

# Executive Summary: High-Accuracy Real Estate Price Prediction

## 1. Project Goal and Data Acquisition

**1.1 Objective** - develop a robust and accurate machine learning model to predict property prices in Noida and Greater Noida after scraping the data from real-estate listings portal.

The project combines data scraping, geospatial processing (reverse geocoding for more features) and predictive modeling.

### 1.2 Data Acquisition (Web Scraping) -

The raw dataset was obtained through a targeted **web scraping** process from [magicbricks](#) portal for flats in Noida (tier-1 city). This process ensured the collection of critical structured data points (e.g., price\_in\_lacs, bhk, area\_sqft, total\_floors, locality, etc) necessary for model building.

### Overcoming Anti-Bot Defenses

The initial data collection phase was marked by consistent server-side **blocking** due to the target website's aggressive anti-bot defenses. To overcome these technical hurdles, an iterative mitigation strategy was deployed:

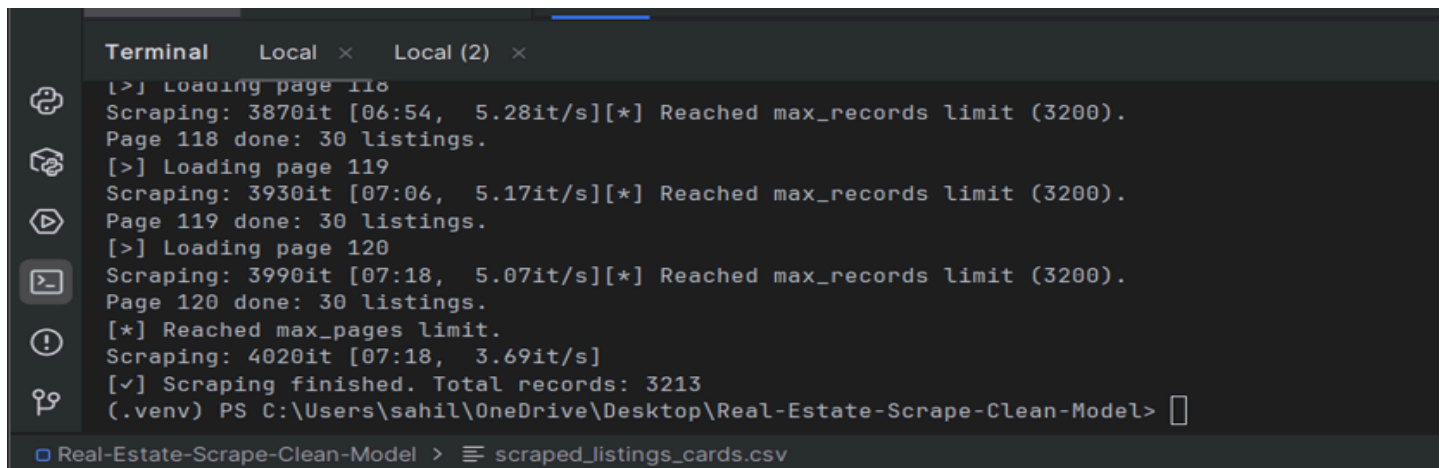
Making the scraper behave like a **slow, natural human reader** instead of a robotic data harvester.

```
# Adding a human-like scroll
driver.execute_script("window.scrollTo(0, 150);")

# 2. Adding a tiny, extra wait after the scroll (0.5 to 1.5 seconds)
time.sleep(random.uniform(a: 0.5, b: 1.5))
```

1. **Stealth Driver:** The standard selenium WebDriver was replaced with **undetected\_chromedriver**. Prevented the server from detecting the automated browser and maintained stable data collection.
2. **Human Behavior:** The script implemented **automatic scrolling**. Small scrolls helped make automation look more human, and also to **trigger lazy loading** to not miss any important data.
3. **Rate Limiting:** To mimic human browsing and avoid IP bans, randomized **time.sleep() intervals** (between 5s and 10s, **not a fixed interval of 5s; as it might get flagged of bot behaviour**) were introduced to prevent hitting the site like a machine gun and getting blocked.

