

Comparative Analysis of Regression Algorithms for Housing Price Prediction

Dr. Chaintanya Kishore Reddy Maddireddy, Boddupalli Naga Sai Vignesh,
Gangula Magadeesh, Gogireddy Keerthy Reddy, Gope ChandraMouli,
Dandiboyina Sharan Kumar

Department of Information Technology, NRI Institute of Technology, Agiripalli, Andhra Pradesh, India

ABSTRACT

Housing price estimation is a regression problem in which geographic, demographic, and housing attributes interact to influence property value. This paper presents a web-based system that trains and compares six regression approaches—Linear Regression, Ridge Regression, Random Forest, Gradient Boosting, K-Nearest Neighbours (KNN), and XGBoost—using the California Housing dataset. The workflow combines data cleaning, feature engineering, train–test separation, model evaluation, persistence of trained estimators, and a Flask interface for interactive prediction. Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and coefficient of determination (R^2) are used to assess predictive performance. The project report identifies Random Forest as the best-performing model among the evaluated alternatives, reporting an R^2 score of 88.4% in its comparison graphic and lower error metrics than the other displayed models. Three illustrative interface test cases produce increasing estimates for small, medium, and large properties. The work demonstrates an integrated workflow for model comparison and user-facing prediction; however, reproducible quantitative assessment is constrained by incomplete reporting of dataset split settings, model hyperparameters, and exact metric values.

Index Terms—housing price prediction, regression, Random Forest, XGBoost, California Housing, Flask, model evaluation.

Date of Submission: 09-10-2026

Date of acceptance: 20-10-2026

I. INTRODUCTION

Property valuation supports buyers, sellers, developers, and investors, but prices are affected by multiple interacting variables. Manual valuation and simple statistical models can be difficult to apply consistently when relationships are nonlinear or when many attributes must be considered. Machine-learning regression offers a data-driven alternative by learning associations between input features and a continuous target value.

This project develops an end-to-end application rather than an isolated prediction script. Six regression algorithms are trained on a common housing dataset, evaluated using the same metrics, and connected to a web interface. The comparison is intended to show how model choice changes predictive error and to make trained estimators accessible to users without requiring direct interaction with Python code.

The contributions documented in the project are:

1. A shared preprocessing and feature-engineering workflow
2. Implementation of six regression methods
3. Comparison using four complementary metrics
4. Integration of saved models into a Flask-based prediction interface.

II. RELATED WORK

Linear regression provides an interpretable baseline by representing the target as a weighted combination of input variables. Ridge regression adds an L2 penalty to constrain coefficient magnitude and can reduce instability when predictors are correlated. These methods are useful baselines, but their functional form may not capture complex interactions without additional feature transformations.

Tree ensembles provide a flexible alternative. Random Forest aggregates predictions from multiple decision trees, reducing dependence on any single tree. Gradient Boosting constructs an additive sequence of learners to reduce residual error, while XGBoost develops a scalable, regularized boosting implementation for structured data. KNN regression predicts from nearby training examples and is sensitive to the distance measure, feature scales, and neighbour count.

The project report motivates comparison under common data and evaluation conditions rather than selecting an algorithm solely from general claims in the literature. It also identifies an accessibility gap: model development often remains in a programming environment, whereas a web interface can allow users to submit features and view predictions. The current report does not provide a formal systematic literature-search protocol; related work is therefore summarized at the algorithm level.

III. PROBLEM FORMULATION

Let each observation be represented by a feature vector x_i and a continuous housing target y_i . A regression model $f(\cdot)$ estimates $\hat{y}_i = f(x_i)$. The objective is to select a model whose predictions minimize error on held-out observations while retaining a practical path to interactive inference. The project hypothesis is that regression methods differ in performance on the same dataset and that ensemble approaches may outperform simpler linear estimators.

The report describes cleaning missing, infinite, or invalid values, then dividing the processed data into training and testing subsets. It does not specify the exact split ratio, random seed, or complete feature schema; these are important reproducibility details and are not inferred here.

IV. PROPOSED METHODOLOGY

The system is organized into offline model development and online prediction. During offline development, the dataset is loaded with Python data tools, cleaned, transformed, and augmented with derived features. The data is split into training and test sets. Each algorithm is fitted on the training subset and evaluated on the held-out subset. The trained estimators are serialized with Joblib for reuse by the application.

