IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131126

Predicting Residential Property Prices in Pune, India: A Comparison of Linear, Decision-Tree and Random Forest Models, with an Underfitting and Overfitting Analysis

Gauri Dnyaneshwar Gunwant[1], Sonali Garad[2], Shrawani Avinash Wavre[3],

Prof. Jaydeep Karpe[4]

MCA – Semester III (2026–2027), Adsul's Technical Campus, Chas, Ahilyanagar[1-3]

Guide, MCA – Semester III (2026–2027), Adsul's Technical Campus, Chas, Ahilyanagar[4]

Abstract: Buying or selling a house is hard when prices are not clearly known, especially in fast-growing Indian cities. In this study we build machine-learning models that predict house prices in Pune, Maharashtra. We use a dataset of 4,224 listings from 20 localities. We compare four models on the same 80/20 train–test split: linear regression, a shallow decision tree, a deep decision tree and a random forest. We also build a fifth model, a 400-tree random forest that uses five inputs (locality, property type, area, BHK and year built). This model runs behind an interactive R Shiny dashboard that gives a price estimate instantly. We measure each model with R^2, mean absolute error (MAE) and root mean squared error (RMSE), and we compare training scores with testing scores to find underfitting and overfitting. The shallow tree underfits (test $R^2 = 0.7598$). Linear regression does well but still leaves an error (test RMSE = 26.24 Lakhs). The deep tree and the random forest are almost perfect on test data (test R^2 of 0.998 or more). Built-up area (about 82% of importance) and locality (about 17%) explain almost all of the price. When we looked closely at the data, we found that every price equals area multiplied by a fixed rate for each locality. This explains the near-perfect scores, and we discuss what it means for real market data.

Keywords: house price prediction, random forest, decision tree, overfitting, underfitting, hedonic pricing, Pune, R Shiny

1. INTRODUCTION

For most families, a house is the biggest purchase of their lives. In India, cities are growing, incomes are rising and IT jobs are increasing, and because of this Pune has become one of the busiest housing markets in the country. Prices in Pune change a lot from one locality to another. Older central areas such as Koregaon Park and Kothrud cost far more per square foot than outer areas such as Hadapsar or Undri. A buyer usually has only a broker's quote and a few similar listings to decide whether a price is fair.

Machine learning gives another way to look at this problem. If we have many past listings, a model can learn how location, area and property type affect the price, and then estimate the price of a house it has not seen before. The main risk is that the model may learn the wrong thing. A model that is too simple will underfit, which means it misses real patterns in the data. A model that is too complex will overfit, which means it memorises its training data and does badly on new houses. To find out which one has happened, we have to compare the results on training data with the results on test data that the model has never seen.

1.1 Objectives

- To build models that predict house prices in Pune from locality, property type, area, BHK and year built.
- To compare linear regression, a shallow decision tree, a deep decision tree and a random forest, and to find which ones underfit or overfit.
- To find out which property features matter most for the predicted price.
- To provide the best model as an easy dashboard with graphs that anyone can use.

1.2 Research questions

We try to answer three questions. First, how well can the price be predicted from a few features that anyone can observe? Second, how does the complexity of a model change the balance between underfitting and overfitting on this data? Third, which features drive the predictions, and does this match what we expect from the property market?

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131126

1.3 Layout of the paper

Section 2 reviews earlier work. Section 3 describes the data and how we prepared it. Section 4 explains the models and how we tested them. Section 5 gives the results. Section 6 discusses them and points out an important problem with the dataset. Section 7 gives the conclusion, the limitations and ideas for future work.

2. LITERATURE REVIEW

2.1 Hedonic pricing

The basic economic idea behind house valuation is hedonic pricing (Rosen, 1974). It treats a house as a set of features, such as size, location and quality, and each feature has its own implicit price. A well-known example is the Boston housing study of Harrison and Rubinfeld (1978), which used linear regression to estimate these prices. Linear models are still popular because they are easy to explain. The coefficient of floor area is simply the average price of one extra square foot when everything else stays the same.

