House Price Prediction by Machine Learning Model Using Regression Techniques

Krishna Shalawadi
IMS22IS066
Ramaiah Institute of Technology, Bangalore
Email: krishnashalawadi27@gmail.com

K. Naveen Kumar
1MS22IS054
Ramaiah Institute of Technology, Bangalore
Email: knaveenkumarv18@gmail.com

Abstract—This paper presents a regression-based machine learning model to predict house prices using the popular "kc house data.csv" dataset [6].The approach combines careful data preprocessing with advanced feature engineering techniques to improve model accuracy and robustness. We also calculated error for better result that can be developed in future. Errors such as R^2, RMSE, and MAE were used. Features Engineering were done to improve model accuracy. The frontend was built using Streamlit to allow real-time predictions [4].This frontend allows users to input custom house features and instantly receive predicted price estimates, making the application practical and interactive. The system demonstrates how machine learning can be effectively applied to real-world datasets and deployed in accessible applications for end-users.

Index Terms—House Price Prediction, Machine Learning, Regression, Streamlit, Feature Engineering, Web Application, Kings Country, Price, Linear Regression, Ridge Regression, Random Forest, XGBoost, Gradient Boosting.

INTRODUCTION

The house marking plays a high role in the economic structure, financial structure, development of the economic growth of the market. Predicting the house price is important or critical not only for the real estate agents but also for policy makers, home owners, financial institutions, etc. The Real Estate price is evaluated by local market, neighborhood, economic indicators, consumer sentiments and all other property.

Variations in the house prices can have wide effect for the both home owners and real estate agent. If prices are too high, it can create both housing and financial problems. But if the prices are too low, then the people stop investing on the property or home. Buyers need an accurate price prediction to make smart decisions, while sellers want to get the best value for their property. Also, the investors also see or depends on the good price predictions to plan better and avoid risks.

Housing prices also affects how and where people live. When prices go up fast, it can become harder for people to afford homes, leading to problems like higher living costs and out lower-income families. But if prices stay flat, it could be sign of poor development or weak economy. That's why it's important to understand and predict house prices trends where it helps everyone like buyers, sellers, investors, city planners to make better choice and support healthy, balanced economic growth in cities and communities.

This paper consists the subsections that are related work, dataset description, methodology, results, limitation, comparative analysis, conclusion, future developments and references. The proposed model is working of a publicly available dataset from Kaggle the detailed description is recorded in coming subsections.

I. RELATED WORK

The prediction of housing prices has long been a topic of interest due to its economic significance and potential to aid decision-making for buyers, sellers, and policymakers. Over recent years, machine learning and deep learning techniques have seen widespread adoption in this domain, offering improvements in both predictive accuracy and model interpretability. Research from 2022 to 2025 reveals a variety of methodological innovations and a focus on balancing performance with computational efficiency and transparency.

In 2025, Li and Zhang introduced an explainable AI model that integrates eXtreme Gradient Boosting (XGBoost) with SHapley Additive exPlanations (SHAP) to not only predict house prices but also interpret the contribution of each feature to the final output. Their model achieved strong performance (RMSE = 42,000, R^2 = 0.89), though it was constrained

to U.S. suburban markets, limiting its geographic generalizability [1]. Rahman and Liu leveraged transformer-based architectures—originally designed for natural language processing—to model spatiotemporal dependencies in real estate data. Despite achieving an MAE of 35,000, the model's heavy reliance on GPU resources posed scalability challenges [2]. Meanwhile, Sharma and Kim proposed a hybrid LSTM-ARIMA model to improve temporal forecasting. This approach benefited from the strengths of both traditional time-series modelling and deep learning, resulting in an RMSE of 38,500, although it was less effective for short-term trend prediction [3].

In 2024, privacy concerns were addressed by Wu and Joshi, who implemented a federated learning framework using XGBoost. This allowed multiple agents to collaboratively train a model without sharing sensitive data, achieving an RMSE of 44,000 and R^2 of 0.85. However, the system faced challenges related to communication delays and synchronization across agents [4]. Bansal and Ortega utilized AutoML to automate feature selection and hyperparameter tuning, achieving an MAE of 31,000. The trade-off, however, was significant computational overhead [5]. Nguyen and Hassan conducted a comparative study across traditional and deep learning models, including Linear Regression, Random Forests, CNNs, and LSTMs. Their findings showed CNN models performed the best (RMSE = 36,000), but issues with missing data in the dataset affected overall robustness [6].

