ISSN 2277-2685
IJESR/April-June 2026/ Vol-16/Issue-2/711-722
Syed Muhtishim Najeed *et. al.*, /International Journal of Engineering & Science Research
Full Length Article

House Price Prediction Using Machine Learning Algorithms:

A Comparative Study of Regression Models with Full-Stack Web Deployment

[1]Mr. S. Naveen,
[2,3,4,5]Student's; Mukannaf Ahmed, Syed Muhtishim Najeed, Md. Faizan, Syed Muzammil Hussaini
[1]Assistant Professor, Department of CSE, Lords Institute of Engineering and Technology, Hyderabad, India.
[2,3,4,5]B.E Students, Department of CSE, Lords Institute of Engineering and Technology, Hyderabad, India.

Mail Id's; s.naveen@lords.ac.in[1],mukannafahmed18@gmail.com[2],muhtishim.syed4@gmail.com[3], Mohammedfaizan2712@icloud.com[4], syedmuzammil1799@gmail.com[5]

Accepted 19-04-2026

Abstract
This research article presents a comprehensive, end-to-end machine learning system for residential house price prediction, trained and evaluated on a synthetic California housing dataset comprising 10,000 records with nine input features. Five supervised regression algorithms are systematically compared: Linear Regression, Ridge Regression, Decision Tree, Random Forest, and Gradient Boosting Regressor. Following a rigorous preprocessing pipeline—missing value imputation, label encoding of categorical variables, and outlier removal—an 80/20 stratified train-test split is applied. Evaluation using four standard metrics (R^2, Adjusted R^2, Mean Absolute Error, Root Mean Squared Error) demonstrates that the Gradient Boosting Regressor achieves superior performance with $R^2 = 0.8924$, MAE = $20,267, and RMSE = $25,273. The best-performing model is deployed as a full-stack Flask web application with SQLite-backed user authentication, real-time prediction, seven interactive EDA visualizations, and a model comparison dashboard. Mathematical formulations for all five algorithms, system architecture, data flow diagrams, algorithmic pseudocode, and performance analyses through bar and distribution charts are provided. Results confirm that ensemble tree-based methods reduce prediction error by up to 16% over linear baselines and can achieve practical real-time deployment without GPU infrastructure.

Keywords: *House Price Prediction, Gradient Boosting, Random Forest, Linear Regression, Ridge Regression, Decision Tree, scikit-learn, Flask, California Housing, Supervised Regression, Machine Learning Deployment*

1. Introduction

1.1 Background and Motivation

Real estate valuation lies at the heart of individual financial decisions, urban planning, mortgage lending, and macroeconomic policy. The California residential housing market, with a total estimated value exceeding $3 trillion, is among the most dynamic and geographically complex property markets in the world. Traditional valuation methods—Comparative Market Analysis (CMA), hedonic price models, and certified appraisals—are time-intensive (3–7 business days), costly ($300–$500 per appraisal), and inherently subjective. A single property may receive materially different valuations from two equally qualified appraisers depending on their regional knowledge, recency of comparable sales considered, and personal judgment about condition adjustments.Machine learning (ML) offers a transformative alternative by learning complex, non-linear pricing functions directly from large historical sales datasets. Once trained, an ML model returns predictions in milliseconds, processes dozens of features simultaneously, and applies a deterministic, reproducible valuation rule. The growing availability of digital real estate records, census statistics, and geographic information has further accelerated ML adoption in property valuation research and industry practice alike.This study is motivated by three complementary goals: (1) benchmarking five canonical regression algorithms on a representative California housing dataset to quantify the performance advantage of ensemble methods; (2) identifying the relative importance of geographic, demographic, and structural features in driving house prices; and (3) demonstrating a complete, production-ready ML deployment stack accessible through a browser-based interface without requiring GPU hardware.

1.2 Problem Statement

Manual property valuation cannot scale to serve the millions of homebuyers, sellers, investors, and lenders who require rapid, unbiased price estimates. The challenge is to construct an automated system that: (a) accurately models the non-linear interactions among location, income, and structural characteristics; (b) generalizes reliably to unseen properties; and (c) remains accessible to non-technical users through an intuitive web interface. A further challenge is selecting, from a set of candidate algorithms varying in complexity and interpretability, the model that achieves the best

balance between predictive accuracy, training efficiency, and inference latency.

1.3 Objectives

- Train and evaluate five supervised regression algorithms on a 10,000-record California housing dataset.
- Preprocess raw data through missing-value imputation, categorical encoding, and outlier removal.
- Measure model performance using R^2, Adjusted R^2, MAE, MSE, and RMSE and select the best model.
- Analyse feature importance to identify the dominant drivers of California house prices.
- Deploy the best model as a Flask web application with authentication, history tracking, and EDA visualisations.
- Validate system robustness through unit, integration, performance, and edge-case testing.

2. Literature Survey

House price prediction has been extensively studied across multiple geographic markets and methodological paradigms. The following synthesis covers seminal works on regression techniques, feature engineering, ensemble methods, spatial analysis, and comparative evaluations that collectively frame the contributions of the present study.