However, linear models assume that every feature adds to the price in the same way everywhere. Real markets do not work like this. An extra square foot is worth much more in an expensive locality than in a cheaper one. This is an interaction between area and location, and a simple linear model cannot capture it unless the analyst adds interaction terms by hand.

2.2 Decision trees and forests

Regression trees (Breiman, Friedman, Olshen and Stone, 1984) split the data into groups again and again and predict one value for each group. They find non-linear patterns and interactions on their own and are easy to draw. But one deep tree is unstable and overfits easily, because it can keep splitting until each group has only a few houses. Random forests (Breiman, 2001) solve this by averaging many trees. Each tree is built on a random sample of the data and looks at a random set of features at each split, which lowers the variance and keeps the bias low. In R, forests can be built with the randomForest package (Liaw and Wiener, 2002) and the ranger package (Wright and Ziegler, 2017).

2.3 Underfitting and overfitting

The bias–variance trade-off is a main idea in statistical learning (Hastie, Tibshirani and Friedman, 2009). A model with high bias underfits, so its error is high on both training and test data. A model with high variance overfits, so its training error is very low but its test error is higher. The usual way to check this is to compare training and test scores, preferably with cross-validation or, for forests, out-of-bag estimates (Kuhn and Johnson, 2013). We use this check as the main idea of our project.

2.4 Research gap

Many house-price studies report only one best model and one accuracy number. Few of them place several models side by side so that underfitting and overfitting are easy to see, and even fewer look at Pune. Our project compares four model types on Pune listings and shows the results in a dashboard with graphs.

3. DATA

3.1 Source and structure

The dataset (pune_housing_data_locality_prices.csv) has 4,224 house listings for Pune, with 24 columns for each listing. The columns cover the identity of the listing, the location (state, city, locality), the property (type, BHK, area, year built, age, floor, furnishing), the surroundings (nearby schools and hospitals, public transport), amenities, owner and availability, and the price (price in Lakhs and price per square foot). One Lakh is 100,000 rupees. There are no missing values. The listings come from 20 localities, with 180 to 248 listings in each, so no single locality dominates the data.

3.2 Variables used

Our R script uses only six columns: the target Price_in_Lakhs and five inputs. We chose these inputs because a buyer can state them without any inside knowledge of the listing.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131126

Table 1. Variables used in the models.

Variable	Type	Description
Price_in_Lakhs	Numeric (target)	Listing price in Lakhs of rupees
Locality	Categorical (20)	Area of Pune, e.g. Aundh, Baner, Kothrud
Property_Type	Categorical (3)	Apartment, Independent House or Villa
Size_in_SqFt	Numeric	Built-up area in square feet
BHK	Numeric (1–5)	Number of bedrooms, hall and kitchen
Year_Built	Numeric	Year of construction (1990–2023)

3.3 Basic statistics

Prices go from 32.58 to 692.72 Lakhs. The mean is 257.39 and the median is 245.61 Lakhs (standard deviation 135.33). Area goes from 500 to 4,999 sq ft, with a mean of about 2,728 sq ft. The three property types are almost equally common (1,441 villas, 1,428 independent houses and 1,355 apartments) and have similar average prices (261.6, 253.4 and 257.3 Lakhs).

Price is strongly related to area ($r = 0.902$), but almost not related to BHK ($r = -0.013$) or year built ($r = 0.008$). Figure 1 plots price against area. The points form a fan-shaped band: the price grows with size, but the slope is different in different localities. This is the interaction we described in Section 2.1.

