K 30K 40K 50K 60K 70K 80K 90K 100K 110K
Ownership count
Bar Chart Describing Ownership by Total Listings
Open 11,204
Covered 66,183
Not Known 93,380
Car Parking
Unknown 525
Ready to Move 170,242
Status of Property


Facing Not Known
Facing East
Facing North - East
Facing North
Facing West
Furnishing Status
Unfurnished 68,852
Semi-Furnished 81,336
gurgaon
jaipur
bangalore 22,355
kolkata 20,209
new-delhi 24,914
Top 25 Locations
Transaction
New Property 37,101
Not Known 59
Other 702
Rent/Lease 2
Resale 132,903
View on Tableau Public
Share



Some charts showing various relationships.
https://public.tableau.com/app/profile/santosh.kriplani/viz/Project2_17544349064900/Main?publish=yes

Our Tableau Link

DEPLOYMENT

Deployment has been made on AWS EC2

Instance used: c5a large (paid instance)

Ram: 4 GB

CPU – 4 virtual cores

Burstable – No (100% of CPU can be utilized)

Elastic IP – Yes (hence fixed public IP)

Zone – Mumbai

FUTURE SCOPE

- **Advanced Algorithms** : Deep Learning (Example Neural Networks) for model improvement and optimization

- **Integration with real time data** : incorporate real time market trends.

- **Explainable AI** : Integrate tools like SHAP or LIME to explain model predictions.

- **Natural Language description Generation using LLMs** : Fine tune LLMs like T5, LLaMA, on your dataset to automatically generate detailed and well written property descriptions from structured data.