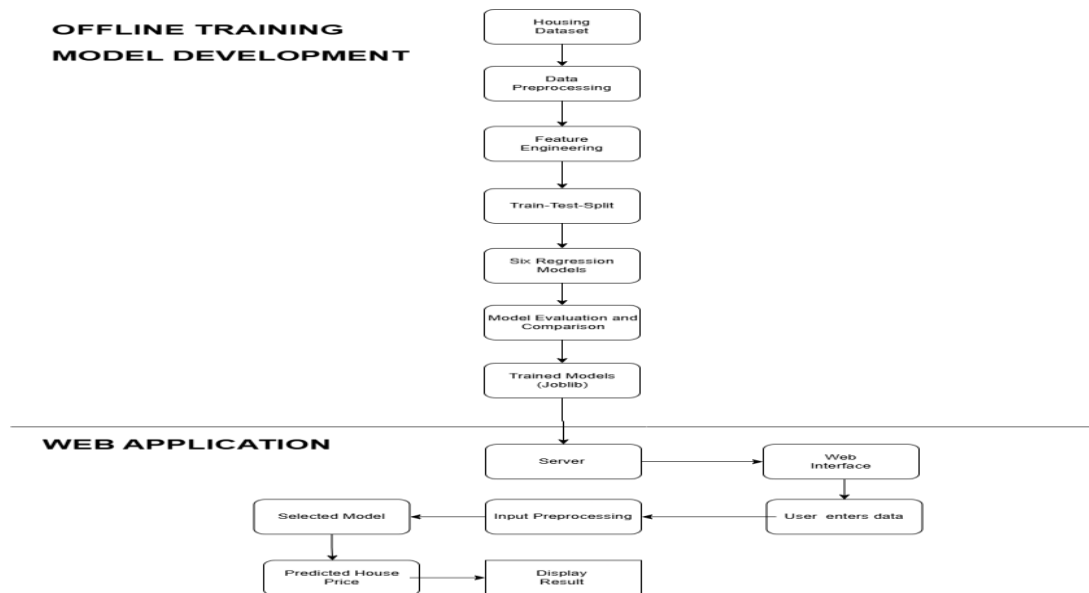


Fig. 1. Project workflow from dataset preparation and model training to web-based prediction.

The six methods described in the main system design are Linear Regression, Ridge Regression, Random Forest Regression, Gradient Boosting Regression, KNN Regression, and XGBoost Regression.

The derived features are intended to express household density and income context in forms that may be more informative than raw variables alone. For a derived feature to be valid, division-by-zero and invalid-value cases must be handled during preprocessing.

Linear Regression estimates coefficients by minimizing squared residuals and offers a straightforward baseline. Ridge Regression uses coefficient regularization to reduce sensitivity to correlated predictors. Their relative performance depends on whether the feature representation captures the relationships present in the target. These

models are valuable reference points because their behaviour is easier to inspect than that of more complex ensembles.

Random Forest trains a collection of decision trees using randomized samples and feature subsets, then aggregates their outputs for regression. This design can represent nonlinear thresholds and interactions without requiring those interactions to be specified manually. Gradient Boosting instead fits learners sequentially, with each stage targeting residual structure left by earlier stages. XGBoost implements a regularized and computationally optimized boosting framework. Both boosting approaches can be effective on structured data, but their performance depends on learning rate, tree depth, number of estimators.

KNN Regression is instance-based: a new input is compared with training examples and the target is estimated from nearby examples. Feature scales and the choice of k can materially change neighbour relationships. The project lists the algorithm but does not document its scaling strategy or neighbour count. As a result, the reported ranking should not be generalized to other datasets or parameter settings.

The comparison uses a single family of metrics to assess error from different angles. If y_i and $\hat{y}_i$ denote the actual and predicted values and $\bar{y}$ is the mean target, MSE and RMSE place relatively greater weight on large residuals, whereas MAE gives a linear penalty to each absolute deviation. R^2 can be negative when a model performs worse than the mean-target baseline on test data.

A. Evaluation Metrics

Four regression metrics are used. For n test observations, MSE is the mean squared residual, RMSE is its square root, and MAE is the mean absolute residual. R^2 compares residual sum of squares with total sum of squares; higher values generally indicate that the model explains more target variation, while lower MSE, RMSE, and MAE indicate smaller errors.

$$MSE = (1/n) \sum_i (y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{MSE}$$

$$MAE = (1/n) \sum_i |y_i - \hat{y}_i|.$$

$$R^2 = 1 - [\sum_i (y_i - \hat{y}_i)^2 / \sum_i (y_i - \bar{y})^2].$$

The metrics provide complementary views: MSE penalizes larger errors quadratically; RMSE returns the error to the target scale; MAE is less sensitive to large residuals than squared-error metrics; and R^2 measures fit relative to a mean-target baseline.