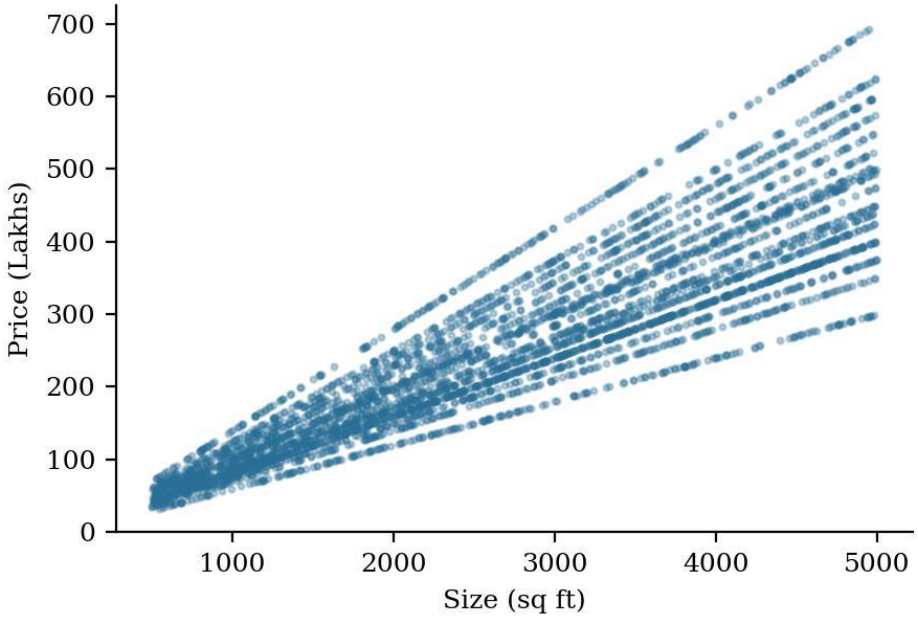


Fig. 1. Price versus built-up area for all 4,224 listings.

Locality has a big effect on price. Figure 2 ranks the localities by average price. Koregaon Park is the most expensive (mean 361.9 Lakhs), followed by Kothrud (332), Model Colony (329.9) and Baner (307.4). The cheapest are Hadapsar (159.5), Undri (191.4) and Nibm Road (200.8). So the most expensive locality costs more than twice as much as the cheapest one on average.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131126

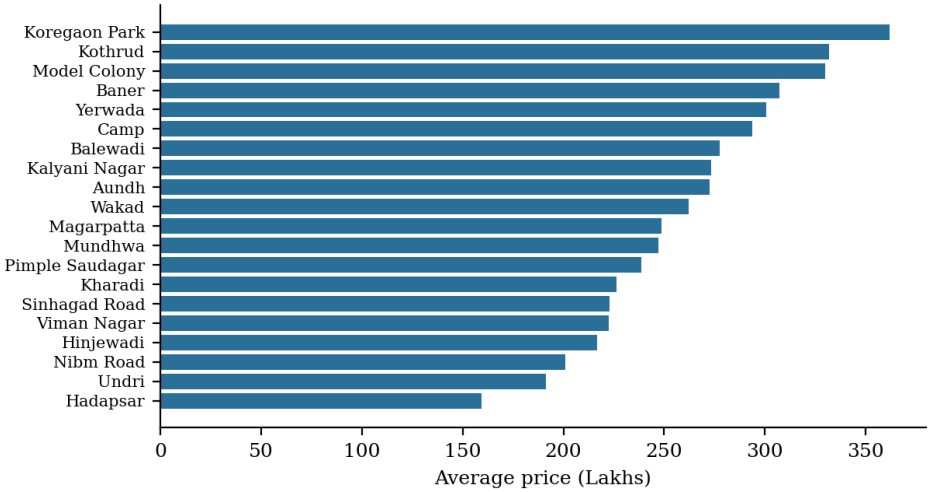


Fig. 2. Average listing price by locality (Lakhs).

3.4 Preprocessing

Before modelling, the R script does these steps: (i) removes rows with no price; (ii) converts the inputs to numbers where needed; (iii) fills any missing number with the median of its column; (iv) drops incomplete rows; and (v) converts Locality and Property_Type to factors. This dataset has no missing values, so steps (i) to (iv) change nothing here. They are still useful if the file is later replaced with a messier one.