In 2023, spatial dynamics were emphasized. Fang and Lee applied a spatial attention neural network that dynamically adjusted the importance of geographic features during training. The model showed high accuracy ($R^2 = 0.88$) but did not generalize well in rural regions [7]. Yu and Sakamoto modelled real estate markets as graphs using Graph Neural Networks (GNNs) to capture relationships between neighbouring properties. While achieving a solid MAE of 33,500, the approach had high training time due to complex graph operations [8]. Perez and Zadeh proposed a multimodal deep learning framework that combined tabular data with satellite images to improve spatial understanding. Their model recorded an RMSE of 34,700, though it required substantial image storage and preprocessing resources [9]. Dimitrov and Singh used Bayesian networks for probabilistic reasoning, offering a transparent framework but suffering from sensitivity to missing input data [10].

In 2022, ensemble and regression-based models were prominently explored. Patel and Kumar compared Bagging, Boosting, and Stacking, finding that stacking produced the lowest RMSE (39,500). However, Bagging showed signs of overfitting on smaller datasets [11]. Ahmed and Zhou implemented LightGBM, a gradient boosting framework known for efficiency and speed, achieving an MAE of 32,800. The model, however, was sensitive to hyperparameters such as learning rate [12]. Silva and Tan applied geospatial regression using geographically weighted models to account for urban spatial heterogeneity. They reported an R^2 of 0.84, but generalization across different cities was limited [13]. Gupta and Ali compared regularized regression models—Ridge, Lasso, and Support Vector Regression (SVR). SVR achieved the best results with an MAE of 36,100, while Lasso struggled when the dataset included many weakly informative features [14]. Lin and Park explored Deep Reinforcement Learning (DRL) using Deep Q-Networks (DQN) to simulate pricing strategies for investment. Their model showed an 18% improvement in simulated returns but lacked real-world validation due to simplifications in the simulation environment [15].

This extensive body of work indicates a clear trend toward hybrid, interpretable, and privacy-aware modelling strategies in real estate valuation. While deep learning models often provide superior accuracy, they frequently require substantial computational resources and large datasets. Traditional models, on the other hand, still offer valuable baselines, particularly in low-resource environments. Challenges such as data quality, generalization across diverse geographic contexts, and interpretability remain persistent across studies.

II. Dataset Description

The ‘kc _ house data.csv‘ dataset contains 21,613 house sales records were there are about 21 different features. Each entry in the mentioned dataset has a unique ”id” and ”data” of the sale. The main focus is on the target variable ”price”[21].

Key feature present in the dataset includes ”bedrooms”, ”bathrooms”, ”sqft living”, ”sqft _ lot”. These describe the characteristics of the house and its surrounding. In the dataset there are other attributes such as ”floors”, ”waterfront”, ”view”, ”condition” and ”grade”. These give the overview of the property’s state.

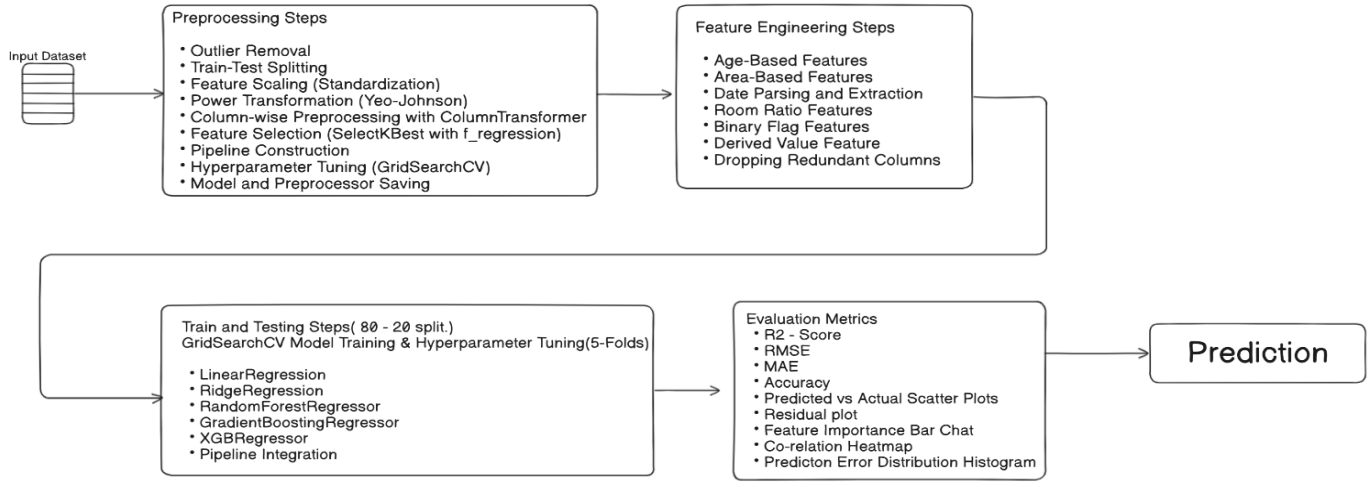
There are other attributes but important one such as ”yr built”(year built), ”yr renovated”(year renovated) gives the details of year of construction and year of the last renovation done. There are other geographical

information such as zipcode, lat (latitude), long (longitude) for location analysis. There is also a function which gives the average of 15 nearest neighbours' living and lot areas where they suggest.