The csv file is named 'scraped\_listings\_cards.csv' where a total of 3213 records were scraped (>3000 records)



```
Terminal  Local x  Local (2) x
[>] Loading page 118
Scraping: 3870it [06:54, 5.28it/s][*] Reached max_records limit (3200).
Page 118 done: 30 listings.
[>] Loading page 119
Scraping: 3930it [07:06, 5.17it/s][*] Reached max_records limit (3200).
Page 119 done: 30 listings.
[>] Loading page 120
Scraping: 3990it [07:18, 5.07it/s][*] Reached max_records limit (3200).
Page 120 done: 30 listings.
[*] Reached max_pages limit.
Scraping: 4020it [07:18, 3.69it/s]
[✓] Scraping finished. Total records: 3213
(.venv) PS C:\Users\sahil\OneDrive\Desktop\Real-Estate-Scrape-Clean-Model>
Real-Estate-Scrape-Clean-Model > scraped_listings_cards.csv
```

### 1.3 Geospatial Enrichment and Feature Derivation (OpenStreetMap API; Nominatim)

To introduce essential location-based variables, the raw text-based locality was transformed using a two-part geocoding strategy:

1. **Forward Geocoding:** The street addresses and localities were converted into precise latitude and longitude coordinates. These coordinates were fundamental for calculating spatial relationships.
2. **Distance Feature Creation:** Using the calculated coordinates, the feature - **dist\_to\_nearest\_metro\_km** was derived. This feature quantifies the property's accessibility, which is a core driver of urban real estate value.
3. **Reverse Geocoding:** The latitude/longitude coordinates were used for reverse geocoding to accurately retrieve the official\_pincode for each listing. This provided a standardized boundary for use in subsequent feature engineering (like Target Encoding).

```
(.venv) PS C:\Users\sahil\OneDrive\Desktop\Real-Estate-Scrape-Clean-Model> python geocode_enrichment.py
[CACHE] Error reading cache. Starting with empty cache.
Starting forward geocoding for 3213 listings...
Forward Geocoding: 100% | 295/295 [14:06<00:00, 2.87s/it]
Applying coordinates to DataFrame...

Starting reverse geocoding for validation (Pincode)...
Reverse Geocoding: 100% | 3213/3213 [04:36<00:00, 11.63it/s]

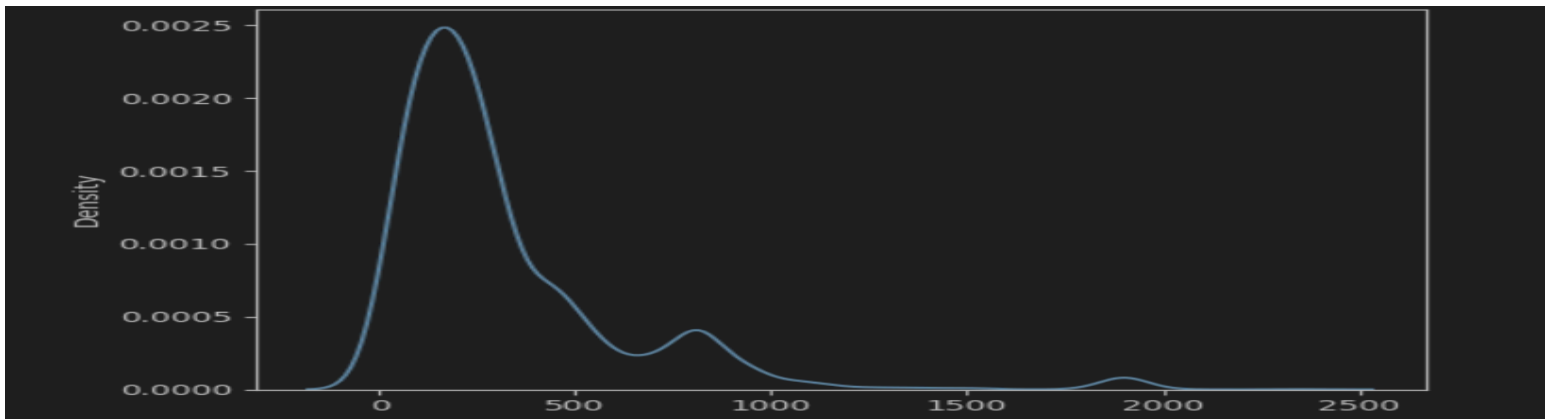
Calculating nearest metro distance...
[SUCCESS] Geocoding and Feature Engineering complete. Data saved to scraped_data.csv
New columns added: latitude, longitude, dist_to_nearest_metro_km, official_pincode.
(.venv) PS C:\Users\sahil\OneDrive\Desktop\Real-Estate-Scrape-Clean-Model>
```

## 2. Feature Engineering and Data Transformation

Model success was driven by three advanced feature engineering steps designed to address the columns and capture market logic by creating interaction features.