4. METHODOLOGY

4.1 Experiment design

To compare the models fairly, we trained all four on the same data. The script splits the data 80/20 with a fixed random seed (42). About 3,380 listings (80%) are used for training, and about 844 listings (20%) are kept aside to test the models on unseen houses. The four comparison models use only two inputs, Locality and Size_in_SqFt. This way we can see the effect of model complexity alone, without adding more features. The script also groups all but the 50 most common localities into one group called “Other”, but since this file has only 20 localities, no grouping happens.

4.2 Models

- Linear regression. An ordinary least squares model, Price ~ Locality + Size. Each locality changes the starting price, and area has one common slope. It is our simple baseline.
- Shallow decision tree. A regression tree (rpart, ANOVA method) with a depth of only 2 and at least 30 listings needed to make a split. It is kept simple on purpose, so we expect it to underfit.
- Deep decision tree. A regression tree with a maximum depth of 20, a minimum split size of 2 and a complexity parameter of 0.0001. It can grow almost without limit, so it has the highest risk of overfitting.
- Random forest. A forest of 200 trees (randomForest package) that tries 2 inputs at each split. Averaging many trees reduces the variance of single trees.

The model behind the dashboard is a separate random forest (ranger package, 400 trees, mtry = 3). It uses all five inputs: Locality, Property_Type, Size_in_SqFt, BHK and Year_Built. It is trained on all listings and saved as pune_price_model.rds, so it loads quickly the next time. We also record the importance of each input for the importance chart.

4.3 Evaluation measures

We calculate three measures on both the training and the test data. R^2 shows how much of the variation in price the model explains. MAE is the average size of the error in Lakhs and is easy to understand. RMSE is the square root of the average squared error, also in Lakhs, and it punishes big errors more. If y is the actual price and $\hat{y}$ is the predicted price:

$$R^2 = 1 - \Sigma(y - \hat{y})^2 / \Sigma(y - \bar{y})^2$$

We say a model underfits when the error is high on both training and test data. We say it overfits when the test error is much worse than the training error.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131126

4.4 Tools and reproducibility

We built the project in R (version 4.6.1) with the data.table, dplyr, tidyr, ranger, randomForest, rpart, caret and ggplot2 packages, and the Shiny package for the dashboard. We ran R from Visual Studio Code, and the setup commands are saved in the project folder. R was not available when we prepared the results given below, so we repeated the same steps in Python with scikit-learn. We used the same inputs, the same 80/20 split, the same seed and similar settings (tree depth, minimum split size, 200 trees for the comparison forest and 400 for the full forest). The numbers from the R script may be slightly different in the last decimals, because the two tools use different random number generators and split rules. The main conclusions do not change.

5. RESULTS

5.1 Model comparison

Table 2 gives the training and testing results of the four comparison models. Figures 3 and 4 show the same results as graphs, as in the dashboard.

Table 2. Training and testing results (RMSE and MAE in Lakhs).

Model	Train R^2	Test R^2	Train RMSE	Test RMSE	Test MAE
Linear Regression	0.9647	0.9638	25.31	26.24	18.67
Shallow Decision Tree	0.7685	0.7598	64.78	67.59	51.19
Deep Decision Tree	0.9992	0.9986	3.88	5.13	3.09
Random Forest	1.0000	0.9998	0.66	1.78	1.15

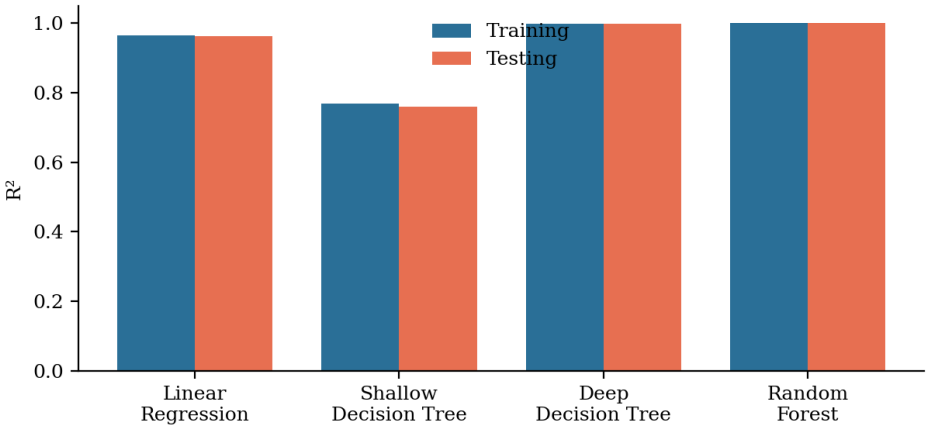


Fig. 3. Training versus testing R^2 by model.