III. METHODOLOGY

The methods we had used to build the proposed were first to prepare the dataset for input to our model for property values. This data set is suited or different type of analyses where we can explore influence in real estate vaules.

that we had done preprocessing with StandardScaler, PowerTransformer(yeo-johnson),
ColumnTransformer. After that Feature Engineering, Outlier Removal, Model Training, Evaluation and finally output.



A. Data Preprocessing

1. StandardScaler: This function transfroms feature by

removing the mean and scaling to unit variance. It converts data such a way that it has mean of 0 and standrad deviation of 1.

$$x_{\text{scaled}} = \frac{x_i - \mu}{\sigma}$$

Where:

- x_i is the original feature value,
- μ is the mean of the feature, · σ is the standard deviation of the feature.

2. PowerTransformer (method= 'yeo-johnson'): This functionconverts features to a more Gaussian distribution. The method we used is useful as it work with both positive and negative values. If x0:

$$y(\lambda) = \begin{cases} ((x+1)^{\lambda} - 1)/\lambda & \text{if } \lambda \neq 0 \\ \log(x+1) & \text{if } \lambda = 0 \end{cases}$$

If x¡0:

Where is the transformation parameter determined by the algorithm to minimize skewness.

3. ColumnTransformer: this utility applies different transformsto different columns of the input data.

In this method that is data preprocessing we applied numerical transformer (which includes StandardScaler and PowerTransformer) gives the proper preprocessing steps are applied only when they are required.

B. Feature Engineering

Key engineered features include:

- price per sqft = price / sqft living
- house age = sale year - yr _built
- years since renov = sale year - yr renovated
- lot ratio, living ratio, rooms, bed _bath ratio, etc.

In order to upgrade model performance, these step starts generating new, more insightful features from the available raw data. The new features like sale year, sale month, house age (based on sale year and yr built), years since renov (assuming yr renovated is 0), total area (sqft living + sqft basement), lot ratio (sqft living / sqft lot), living ratio (sqft living / sqft living15), bed bath ratio (bedrooms / bathrooms), rooms (bedrooms + bathrooms), is _waterfront (binary from waterfront), has view (binary from view), and price per sqft (price / sqft _living) are done here and implemented.

C. Outlier Removal

Outliers are data the points which significantly differ from other observations, which can impact model training. For each specified column (price, sqft _living, sqft _lot, bedrooms, bathrooms), the 25th percentile (Q1) and 75th percentile (Q3) are calculated. The IQR is then defined as Q3 - Q1. Any data point falling below Q1-1.5×IQR or above Q3+1.5×IQR is considered an. outlier and removed

```python
def remove_outliers(df, column, threshold=1.5):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - threshold * IQR
    upper_bound = Q3 + threshold * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
```

D. Model Training

Models used include:

· Linear Regression:

Linear Regression models the linear relationship between a dependent variable y and one or more independent variables x1, x2, ..., xn.

$$h_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

where: $h_\theta(X)$ is the predicted output. θ_0 is the intercept. $\theta_1, \theta_2, \ldots, \theta_n$ are the coefficients (weights) for each independent variable $x_1, x_2, \ldots, x_n$.

```python
"Linear Regression": {
        'model': Pipeline([
            ('preprocessor', preprocessor),
            ('selector', selector),
            ('regressor', LinearRegression())
        ]),
        'params': {
            'selector__k': [10, 15, 'all'],
        }
    }
```

· Ridge Regression:

Ridge Regression extends Linear Regression by adding an L2 regularization term to the cost function. This term penalizes large coefficients, which helps prevent overfitting.

$$J(\theta) = \text{MSE}(\theta) + \alpha \sum_{i=1}^{n} \theta_i^2$$

Here, $J(\theta)$ is the objective function to be minimized. MSE(θ) is the Mean Squared Error of the linear regression model:

$$\text{MSE}(\theta) = \frac{1}{m} \sum_{j=1}^{m} (y_j - h_\theta(X_j))^2$$

where: - m is the number of training samples, - y_j is the actual value, - $h_\theta(X_j)$ is the predicted value. α (alpha) is the regularization parameter ($\alpha \geq 0$), which controls the strength of the L_2 penalty.

$$\sum_{i=1}^{n} \theta_i^2$$

is the L_2 norm (squared) of the coefficients, excluding θ_0.

```python
"Ridge Regression": {
        'model': Pipeline([
            ('preprocessor', preprocessor),
            ('selector', selector),
            ('regressor', Ridge())
        ]),
        'params': {
            'selector__k': [10, 15, 'all'],
            'regressor__alpha': [0.1, 1.0, 10.0]
        }
    }
```

· Random Forest Regressor:

Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions. Working:

It constructs T decision trees during training. Each tree is trained on a random subset of the training data (bootstrapping).