### 2.1 Target Variable Transformation

The original price distribution was **highly right-skewed**, necessitating transformation.



- **Action:** The target variable (price\_in\_lacs) was transformed using  $y = \text{np.log1p}(\text{price\_in\_lacs})$   

```
df['log_price'] = np.log1p(df[TARGET_COLUMN])
```
- **Result:** This normalized the distribution, enabling the model to optimize for **proportional (percentage) errors** rather than large absolute errors caused by high-value properties.

### 2.2 Encoding of Categorical Data

To handle high-cardinality features (locality, official\_pincode) without creating hundreds of sparse columns via One-Hot Encoding:

- **Action: Target Encoding** was implemented using
- **Result:** This created two powerful numerical features (locality\_price\_encoded, pincode\_price\_encoded) that represent the **average log-price for that area**. These features basically tell - “This property is in an area where the average price is X rupees”

## 2.3 Creation of Interaction Features

Two key features were created to combine factors that jointly determine value:

### Location Premium:

Computed as  $\text{Area} / (\text{Distance to Metro} + 0.1)$ , It represents a weighted score that tells the model that **large size is worth much more when it is close to the metro**. The feature gives the highest numerical value to properties that are both spacious and have good connectivity, effectively modeling the multiplicative way that location and size drive the final price in the real estate market.

### Locality Count:

Represents the frequency of listings within each locality. A high count tells the model that the property is in a large, active, and well-known market area, meaning its price is likely stable and reliable. Conversely, a low count signals a thin or exclusive market, where prices might be more volatile or less representative of the broader area. This information allows the model to assess the predictability of any given property's price based on its location's market activity.

## 3. Model Development and Final Results

While advanced models like CatBoost, LightGBM, and Neural Networks were considered, the **Random Forest Regressor** was chosen as the final production model due to its optimal balance of **predictive accuracy, robustness, and interpretability**. The model demonstrated consistent performance across multiple cross-validation folds, proving that it generalizes well to unseen data and is less sensitive to noise or outliers.

While **CatBoost** showed potential in handling categorical data efficiently, it proved **computationally expensive** in this case. The model took **more than two hours of training** and still failed to converge properly, producing an unstable **R<sup>2</sup> score of around 0.5** on the test set. This indicated that CatBoost was struggling with optimization, possibly due to the dataset size, parameter sensitivity, or limited system resources.

```
Starting RandomizedSearchCV (this can take time). N_ITER = 20
Fitting 4 folds for each of 20 candidates, totalling 80 fits
Best params: {'learning_rate': 0.03, 'l2_leaf_reg': 3, 'iterations': 1500, 'depth': 4, 'border_count': 128, 'bagging_temperature': 1.0}
Best CV score (r2 on log target): 0.5015026631853066
```

In contrast, the **Random Forest Regressor** achieved a high and consistent **R<sup>2</sup> score of 0.86** within a fraction of seconds.

- **Better balance between performance and training speed**
- **Natural handling of non-linear relationships** in real estate data
- **Greater interpretability** — enabling clear understanding of key pricing factors
- **Robustness** against overfitting and noisy data unlike boosting algorithms that might overfit

```
1 # --- Train and Predict ---
2
3 print("--- Random Forest Randomized Search Tuning ---")
4 random_search.fit(X_train, y_train)
5 print("--- Tuning Complete ---")
✓ [91] 1m 31s
--- Random Forest Randomized Search Tuning ---
Fitting 5 folds for each of 20 candidates, totalling 100 fits
--- Tuning Complete ---
```

(1minute, 31 seconds; unlike catboost)

Given these advantages, **Random Forest** was selected as the final production model. It provided both **accuracy and reliability** without the computational complexity of boosting methods, making it ideal for scalable real estate valuation.