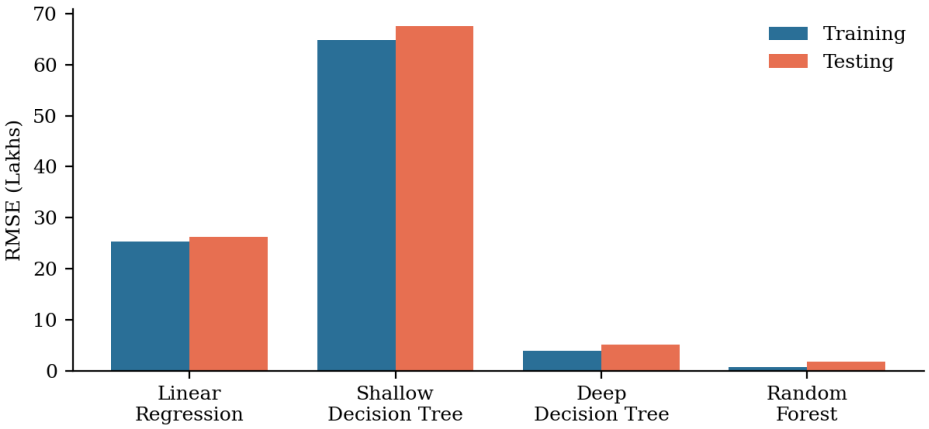


Fig. 4. Training versus testing RMSE by model (Lakhs).

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131126

5.2 Underfitting

The shallow decision tree clearly underfits. With only two levels it can make at most four groups of houses, so it cannot handle twenty localities and a changing area effect together. Its R^2 is 0.7685 on training data and 0.7598 on test data, and its test RMSE of 67.59 Lakhs is about 2.6 times the RMSE of linear regression. The training and test scores are almost the same, which tells us the problem is high bias and not high variance. The model is equally wrong on data it has seen and data it has not seen.

Linear regression is much better (test $R^2 = 0.9638$) and the gap between training and test is very small, so it does not overfit. But it still leaves an error of about 26.24 Lakhs RMSE. This is a smaller kind of underfitting. Linear regression uses one common slope for area, but the price of an extra square foot is different in each locality.

5.3 Overfitting

The deep tree and the random forest have the biggest gap between training and test results when we compare them in relative terms. The RMSE of the deep tree goes from 3.88 on training data to 5.13 on test data, which is about 32% higher. The RMSE of the random forest goes from 0.66 to 1.78, almost three times higher. This is a mild case of overfitting, because both models fit the training data a little better than new data.

But we should look at this gap in the right context. The test errors are very small in real terms. The random forest is off by only 1.15 Lakhs on average, while the mean price is 257.39 Lakhs, and its test R^2 (0.9998) is almost equal to its training R^2. So the overfitting is only statistical and does not matter much in practice. The random forest is the most accurate model on unseen data, and it is better than the single deep tree on every test measure. This is what we expect when many trees are averaged.