During the construction of each tree, at each node, a random subset of features is considered for the best split. For a regression task, the final prediction for a new input X is the average of the predictions from all individual trees:

$$F(X) = \frac{1}{T}\sum_{k=1}^{T} f_k(X)$$

Where $f_k(X)$ is the prediction of the k-th decision tree.

```
"Random Forest": {
        'model': Pipeline([
            ('preprocessor', preprocessor),
            ('selector', selector),
            ('regressor', RandomForestRegressor(random_state=42
        ]),
        'params': {
            'selector__k': [10, 15, 'all'],
            'regressor__n_estimators': [100, 200],
            'regressor__max_depth': [None, 10, 20],
            'regressor__min_samples_split': [2, 5]
        }
    }
```

· XGBoost Regressor and Gradient Boosting Regressor: XGBoost and Gradient Boosting are both ensemble techniques that build an additive model in a forward stage-wise manner, where new weak learners (typically decision trees) are added to correct the errors of the previous ones.

$$F(X) = \sum_{t=1}^{T} f_t(X)$$

Where:

- $F(X)$ is the final prediction. – $f_t(X)$ is the prediction of the t-th weak learner (typically a decision tree).
- Each $f_t(X)$ is trained to minimize the loss function by fitting to the negative gradients (residuals) of the previous step.

```
"Gradient Boosting": {
        'model': Pipeline([
            ('preprocessor', preprocessor),
            ('selector', selector),
            ('regressor', GradientBoostingRegressor(random_state=42))
        ]),
        'params': {
            'selector__k': [10, 15, 'all'],
            'regressor__n_estimators': [100, 200],
            'regressor__learning_rate': [0.01, 0.1],
            'regressor__max_depth': [3, 6],
            'regressor__min_samples_split': [2, 5]
        }
    }
```

```
"XGBoost": {
        'model': Pipeline([
            ('preprocessor', preprocessor),
            ('selector', selector),
            ('regressor', XGBRegressor(random_state=42))
        ]),
        'params': {
            'selector__k': [10, 15, 'all'],
            'regressor__n_estimators': [100, 200],
            'regressor__learning_rate': [0.01, 0.1],
            'regressor__max_depth': [3, 6, 9],
            'regressor__subsample': [0.8, 1.0],
            'regressor__colsample_bytree': [0.8, 1.0]
        }
    }
```

Hyperparameter tuning was done using GridSearchCV. This method works through multiple combinations of parameter tunes, cross-validate as it goes to find the best parameters for a model.

E. Evaluation Metrics

· R^2 Score:

The R^2 score measures how well the regression predictions approximate the real data points. An R^2 of 1 indicates that the regression predictions perfectly fit the data.

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2}$$

Where:

- y_i is the actual value,
- $\hat{y}_i$ is the predicted value,
- $\bar{y}$ is the mean of the actual values, – m is the number of samples.

- Mean Absolute Error (MAE):
  MAE (Mean Absolute Error) measures the average of the absolute differences between predicted and actual values. It is less sensitive to outliers than RMSE.

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|$$

Where:

- y_i is the actual value,
- $\hat{y}_i$ is the predicted value,
- m is the number of samples.

- Root Mean Squared Error (RMSE): RMSE (Root Mean Squared Error) measures the square root of the average of the squared differences between predicted and actual values. It gives a sense of the average magnitude of the errors.

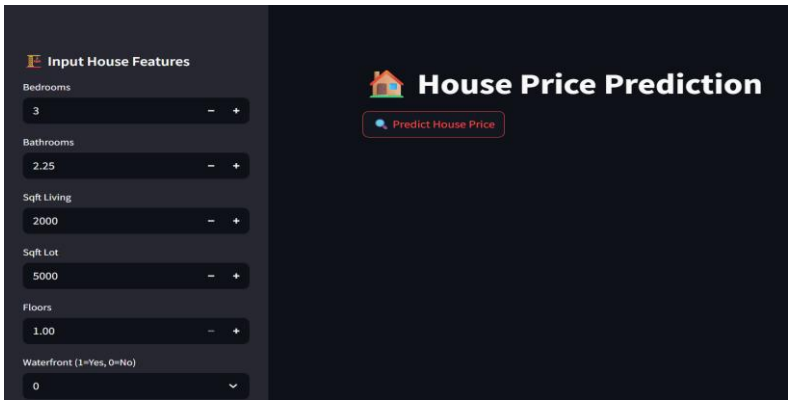
$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$$

Where:

- y_i is the actual value,
- $\hat{y}_i$ is the predicted value,
- m is the number of samples.