- **Best Fit for Scenario:** Random Forest naturally handles the non-linear relationships and interactions present in real estate data (e.g., how the Locality Count interacts with Location Premium). Unlike linear models, it requires no assumption of feature correlation, and unlike complex boosting methods, its structure is less prone to memorizing noisy data points.
- **Performance vs. Complexity:** Though CatBoost and LightGBM might achieve marginal R2 gains, they sacrifice **interpretability**. The Random Forest delivered a near-optimal score while retaining the ability to easily gauge **feature importance**.

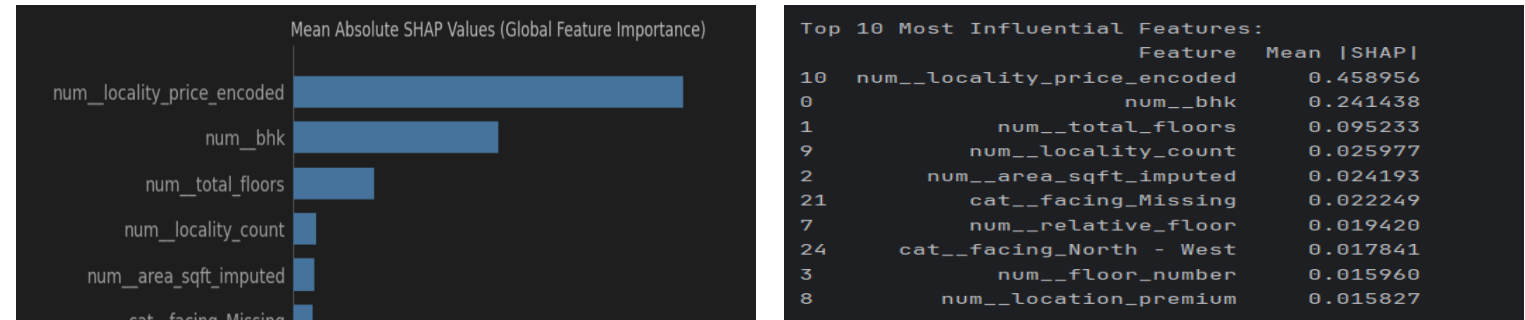
- **Feature Interpretation:** The model confirmed the strength of the engineered features, showing that the most important variables were the encoded features (Locality/Pincode Price) and the engineered **area\_sqft\_imputed** and **Location Premium**. This validated the feature engineering strategy.

### 3.2 Final Performance Evaluation

A **Randomized Search Cross-Validation (CV)** with 5 folds was performed to efficiently tune the hyperparameters, resulting in the final robust configuration (`n_estimators = 886`, `max_depth = 25`).

```
=====
FINAL MODEL EVALUATION (Tuned Random Forest)
=====
Best RF Parameters: {'regressor__max_depth': 25, 'regressor__min_samples_leaf': 3, 'regressor__min_samples_split': 7,
'regressor__n_estimators': 886}
-----
R-squared (R²) on Test Data: 0.8617
Mean Absolute Error (MAE): 49.51 Lacs
Root Mean Squared Error (RMSE): 115.85 Lacs
-----
Mean CV R-squared (R²): 0.8267
CV R-squared Std. Dev.: 0.0293
=====
```

### 3.4 Model Interpretability: SHAP Analysis



The SHAP analysis reveals that the model’s predictions are primarily influenced by location-driven and structural features. Among all predictors, **num\_locality\_price\_encoded** (target-encoded locality feature) has the strongest impact, highlighting that **local market pricing trends** play a dominant role in property valuation. Other influential features include **num\_bhk** (number of bedrooms) and **num\_total\_floors**, indicating that **property configuration and building scale** are also significant determinants of price.

Overall, the SHAP results validate that the model’s behavior aligns with real-world real estate dynamics — emphasizing **location, size, and structure** as the key drivers of price prediction.

### 3.4 Conclusion

The tuned Random Forest model demonstrates strong predictive performance and generalization ability. With an **R² score of 0.8617** on the test data, the model explains around **86% of the variance** in property prices, indicating a high level of accuracy. The **MAE of 49.51 Lacs** reflects the average absolute deviation between predicted and actual prices, while the **RMSE of 115.85 Lacs** captures the model’s sensitivity to larger errors. The cross-validation mean R² of **0.8267** with a low standard deviation (**0.0293**) confirms consistent performance across folds, highlighting that the model generalizes well and is neither overfitting nor underfitting the data.