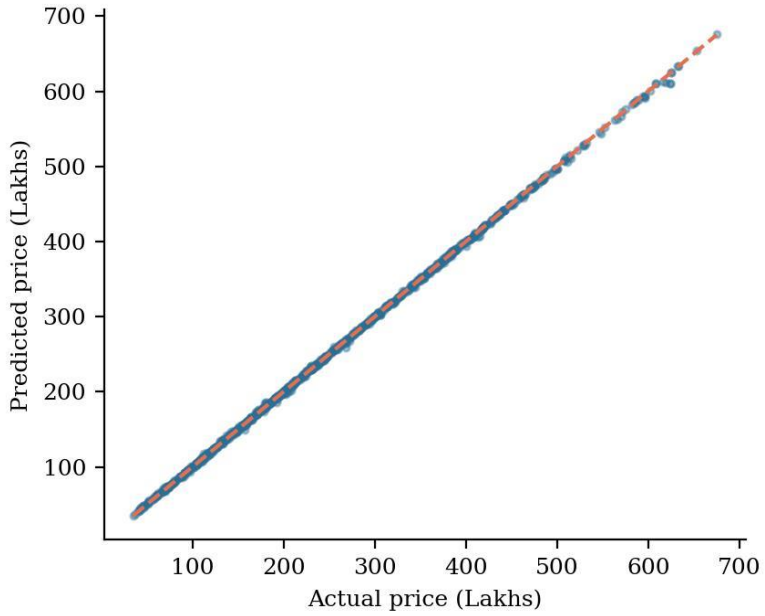


Fig. 5. Random forest: actual versus predicted price on test data. Points on the dashed line are perfect predictions.

5.4 The five-input model used in the dashboard

The forest used in the dashboard also uses Property_Type, BHK and Year_Built. On the 20% test data it gives $R^2 = 0.9986$, MAE = 3.42 Lakhs and RMSE = 5.14 Lakhs. Its out-of-bag R^2 is 0.9988, which is very close to the test value, so the estimate looks reliable. Its test error is a little higher than that of the two-input forest (RMSE 5.14 against 1.78 Lakhs). The reason is that BHK, year built and property type carry no real information about the price here. They only give the trees more useless choices when splitting.

As an example of the dashboard output, the setup notes use a 3-BHK apartment of 1,200 sq ft in Kalyani Nagar, built in 2020. The model estimates its price at about 112.34 Lakhs. This is far below the Kalyani Nagar average of 273.19 Lakhs, which is reasonable because this flat is much smaller than a typical listing in that area.

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131126

5.5 Feature importance

Figure 6 shows the importance of each input in the forest. The 20 locality columns are added together into one Locality bar. Built-up area gives about 82.2% of the total importance and locality gives about 16.8%. Year built (0.5%), BHK (0.3%) and property type (0.2%) together give less than one percent. This agrees with the correlations in Section 3.3.

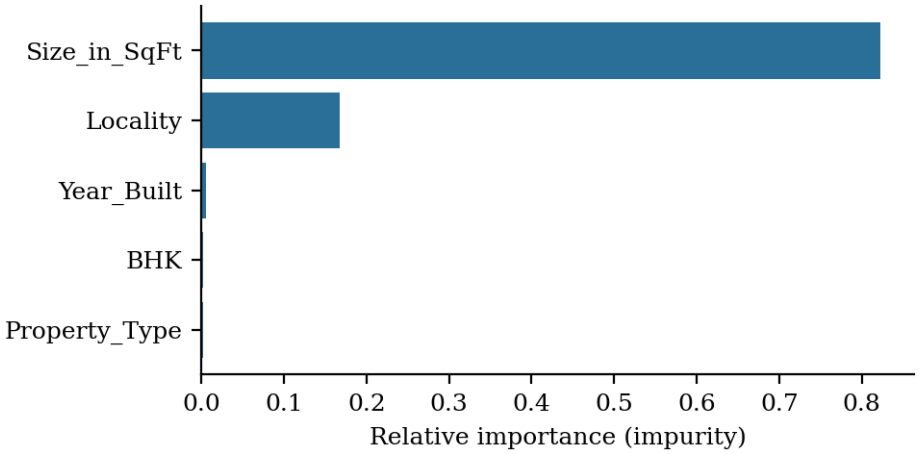


Fig. 6. Combined random forest feature importance.

6. DISCUSSION

6.1 What the comparison shows

The results show the bias–variance trade-off clearly. The shallow tree has high bias. The deep tree has lower bias but higher variance. The random forest keeps the low bias of deep trees and removes much of the variance by averaging. Linear regression sits in between, with low variance but some bias because of its linear form. The path from the shallow tree to the forest shows that the complexity of a model must match the structure of the data, and that comparing training and test scores tells us more than any single score.