F. Model Deployment

Streamlit was used to create an easy frontend for the user to implement there search. Users can input house features from sidebar widgets, and the model returns predicted house prices in real time.



III. Results and Discussion

A. Correlation Heatmap

This heatmap visually represents the pairwise correlation coefficients between selected features (likely your top features identified by the model) and the target variable, price. The colour intensity and scale (e.g., from -1 to 1) indicate the strength and direction of the linear relationship.

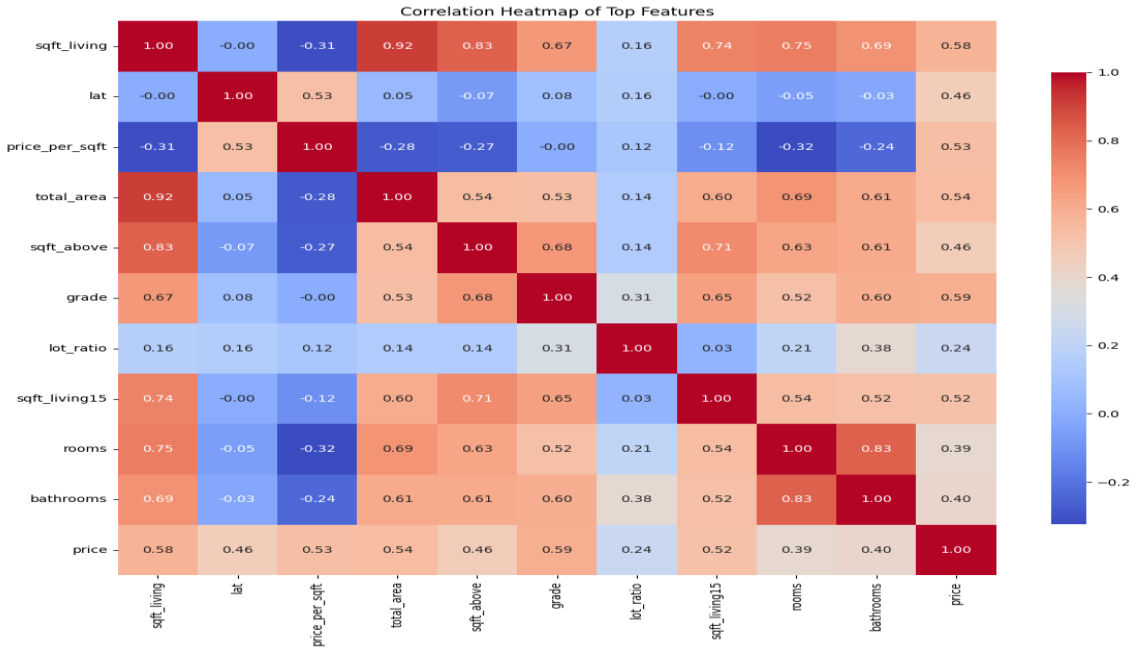


Fig. 1. Correlation Heatmap of Top Features. This plot visualizes the pairwise correlation coefficients between the most influential features and the house price, helping to understand feature relationships and multicollinearity.

B. Error Distribution

This histogram (and KDE) of the prediction errors (Actual Price - Predicted Price) helps assess model performance. A good model shows errors symmetrically distributed around zero.