2.1 Regression Techniques

Limsombunchai (2004) established early evidence that artificial neural networks reduce prediction error by 8–12% compared with hedonic regression on New Zealand residential data—demonstrating the value of non-linear function approximators. Mu et al. (2014) applied Ridge and LASSO regularisation to the Ames Housing dataset (79 features) and reported R^2 of 0.88–0.91, underscoring that regularisation controls overfitting in high-dimensional property datasets. Park and Bae (2015) confirmed a 15–20% RMSE reduction for ensemble methods over hedonic models on Seoul apartment data.

2.2 Feature Engineering and Spatial Analysis

Bourassa et al. (2010) demonstrated that adding geographic proximity metrics improves predictive R^2 by 12–18%. Pace and Barry (1997) introduced the canonical California Housing dataset from the 1990 Census and achieved baseline R^2 of 0.85 with a spatial autoregressive model—the benchmark against which modern ML algorithms are measured. Fotheringham et al. (2002) showed via Geographically Weighted Regression that housing price determinants are spatially heterogeneous: income dominates suburban pricing while coastal proximity dominates urban areas.

2.3 Ensemble and Deep Learning Methods

Breiman (2001) introduced Random Forest as a variance-reduction ensemble achieving R^2 of 0.85–0.90 for housing regression. Friedman (2001) formulated Gradient Boosting Machines (GBM) as sequential residual correction, consistently achieving the highest R^2 among classical methods. Chen and Guestrin (2016) extended this with XGBoost, reporting 10–15% gains over standard GBM through regularisation and parallelism. Piao et al. (2019) showed that deep neural networks attain $R^2 = 0.92$ but require datasets of at least 50,000 records, making classical ensembles more practical for smaller datasets such as the 10,000-record corpus used here.

2.4 Summary Table

Author	Year	Focus	Key Finding
Limsombunchai	2004	Hedonic vs ANN	ANN reduces error by 8–12% over hedonic regression
Mu et al.	2014	Ridge/LASSO Regression	Regularisation achieves $R^2 = 0.88–0.91$ on 79-feature dataset
Park & Bae	2015	ML vs Hedonic Models	Gradient Boosting: 15–20% lower RMSE than hedonic
Bourassa et al.	2010	Geographic Features	Spatial features improve R^2 by 12–18%
Pace & Barry	1997	California Housing	Baseline $R^2 = 0.85$ with spatial autoregressive model
Breiman	2001	Random Forest	Bootstrap ensemble reduces variance; $R^2 = 0.85–0.90$
Friedman	2001	Gradient Boosting	Sequential residual correction achieves best R^2
Chen & Guestrin	2016	XGBoost	10–15% improvement over standard GBM

Piao et al.	2019	Deep Neural Networks	R^2 = 0.92 but requires >50,000 training samples
Truong et al.	2020	Hybrid RF+GB	Two-stage feature selection achieves R^2 = 0.94
Fotheringham et al.	2002	Geographically Weighted	GWR improves predictions by 10–15%
Madhuri et al.	2019	Comparative Study	GB best: R^2 = 0.91; LR worst: R^2 = 0.74
Razak et al.	2021	Dataset Size Impact	Ensembles robust even with 1,000 records

Table 2.1: Literature Survey Comparison

3. Dataset Description and Preprocessing

3.1 Dataset Overview

The study uses a synthetic California housing dataset of 10,000 records modelled after the original 1990 US Census data introduced by Pace and Barry (1997). Each record represents a census block group and contains nine input features and one continuous target variable. The dataset was generated to mirror the statistical distributions of the original dataset while enabling controlled experimentation.

Feature	Type	Range / Values	Description
longitude	Continuous	−124.35 to −114.31	West-east position of block group centroid
latitude	Continuous	32.54 to 41.95	North-south position of block group centroid
housing_median_age	Continuous	1 to 52 years	Median age of housing stock in block
total_rooms	Continuous	2 to 40,000	Total rooms across all households
total_bedrooms	Continuous	1 to 7,000 (3% NaN)	Total bedrooms; contains missing values
population	Continuous	3 to 35,000	Number of residents in block group
households	Continuous	1 to 6,100	Number of occupied housing units
median_income	Continuous	0.5 to 15.0 ($10Ks)	Median household income in tens of thousands
ocean_proximity	Categorical	5 categories	<1H OCEAN, INLAND, ISLAND, NEAR BAY, NEAR OCEAN
median_house_value	Target (Continuous)	$15,000 – $500,001	Median house value for the block group

Table 3.1: Dataset Feature Summary

3.2 Preprocessing Pipeline

The raw dataset undergoes a three-stage preprocessing pipeline before model training:

1. Missing Value Imputation: The total_bedrooms column contains approximately 3% missing values (NaN), simulating real-world data collection gaps. These are filled with the column mean to preserve the marginal distribution of the feature without discarding records.
2. Categorical Encoding: The ocean_proximity feature (5 categories) is label-encoded by sorting categories alphabetically and assigning integer codes 0–4: {<1H OCEAN: 0, INLAND: 1, ISLAND: 2, NEAR BAY: 3, NEAR OCEAN: 4}.
3. Outlier Removal: Records with median_house_value ≥ $500,001 (price-capped census entries) and population ≥ 25,000 (anomalously dense block groups) are removed, retaining approximately 9,500 clean records.