B. Web Application

The online stage loads the persisted estimator and accepts housing-related attributes through a Flask web interface. Inputs are converted into the feature format expected by the selected model, passed to the prediction function, and returned to the page as an estimated value. The screenshots show model selection cards, property fields, a prediction action, an output panel, and a comparison view of model metrics.

Fig. 2. Prediction interface showing model selection and housing input fields.

V. IMPLEMENTATION AND EXPERIMENTAL SETUP

This uses Python 3.8 or later, pandas and NumPy for data handling, scikit-learn for preprocessing, regression estimators and metrics, XGBoost for boosting, Joblib for persistence, Matplotlib for visualisation, and Flask for the web application. Windows, macOS, and Linux are supported operating-system options and a GPU is not required for the implemented models.

Testing is described through three representative interface scenarios: a small house (1 BHK, 1,000 sq. ft., unfurnished, nine years old), a medium house (3 BHK, 1,450 sq. ft., semi-furnished, three years old), and a large house (4 BHK, 2,200 sq. ft., semi-furnished, six years old). The recorded predictions are ₹50,23,000, ₹1,23,14,000, and ₹2,93,88,000, respectively.

TABLE I. DOCUMENTED INTERFACE TEST CASES AND OUTPUTS

Test Case	Documented input profile	Predicted price (INR)
Small house	1 BHK; 1,000 sq. ft.; unfurnished; 9 years	5,023,000
Medium house	3 BHK; 1,450 sq. ft.; semi-furnished; 3 years	12,314,000
Large house	4 BHK; 2,200 sq. ft.; semi-furnished; 6 years	29,388,000

The three interface examples show monotonic increases in predicted price as the example properties become larger and contain more bedrooms. This is a useful functional check that the application responds to different input profiles. However, the examples change multiple variables simultaneously, so they do not isolate the effect of any one feature.

VI. RESULTS AND DISCUSSION

The R^2 comparison graphic ranks Random Forest first, followed by XGBoost and Gradient Boosting, with Ridge, KNN, and Linear Regression lower in the displayed ordering. The Random Forest is at 88.4% R^2 . A separate error-metric graphic indicates lower RMSE and MAE for Random Forest than for the other displayed methods. The narrative in the source likewise identifies Random Forest as the strongest overall model in this experiment.



Fig. 3. R^2 comparison reproduced from the project report; Random Forest is reported at 88.4%.

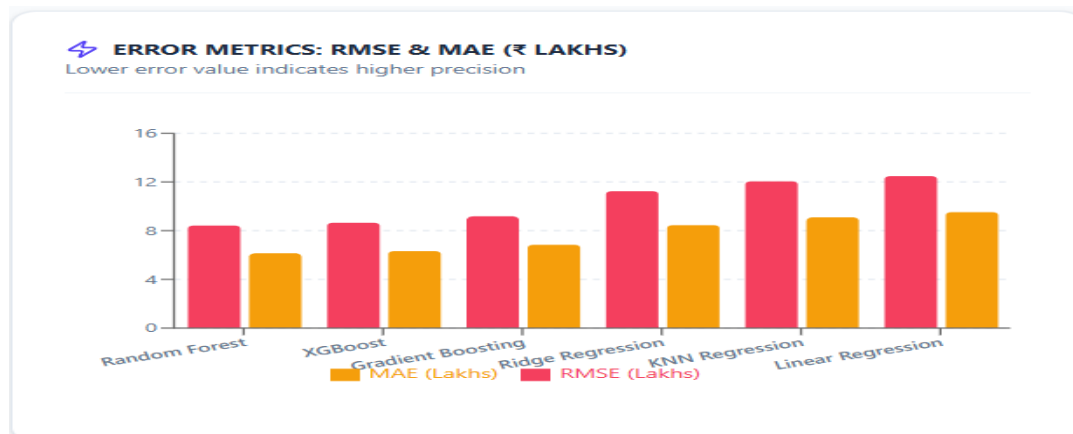


Fig. 4. RMSE and MAE comparison reproduced from the project report.

These findings are consistent with the ability of tree ensembles to represent nonlinear feature interactions. the ranking should be interpreted as the observed result rather than proof of universal superiority.

The metric dashboard in the source presents per-model R^2 , RMSE, MAE, MSE, and training time.