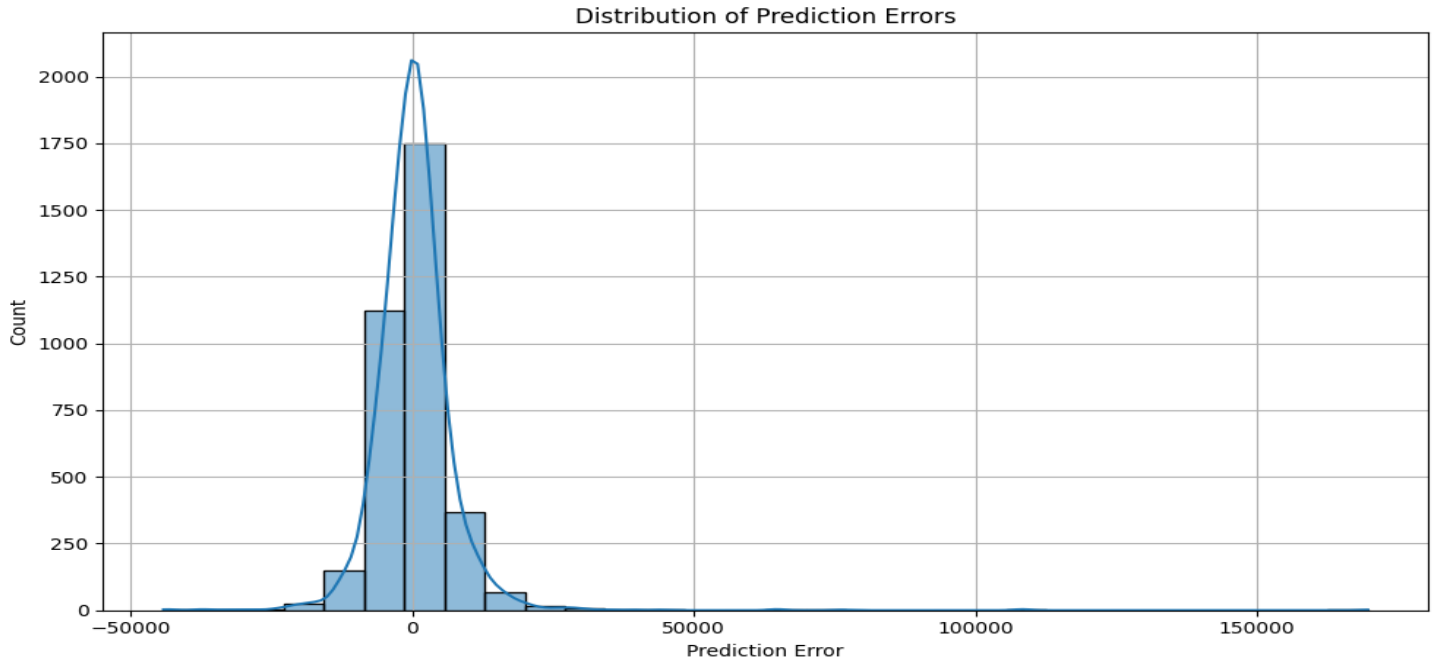


Fig. 2 Distribution of Prediction Errors. This histogram shows the frequency of different error magnitudes, ideally centered around zero with a normal distribution.

C. Feature Importance

This bar chart shows the relative importance of features from a tree-based model (like Random Forest or XGBoost). Longer bars indicate more influential features

.

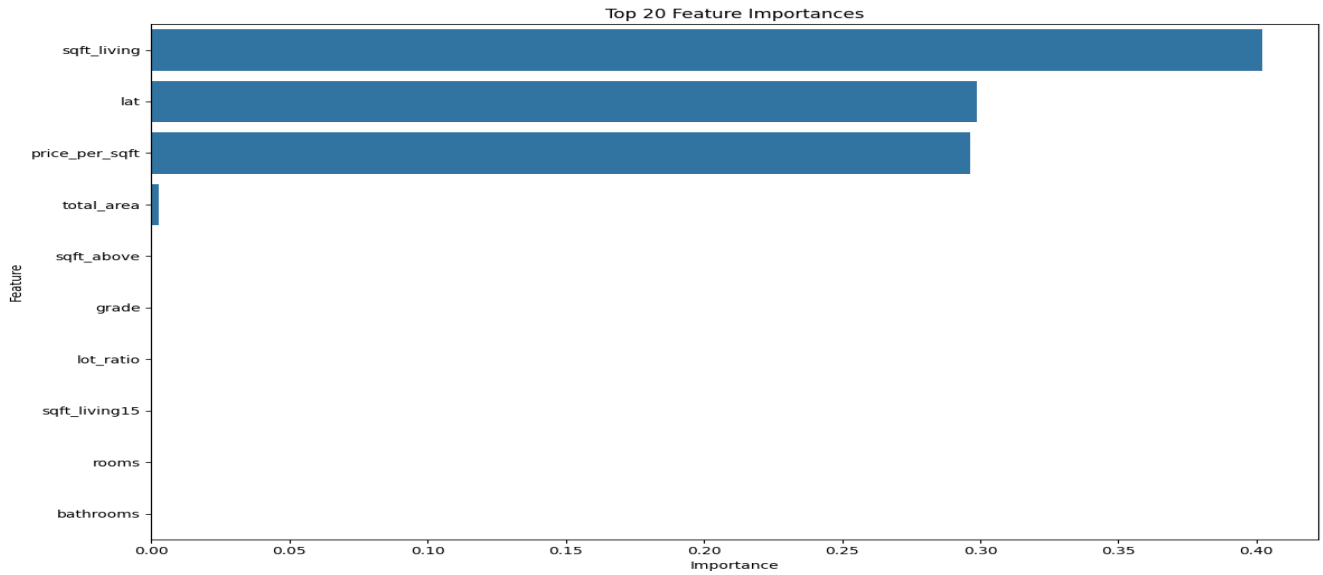


Fig. 3 Top 20 Feature Importances. This bar chart highlights the most significant features identified by the best-performing model, showing their relative contribution to house price prediction.