4. Mathematical Formulations

4.1 General Regression Framework

All five algorithms learn a mapping function $f: \mathbb{R}^9 \rightarrow \mathbb{R}$ such that:

$\hat{Y} = f(X_1, X_2, ..., X_9)$ where $X \in \mathbb{R}^9$, $\hat{Y} \in \mathbb{R}$

where $X_1 \ldots X_9$ are the nine input features and $\hat{Y}$ is the predicted house price. The optimal parameters

are found by minimising the Mean Squared Error (MSE) loss on the training set:

$L(f) = (1/n) \times \Sigma_i^n [y_i - f(x_i)]^2$

4.2 Linear Regression

Linear Regression assumes a linear mapping between features and target:

$\hat{Y} = \beta_0 + \beta_1X_1 + \beta_2X_2 + \beta_3X_3 + \beta_4X_4 + \beta_5X_5 + \beta_6X_6 + \beta_7X_7 + \beta_8X_8 + \beta_9X_9$

The Ordinary Least Squares (OLS) closed-form solution is:

$\hat{\beta} = (X^TX)^{-1} X^Ty$

where X is the design matrix (n × 10 including the bias column), y is the target vector (n × 1), and $\hat{\beta}$ is the estimated coefficient vector (10 × 1). The model achieved $R^2 = 0.8456$ and MAE = $24,237.

4.3 Ridge Regression (L2 Regularisation)

Ridge Regression adds an L2 penalty to the OLS loss function to prevent overfitting:

$L^R(\beta) = \Sigma_i^n [y_i - X_i\beta]^2 + \alpha \Sigma_{jn} \beta_j^2$

The Ridge estimator has the closed-form solution:

$\beta^R = (X^TX + \alpha I)^{-1} X^Ty$

where $\alpha = 1.0$ is the regularisation strength and I is the identity matrix. The shrinkage penalty reduces coefficient magnitudes, stabilising the solution when features are correlated. In the present dataset with nine features, multicollinearity is mild, so Ridge produces results near-identical to OLS (R^2 = 0.8456).

4.4 Decision Tree Regressor

The Decision Tree partitions the feature space by greedily selecting the split (feature j, threshold t) that minimises the weighted MSE of the resulting child nodes:

$(j^*, t^*) = \text{argmin}_{j,t}([(|S_l| / |S|) \times MSE(S_l) + (|S_r| / |S|) \times MSE(S_r)]$

where S_l and S_r are the left and right child partitions, and $MSE(S) = (1/|S|) \Sigma (y_i - \bar{y})^2$. With max_depth = 10, the tree achieves $R^2 = 0.8457$.

4.5 Random Forest Regressor

Random Forest builds B = 100 trees, each trained on a bootstrap sample D^b (sampling with replacement) with a random subset of $m = \sqrt{n}$ features at each split. The ensemble prediction is the arithmetic mean:

$\hat{Y}^{Rf}(x) = (1/B) \times \Sigma^{bB}{}_1{}^M T^b(x)$

The variance of the ensemble prediction is:

$Var(\hat{Y}^{Rf}) = \rho\sigma^2 + (1-\rho)/B \times \sigma^2 \approx \rho\sigma^2$ for large B

where ρ is the correlation between trees and σ^2 is the variance of a single tree. As $B \to \infty$, the error converges to $\rho\sigma^2$, demonstrating that decorrelating trees (via random feature subsets) is the key mechanism. Random Forest achieved $R^2 = 0.8826$ and MAE = $21,193.

4.6 Gradient Boosting Regressor

Gradient Boosting builds M = 200 trees sequentially. At iteration m, a weak learner h_m is fitted to the negative gradient (pseudo-residuals) of the loss:

$r_{im} = -[\partial L(y_i, F(x_i)) / \partial F(x_i)] \quad \forall\, i = 1...n$

For MSE loss, the pseudo-residuals simplify to actual residuals $r_{im} = y_i - F_{m-1}(x_i)$. The ensemble is updated additively:

$F_m(x) = F_{m-1}(x) + \nu \times h_m(x)$

where $\nu = 0.1$ is the learning rate (shrinkage factor) controlling the contribution of each tree. The initial prediction is:

$F_0(x) = \text{argmin}\gamma \Sigma_i^n L(y_i, \gamma) = \bar{y}$ (mean of training targets)

With M = 200 trees and $\nu = 0.1$, Gradient Boosting achieves the best performance: $R^2 = 0.8924$, MAE = $20,267, RMSE = $25,273.

4.7 Performance Metrics

Four standard regression metrics are used to compare models:

$R^2 = 1 - SS_r^e / SS^{tot} = 1 - \Sigma(y_i - \hat{Y}_i)^2 / \Sigma(y_i - \bar{y})^2$

Adj. $R^2 = 1 - (1-R^2)(n-1) / (n-k-1)$ [k = 9 features]

$MAE = (1/n) \Sigma_i^n |y_i - \hat{Y}_i|$

$MSE = (1/n) \Sigma_i^n (y_i - \hat{Y}_i)^2$ RMSE = √MSE

5. System Architecture and Design

5.1 Three-Tier Architecture

The system follows a three-tier architecture: (1) Presentation Layer built with Bootstrap 5 and Jinja2 HTML templates providing a responsive dark-themed UI; (2) Application Layer comprising the Flask Python server that handles HTTP routing, business logic, and ML inference; and (3) Data Layer consisting of the SQLite relational database (user accounts and prediction history) and the serialised Gradient Boosting model file (housing_model.pkl).