6.2 An important problem with the dataset

Test scores above 0.998 are very unusual for real house data. In real data, things like condition, view, bargaining and timing leave a large part of the price unexplained. When we checked the file, we found the reason. Every locality has exactly one value of Price_per_SqFt (for example 6,000 in Hadapsar, 10,100 in Kalyani Nagar and 14,000 in Koregaon Park). For all 4,224 listings, Price_in_Lakhs is equal to Size_in_SqFt multiplied by that rate and divided by 100,000, with zero difference. So the price in this dataset is fully decided by only two things, locality and area. This is also why BHK, year built and property type are of no use: they were never used to create the price.

This has two results. First, the models are not discovering a hidden market rule. They are finding a formula, and that is why the flexible models (deep tree and forest) are almost perfect while the linear model is not. Second, these accuracy numbers should not be used to say how well the same method would work on real Pune sales. With real data we would expect a lower R^2, a bigger gap between training and test scores, and a bigger role for age, amenities and condition. The comparison of the model types is still useful to show underfitting and overfitting, but the exact numbers belong to this dataset only.

6.3 Use of the dashboard

The Shiny dashboard shows the analysis only through graphs, with no console output or tables. The user selects locality, property type, area, BHK and year built in a side panel, and seven graphs change immediately. They show the predicted price against the locality and city averages, a price-versus-size curve for the chosen locality, training versus testing R^2 and RMSE, actual versus predicted prices, feature importance, and the top 20 localities with the selected one highlighted. A buyer can easily see, for example, how much 200 extra sq ft changes the price in Baner compared with Hadapsar.

6.4 Limitations

- The way the prices were created (Section 6.2) limits how far the results can be trusted for real data. They should be tested again on real sale or listing prices.
- The dataset has no dates, so trends, inflation and seasons cannot be studied.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

DOI: 10.17148/IARJSET.2026.131126

- Only five inputs are used. Floor, amenities, furnishing, parking, facing and transport access are in the file but were not used. With real prices they may matter.
- We used only one 80/20 split. Repeated cross-validation would give a range for each score.
- The dashboard model was trained on all listings, so its fit on the dashboard should not be read as test accuracy.
- The results here were reproduced in Python and not in R, so small differences from the R output are possible.

7. CONCLUSION AND FUTURE WORK

In this project we compared linear regression, a shallow tree, a deep tree and a random forest for predicting house prices in Pune, and we gave the best model in an interactive dashboard. The shallow tree underfits, with equally poor training and test results. Linear regression is stable but limited by its linear form. The deep tree and the random forest reach almost perfect test accuracy with only slight overfitting, and the random forest is the best (test $R^2 = 0.9998$). Area and locality explain almost everything. The most important point is that in this file the price is simply area multiplied by a fixed rate for each locality. This explains the very high accuracy, and it is a warning not to expect the same results on real markets. In future work, we plan to do five things. (1) Test the method on real prices, for example from registered sale deeds or listings that have dates. (2) Use repeated k-fold cross-validation and tune settings such as mtry and tree depth instead of fixing them. (3) Add the other columns, such as amenities, floor, furnishing and accessibility, and try gradient boosting methods such as XGBoost. (4) Add prediction intervals, for example with quantile regression forests, so that the dashboard shows a price range and not a single number. (5) Add time to study price trends, and let the dashboard update when new listings arrive.

REFERENCES

[1]. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.
[2]. Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and Regression Trees. Wadsworth.
[3]. Harrison, D., & Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. Journal of Environmental Economics and Management, 5(1), 81–102.
[4]. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning (2nd ed.). Springer.
[5]. Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.
[6]. Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R News, 2(3), 18–22.
[7]. Rosen, S. (1974). Hedonic prices and implicit markets: Product differentiation in pure competition. Journal of Political Economy, 82(1), 34–55.
[8]. Wright, M. N., & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. Journal of Statistical Software, 77(1), 1–17.

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License