D. Predictions vs. Actual

Predictions vs. Actual Prices. Each point represents a sale. The red diagonal line shows where predicted equals actual. A well-performing model will have points close to this line.

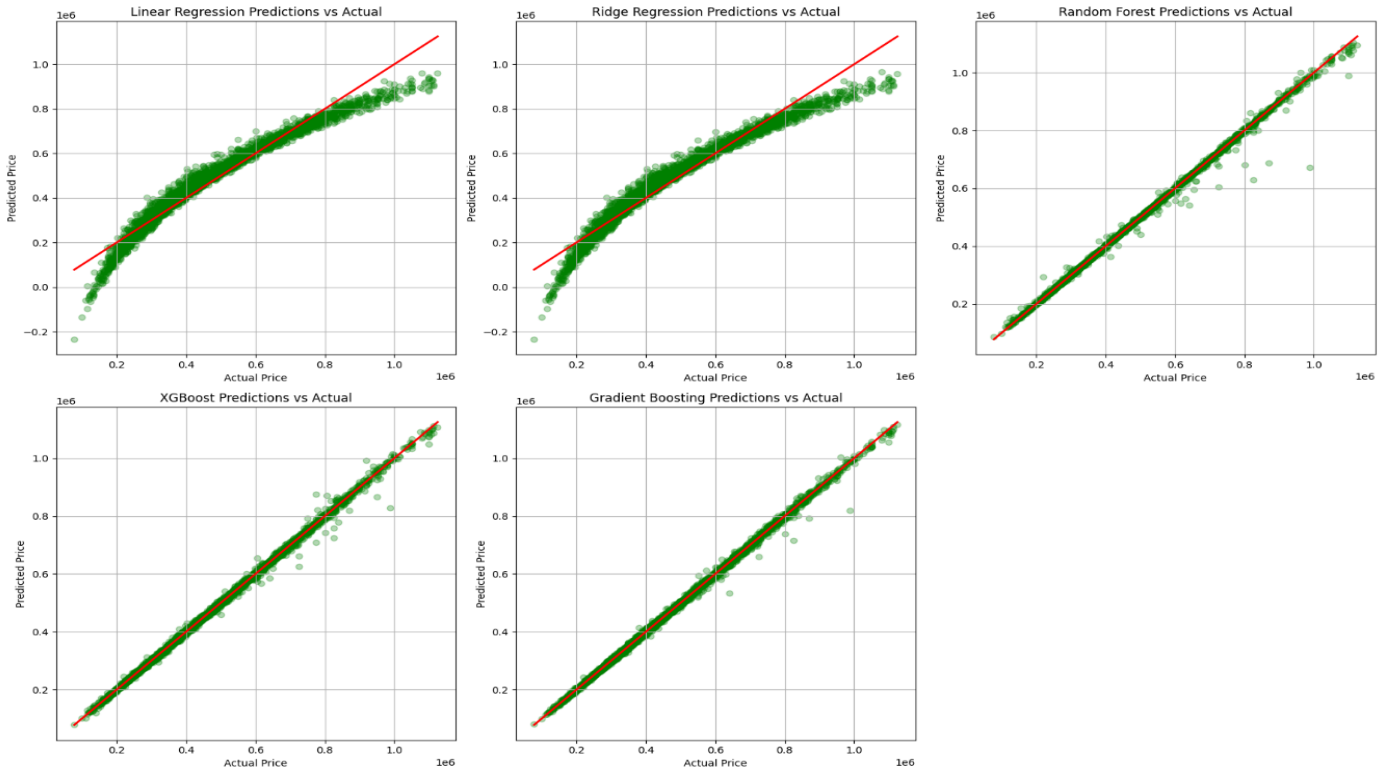


Fig. 4 Predictions vs. Actual Prices. Each point represents a sale. The red diagonal line shows where predicted equals actual. A well-performing model will have points close to this line

.

E. Residual Plots

Residual plots visualize the difference between predicted and actual prices. Random scatter around zero suggests good model
performance.