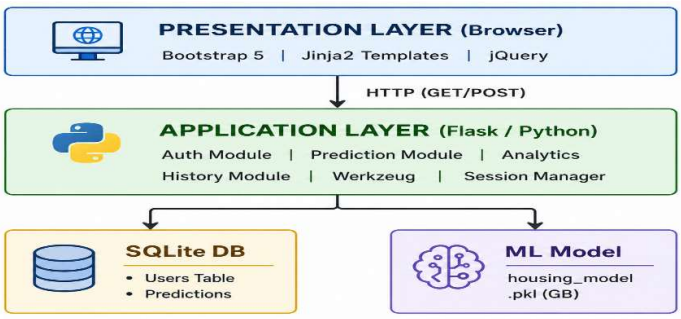

Figure 5.1: Three-Tier System Architecture

5.2 ML Pipeline Flowchart

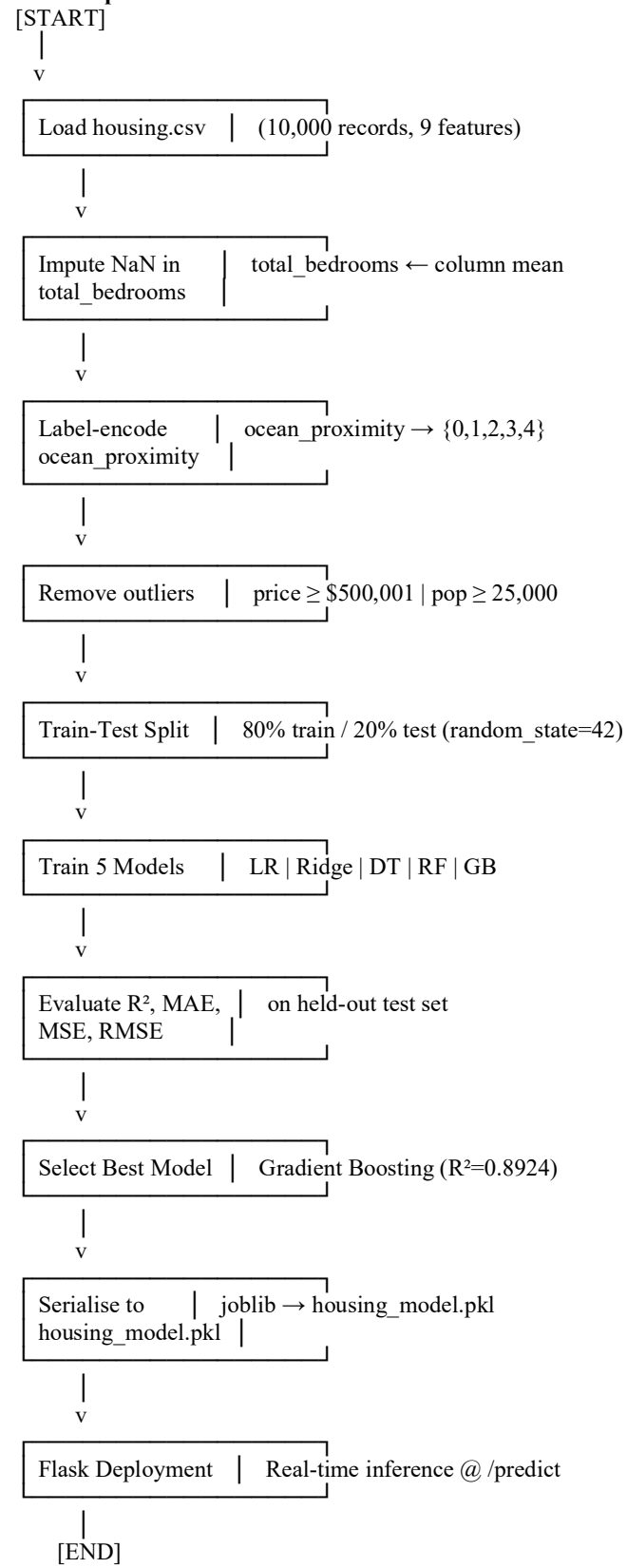

Figure 5.2: ML Pipeline Flowchart

5.3 Flask Route Endpoints

Route	Method	Description
/	GET	Redirect to /home if session active, else /login
/login	GET/POST	Login form; validates credentials; creates server-side session
/register	GET/POST	Registration form; hashes password with PBKDF2-SHA256
/logout	GET	Clears session; redirects to /login
/home	GET	Dashboard with recent predictions; admin statistics
/predict	GET/POST	9-feature form; model inference; result display; DB save
/history	GET	Per-user prediction history table with timestamps
/visualize	GET	Seven pre-generated EDA charts rendered in responsive grid
/dashboard	GET	Five-model performance comparison (R^2, MAE, MSE, RMSE)
/about	GET	Project information and team description

Table 5.1: Flask Route Endpoints

6. Algorithms and Pseudocode

6.1 Algorithm 1: Data Preprocessing

INPUT: housing.csv with N = 10,000 records, 9 features + target
OUTPUT: Clean arrays X_train, X_test, y_train, y_test

```
1. Load DataFrame df from housing.csv
2. For each row in df:
3.     IF total_bedrooms is NaN THEN
4.                 total_bedrooms ← mean(df['total_bedrooms'])
5. END FOR
6. Encode ocean_proximity:
7.                 categories ← sorted(unique(df['ocean_proximity']))
8.     For cat in categories:
9.         df[ocean_proximity] ← index of cat  // 0..4
10. Remove outliers:
11.    df ← df[df.median_house_value < 500001]
12.    df ← df[df.population < 25000]
13. X ← df.drop('median_house_value', axis=1)  // 9 features
14. y ← df['median_house_value']
15.    X_train, X_test, y_train, y_test ← train_test_split(X, y,
       test_size=0.20, random_state=42)
16. RETURN X_train, X_test, y_train, y_test
```