ALGORITHM	CATEGORY	R^2 SCORE	RMSE (₹ L)	MAE (₹ L)	MSE	TRAIN TIME
Random Forest Regressor	Ensemble	0.884 (88.4%)	₹8.42 L	₹6.15 L	70.89	185 ms
XGBoost Regressor	Ensemble	0.879 (87.9%)	₹8.65 L	₹6.32 L	74.82	240 ms
Gradient Boosting Regressor	Ensemble	0.862 (86.2%)	₹9.18 L	₹6.84 L	84.27	210 ms
Ridge Regression (L2)	Linear	0.792 (79.2%)	₹11.24 L	₹8.45 L	126.34	32 ms
K-Nearest Neighbors Regressor	Instance-Based	0.765 (76.5%)	₹12.95 L	₹9.18 L	145.20	45 ms
Multiple Linear Regression	Linear	0.748 (74.8%)	₹12.48 L	₹9.52 L	155.75	18 ms

Fig. 5. Evaluation-metric dashboard shown in the source project.

The reported workflow implies a separation between model development and serving. This separation is useful because training can be computationally more expensive than inference, while persisted estimators can be loaded by the application without fitting the model for each request. In a production implementation, the saved model should be versioned together with the preprocessing logic and feature order; otherwise, input transformations can differ between training and prediction and lead to invalid estimates.

A consistent comparison requires the same train–test observations for all candidates. Preprocessing operations that learn from data, including imputation and scaling where required, should be fitted on the training partition and then applied unchanged to the test partition. This is especially relevant to distance-based KNN and regularized linear regression.

The model comparison is presented visually in the report Fig. 3 communicates the relative R^2 ordering, while Fig. 4 compares the magnitude of two error metrics.

Establishing model accuracy requires comparing predictions withheld-out target values, while establishing usefulness for property decisions requires evaluating the relevance and freshness of the underlying data. These are distinct forms of validation.

The reported comparison also mixes an algorithmic objective with a usability objective. Model evaluation asks whether predicted values agree with held-out targets; interface evaluation asks whether users can enter data and understand the returned estimate. Both matter, but they require separate tests. A future evaluation should report both test-set metrics and interface test outcomes in a traceable format.

Data drift should be monitored because the relationships between income, location, occupancy, and market prices change over time. Periodic retraining should be triggered only after evaluating new data and comparing the updated model with the existing baseline.

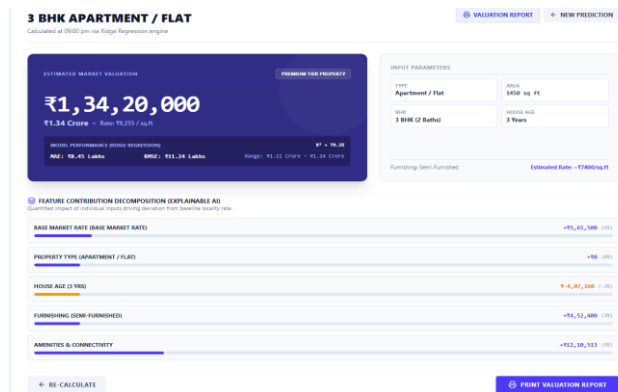


Fig. 6. Example prediction output dashboard reproduced from the project document.

VII. LIMITATIONS AND FUTURE WORK

The transformation from those interface inputs to the trained feature schema should therefore be documented and validated before deployment.

No claim of statistical significance or real-world valuation reliability is made.

Future work includes publishing the dataset source and version, complete feature definitions, preprocessing pipeline, model configurations, and exact evaluation outputs. A repeatable experiment should fit transformations only on training data, apply the same split to all models, and use cross-validation or repeated splits to assess variability. Additional geographic and time-relevant data, explicit handling of location, calibration of the input form to the training schema, and monitoring for market drift would improve practical relevance. Further models may be added only under the same evaluation protocol.

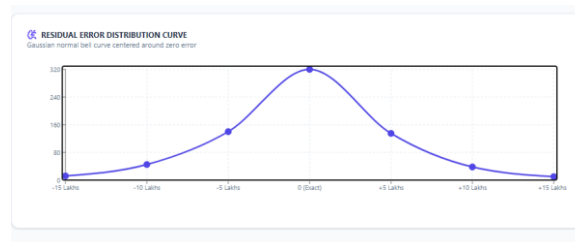


Fig. 7. Residual-error distribution visual included in the project report.

VIII. CONCLUSION

This paper presented a web-based housing price prediction system that compares six regression algorithms and connects saved estimators to an interactive Flask interface. The project report identifies Random Forest as the strongest model in its displayed comparison, with an R^2 score of 88.4% and lower reported error metrics. The integrated workflow demonstrates the practical combination of data preprocessing, feature engineering, comparative evaluation, model persistence, and web-based inference.

REFERENCES

- [1] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.
- [2] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
- [3] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [4] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
- [5] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [6] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [7] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.
- [8] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, 2nd ed. Springer, 2021.