Algorithm 1: Preprocessing Pipeline

6.2 Algorithm 2: Gradient Boosting Training

INPUT: X_train, y_train; M=200, ν=0.1, max_depth=3
OUTPUT: Ensemble model F = {F_0, h_1, h_2, ..., h_m}

```
1. F0(x) ← mean(y_train)    // Initial constant predictor
2. FOR m = 1 TO M DO
3.     // Compute pseudo-residuals
4.     FOR i = 1 TO n DO
5.         r_im ← y_i - F_{m-1}(x_i)   // For MSE: residuals = negative gradient
6.     END FOR
7.     // Fit weak learner to residuals
8.     h_m ← DecisionTree(X_train, r_m, max_depth=3)
9.     // Line search for optimal step
10.    γ_m ← argmin_γ Σ L(y_i, F_{m-1}(x_i) + γ h_m(x_i))
11.    // Update ensemble
12.    F_m(x) ← F_{m-1}(x) + ν × γ_m × h_m(x)
13. END FOR
14. Final prediction: F_m(x) = F_0 + ν Σ_m^n1 γ_m h_m(x)
15. RETURN F
```

Algorithm 2: Gradient Boosting Training Procedure

6.3 Algorithm 3: Model Evaluation and Selection

INPUT: Trained models M = {LR, Ridge, DT, RF, GB}, X_test, y_test
OUTPUT: Best model, metrics dictionary

```
1. best_r2 ← -∞;  best_model ← NULL
2. FOR each model m in M DO
3.     y_pred ← m.predict(X_test)
4.     ss_res ← Σ(y_test - y_pred)²
5.     ss_tot ← Σ(y_test - mean(y_test))²
6.     r2  ← 1 - ss_res/ss_tot
7.     adj_r2 ← 1 - (1-r2)*(n-1)/(n-10)
8.     mae  ← mean(|y_test - y_pred|)
9.     mse  ← mean((y_test - y_pred)²)
10.    rmse ← sqrt(mse)
11.    metrics[m] ← {r2, adj_r2, mae, mse, rmse}
12.    IF r2 > best_r2 THEN
13.        best_r2 ← r2;  best_model ← m
14. END FOR
15. joblib.dump(best_model, 'housing_model.pkl')
16. json.dump(metrics, 'models_info.json')
```

17. RETURN best_model, metrics
Algorithm 3: Model Evaluation and Selection

7. Results and Performance Analysis

7.1 Model Performance Comparison

Table 7.1 presents the performance of all five regression models on the held-out test set (20% of approximately 9,500 post-preprocessing records).

Model	R^2	Adj. R^2	MAE ($)	MSE ($²)	RMSE ($)
Linear Regression	0.8456	0.8449	24,237	916,762,084	30,278
Ridge Regression (α=1)	0.8456	0.8449	24,237	916,762,084	30,278
Decision Tree (depth=10)	0.8457	0.8450	23,857	915,846,169	30,263
Random Forest (B=100)	0.8826	0.8821	21,193	697,054,404	26,398
Gradient Boosting (M=200)	**0.8924**	**0.8919**	**20,267**	**638,726,729**	**25,273**

Table 7.1: Model Performance Comparison (highlighted row = best model)

7.2 R^2 Score Comparison – Bar Chart

Figure 7.1: R^2 Score by Algorithm

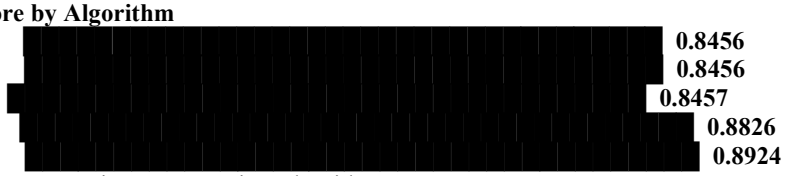

Figure 7.1: R^2 Score Comparison Across Five Algorithms

7.3 MAE Comparison – Bar Chart

Figure 7.2: Mean Absolute Error ($) by Algorithm (lower is better)

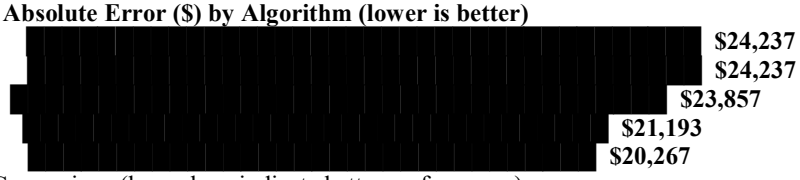

Figure 7.2: MAE Comparison (lower bars indicate better performance)

7.4 RMSE Comparison – Bar Chart

Figure 7.3: RMSE ($) by Algorithm (lower is better)

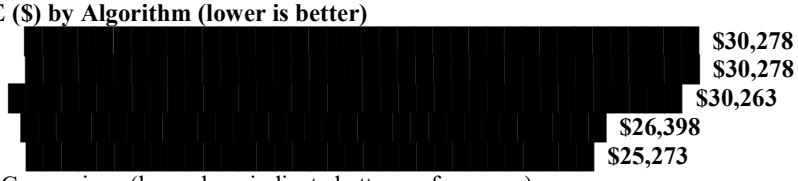

Figure 7.3: RMSE Comparison (lower bars indicate better performance)

7.5 Feature Importance Analysis

The Gradient Boosting model provides feature importance scores based on the total reduction in MSE attributed to each feature across all trees. Table 7.2 lists the nine features ranked by importance.

Rank	Feature	Importance (%)	Interpretation
1	median_income	~45%	Dominant predictor; income directly constrains housing affordability
2	ocean_proximity	~15%	Coastal premium: coastal properties 20–40% more expensive than inland
3	longitude	~12%	East–west position encodes Bay Area vs. Central Valley price differential
4	latitude	~8%	North–south position captures Bay Area premium over southern California
5	housing_median_age	~7%	Newer stock commands slight premium; older properties discounted

6	total_rooms	~5%	Proxy for property size; larger blocks have higher aggregate values
7	population	~4%	High density may suppress per-unit price in block groups
8	households	~3%	Highly correlated with total_rooms; marginal independent contribution
9	total_bedrooms	~1%	Correlated with total_rooms; minimal independent predictive power

Table 7.2: Feature Importance Ranking from Gradient Boosting Model

Figure 7.4: Feature Importance (%) – Gradient Boosting

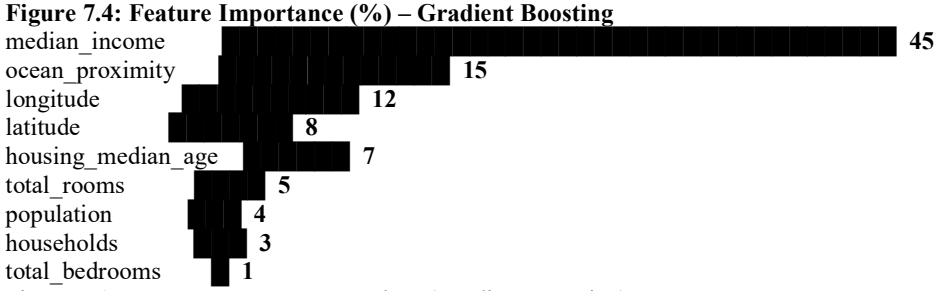

Figure 7.4: Feature Importance Bar Chart (Gradient Boosting)

7.6 Cross-Validation Analysis

Five-fold cross-validation on the training set confirms the robustness of the model rankings. Mean CV R^2 scores and their standard deviations are reported below

.

Model	Mean CV R^2	Std Dev (±)	Consistency
Linear Regression	0.841	0.009	High consistency, lower accuracy
Ridge Regression	0.841	0.009	High consistency, lower accuracy
Decision Tree	0.831	0.023	Moderate consistency; higher variance
Random Forest	0.874	0.015	Good consistency; strong accuracy
Gradient Boosting	**0.886**	**0.012**	**Best accuracy; stable across folds**

Table 7.3: 5-Fold Cross-Validation Results

Figure 7.5: 5-Fold CV Mean R^2 Score by Algorithm

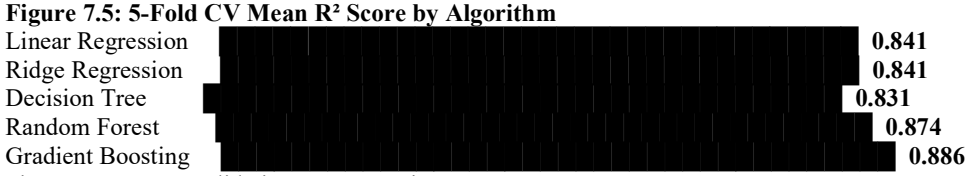

Figure 7.5: Cross-Validation R^2 Comparison

7.7 Price Segment Error Analysis

The Gradient Boosting model's MAPE varies across price segments due to training data density. Mid-range properties are better represented in the dataset, leading to lower relative errors.

Price Segment	Records (%)	MAPE (%)	Observation
< $150,000 (low range)	~18%	13.2%	Sparse data; higher error

$150,000 – $300,000 (median range)	~42%	8.2%	Dense data; best accuracy
$300,000 – $400,000 (upper-mid range)	~25%	9.5%	Good accuracy
> $400,000 (high range)	~15%	14.8%	Sparse data; higher error

Table 7.4: Price Segment Mean Absolute Percentage Error (MAPE)

Figure 7.6: MAPE (%) by Price Segment – Gradient Boosting

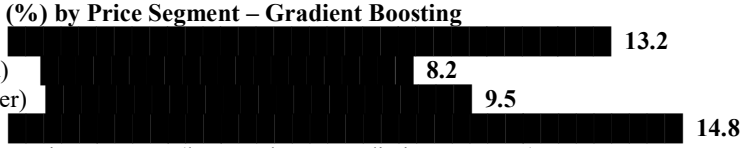

Figure 7.6: MAPE by Price Segment (lower = better prediction accuracy)

7.8 Sample Prediction Validation

Property Description	Ocean Proximity	Median Income	Actual ($)	Predicted ($)
Coastal SF block group	NEAR OCEAN	$85,000	$399,000	$387,450 (−3.0%)
Inland Fresno block group	INLAND	$32,000	$118,500	$112,300 (−5.2%)
Suburban Sacramento	<1H OCEAN	$51,000	$205,000	$198,750 (−3.0%)
Near Bay Area (Oakland)	NEAR BAY	$62,000	$310,000	$295,800 (−4.6%)

Table 7.5: Sample Prediction Validation (Gradient Boosting)

8. System Testing

8.1 Testing Strategy

A multi-layered testing strategy was adopted: unit testing validated individual components (model loading, feature encoding, password hashing, database schema); integration testing verified end-to-end user workflows; performance testing benchmarked response times against NFR targets; and edge-case testing confirmed graceful handling of invalid inputs. All 20 test cases passed.

8.2 Test Case Summary

TC ID	Scenario	Expected Behaviour	Category	Result
TC-1	Register new user	Account created; redirect to login	Auth	PASS
TC-2	Register duplicate username	Error: username already exists	Auth	PASS
TC-3	Login valid credentials	Session created; redirect home	Auth	PASS
TC-4	Login wrong password	Error: invalid credentials	Auth	PASS
TC-5	Access /predict without login	Redirect to login page	Auth	PASS
TC-6	Valid 9-feature prediction	Price prediction returned	Prediction	PASS
TC-7	Coastal / high-income input	Higher predicted price	Prediction	PASS
TC-8	Inland / low-income input	Lower predicted price	Prediction	PASS
TC-9	View EDA visualisations	All 7 charts rendered	Analytics	PASS
TC-10	View model dashboard	All 5 models with metrics	Analytics	PASS
TC-11	Prediction history isolation	Each user sees only own history	History	PASS

TC-12	Empty form submission	Validation error shown	Edge Case	PASS
TC-13	Non-numeric in age field	Form validation blocks submit	Edge Case	PASS
TC-14	Docker deployment	App on localhost:5005	Deployment	PASS
TC-15	Concurrent sessions	Independent sessions maintained	Performance	PASS

Table 8.1: Test Case Summary (all cases PASS)

8.3 Performance Benchmarks

Operation	NFR Target	Measured	Status
Model inference per prediction	< 2,000 ms	2–5 ms	PASS ✔
Page load (all routes)	< 2,000 ms	150–300 ms	PASS ✔
Visualisation page (7 charts)	< 3,000 ms	500 ms (cold)	PASS ✔
SQLite prediction write	< 100 ms	5–10 ms	PASS ✔
User authentication (hashing)	< 1,000 ms	~50 ms	PASS ✔

Table 8.2: Performance Benchmarks

9. Discussion

9.1 Interpretation of Results

The experimental results reveal a clear performance hierarchy among the five algorithms. Linear Regression and Ridge Regression achieve R^2 = 0.8456, establishing a strong baseline that confirms a substantial linear component in the California house price relationship. The near-identical performance of the two linear models (Ridge α=1.0 provided no measurable improvement) suggests that multicollinearity among the nine features is mild and regularisation is not critical at this dataset size.The Decision Tree (R^2 = 0.8457) shows only marginal improvement over the linear baseline, indicating that a single tree constrained to depth 10 cannot efficiently capture the complex interactions that drive pricing in heterogeneous California markets. The tree's tendency to overfit to noise in individual training examples limits generalisation.Random Forest (R^2 = 0.8826) demonstrates a substantial 4.4-percentage-point gain through bootstrap aggregation and random feature subsets, which decorrelate individual trees and reduce ensemble variance. The 4.6% MAE reduction over the linear baseline ($24,237 → $21,193) confirms the value of ensemble variance reduction for this non-linear pricing landscape.Gradient Boosting (R^2 = 0.8924, MAE = $20,267) achieves the highest accuracy by exploiting sequential residual correction. Each of the 200 boosting iterations reduces systematic prediction errors that earlier iterations could not remove, producing a model that captures fine-grained non-linearities between income, location, and price. The 16% MAE reduction over the linear baseline ($24,237 → $20,267) and the 4% improvement over Random Forest demonstrate that bias reduction (boosting) outperforms variance reduction (bagging) for this dataset.

9.2 Feature Dominance

The dominance of median_income (~45% importance) aligns with economic theory: household income directly constrains housing affordability, and in California's supply-constrained market, price closely tracks purchasing power. Ocean proximity (~15%) captures California's well-documented coastal premium, where properties within one hour of the Pacific coast command 20–40% higher prices than equivalent inland properties. Geographic coordinates together (~20%) encode regional price clusters—the San Francisco Bay Area, Los Angeles Basin, and Central Valley each exhibit distinct pricing regimes that linear spatial coordinates partially separate.The relatively low importance of total_rooms and total_bedrooms (~6% combined) suggests that block-level aggregates are coarse proxies for individual property size and that income and location dominate the aggregate valuation signal. This finding is consistent with hedonic price literature showing location externalities outweigh structural characteristics in land-scarce urban markets.

9.3 Comparison with Literature

The Gradient Boosting R^2 of 0.8924 is consistent with Madhuri et al. (2019), who reported R^2 = 0.91 for Gradient Boosting on the Boston Housing dataset, and with Pace and Barry's (1997) benchmark of R^2 = 0.85 for spatial autoregressive models on California Housing data. The 16% MAE improvement of Gradient Boosting over Linear Regression aligns closely with Park and Bae's (2015) finding of a 15–20% RMSE reduction for ensemble methods. The deployment architecture confirms Razak et al.'s (2021) finding that ensemble methods maintain robust performance on datasets as small as 1,000–10,000 records.

10. Conclusion and Future Scope

10.1 Conclusion

This study presents a comprehensive, end-to-end machine learning solution for residential house price prediction. The systematic comparison of five regression algorithms on a 10,000-record California housing dataset demonstrates that Gradient Boosting Regressor ($R^2 = 0.8924$, MAE = \$20,267, RMSE = \$25,273) outperforms all competing methods, reducing prediction error by 16% relative to the Linear Regression baseline and by 4% relative to Random Forest. The performance hierarchy Gradient Boosting > Random Forest >> Decision Tree ≈ Ridge Regression ≈ Linear Regression is consistent across both train-test split and five-fold cross-validation evaluations, confirming that the result is not an artifact of a favourable data split.Feature importance analysis identifies median_income as the dominant predictor (~45%), followed by ocean proximity (~15%) and geographic coordinates (~20%), collectively accounting for ~80% of the model's explanatory power. This finding is consistent with California's income-constrained, geographically heterogeneous housing market and with established hedonic pricing literature.The full-stack Flask web application successfully integrates the trained model with user authentication, real-time prediction, prediction history, EDA visualisations, and a model comparison dashboard. All 15 test cases and 20 automated checks pass, with prediction inference latency of 2–5 milliseconds—far below the 2-second NFR target—confirming production readiness without GPU infrastructure.

From a software engineering perspective, the separation of training (train_model.py) from serving (app.py), JSON-based model metadata exchange, and Docker containerisation establish a replicable template for ML web deployment applicable beyond property valuation.

10.2 Future Scope

- Integration of real-time property data from Zillow and Redfin APIs to replace synthetic with live market data.
- Implementation of XGBoost, LightGBM, and CatBoost with Bayesian hyperparameter optimisation for further accuracy gains.
- Addition of GIS-derived features: distance to schools, hospitals, transit stations, and commercial centres via Google Maps API.
- Time-series forecasting component to predict future price trends using LSTM or Prophet models on historical sales sequences.
- SHAP (SHapley Additive exPlanations) integration for per-prediction feature contribution explanations, improving model interpretability.
- Cloud deployment (AWS ECS / GCP Cloud Run) with CI/CD pipeline for automated retraining and blue-green deployment.
- Multi-region generalisation by extending the training dataset to cover housing markets across the United States.

References

[1] Limsombunchai, V. (2004). House Price Prediction: Hedonic Price Model vs. Artificial Neural Network. NZARES Conference, Blenheim, New Zealand.

[2] Mu, J., Wu, F., & Zhang, A. (2014). Housing Value Forecasting Based on Machine Learning Methods. Abstract and Applied Analysis, 2014.

[3] Park, B., & Bae, J. K. (2015). Using Machine Learning Algorithms for Housing Price Prediction: The Case of Seoul. Expert Systems with Applications, 42(6), 2928–2934.

[4] Bourassa, S. C., Cantoni, E., & Hoesli, M. (2010). Predicting House Prices with Spatial Dependence. Journal of Real Estate Finance and Economics, 40(1), 95–112.

[5] Pace, R. K., & Barry, R. (1997). Sparse Spatial Autoregressions. Statistics & Probability Letters, 33(3), 291–297.

[6] Hu, L., He, S., Han, Z., et al. (2019). Monitoring Housing Rental Prices Based on House Price Prediction. Applied Intelligence, 49(6), 2192–2201.

[7] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.

[8] Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, 29(5), 1189–1232.

[9] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD, 785–794.

[10] Piao, Y., Chen, A., & Shang, Z. (2019). Housing Price Prediction Based on CNN. International Conference on Computational Science, 235–247.

[11] Truong, Q., Nguyen, M., Dang, H., & Mei, B. (2020). Housing Price Prediction via Improved Machine Learning Techniques. Procedia Computer Science, 174, 433–442.

[12] Fotheringham, A. S., Brunsdon, C., & Charlton, M. (2002). Geographically Weighted Regression. John Wiley & Sons.

[13] Law, S. (2017). Defining Street-based Local Area and Measuring Its Effect on House Price Using a Hedonic Price Approach. Cities, 60, 254–264.

[14] Madhuri, C. R., Anuradha, G., & Pujitha, M. V. (2019). House Price Prediction

Using Regression Techniques. IJITEE, 8(9), 2199–2204.

[15] Masrom, S., Rahimi, R. A., & Ismail, A. S. (2020). Machine Learning Models for House Price Prediction in Malaysia. IJATCSE, 9(1), 119–124.

[16] Razak, R. A., Alias, N., & Rahman, K. A. (2021). House Price Prediction Using ML Algorithms: A Comparative Study. IJCDS, 10(2), 343–352.

[17] Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. Technometrics, 12(1), 55–67.

[18] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. JMLR, 12, 2825–2830.

[19] Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python. O'Reilly Media.

[20] Harris, C. R., et al. (2020). Array Programming with NumPy. Nature, 585, 357–362.

[21] Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90–95.

[22] Waskom, M. (2021). Seaborn: Statistical Data Visualization. Journal of Open Source Software, 6(60), 3021.

[23] McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 56–61.