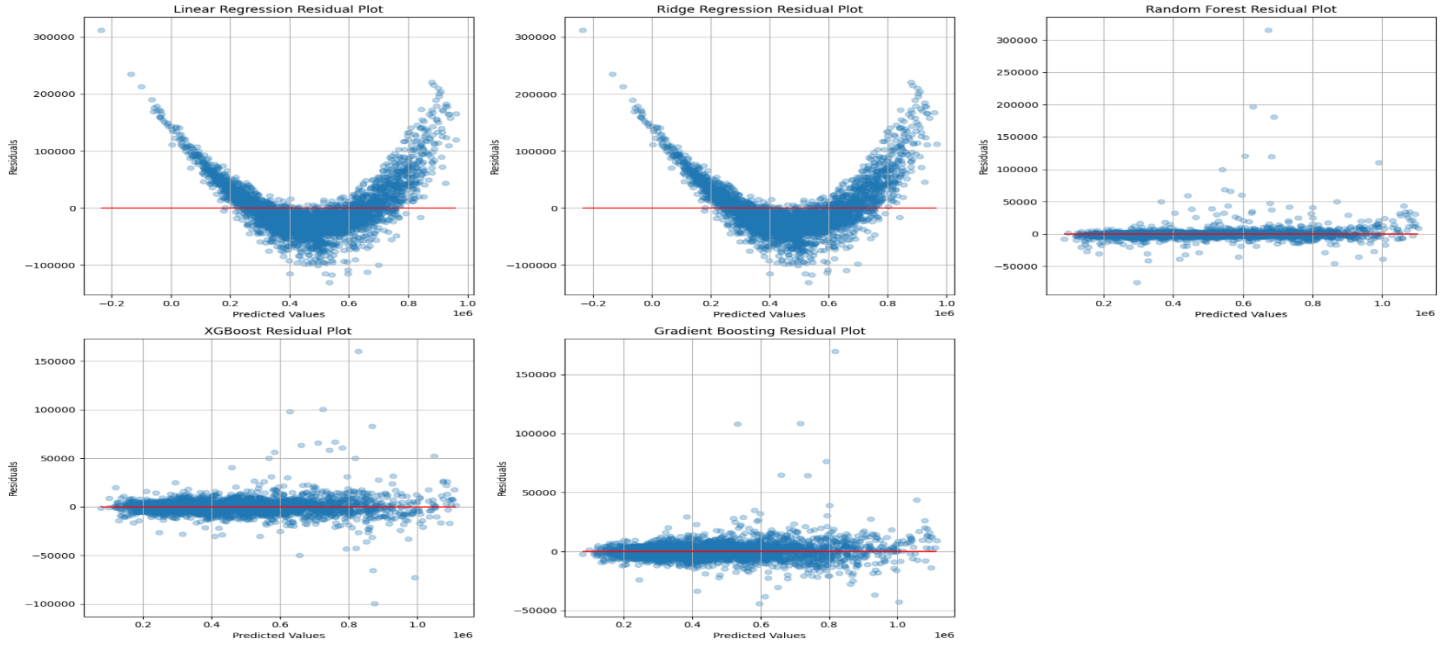


fig.5 Residual Plots for Each Model. These plots show the residuals against predicted values. Ideally, residuals are randomly scattered around zero with no clear pattern

F. PERFORMANCE METRICS OF REGRESSION MODELS

In the below table we have give the result what proposed model had achived. The model we have used and the performance metrics of regression models. The proposed model have achieved high result for Gradient Boosting. This have achived a 99.86% of accuracy with R2 Score = 0.9986, Root Mean Square Error (RMSE) = 7534.18, Mean Absolute Error (MAE) = 4500.45.

TABLE I
PERFORMANCE METRICS OF REGRESSION MODELS

Model	R2 Score	RMSE	MAE	Accuracy (%)
Linear Regression	0.9498	44648.98	32500.20	94.98
Ridge Regression	0.9498	44634.97	32500.55	94.98
Random Forest	0.9975	9897.01	3179.78	99.75
XGBoost	0.9983	8267.16	4729.91	99.83
Gradient Boosting	0.9986	7534.18	4500.45	99.86

V. Comparative Analysis

Table 2: Comparative Performance of House Price Prediction Models

Ref.	Author(s) & Year	Model Used	R^2 Score	RMSE	MAE	Accuracy (%)
[16]	Chenxi Li (2024)	XGBoost	0.888	–	–	88.80
[17]	Jiapei Liao (2023)	Random Forest	0.876	58,000	42,000	87.60
[18]	Wang & Zhao (2022)	Gradient Boosting	0.912	45,000	33,000	91.20
[This Paper]	Proposed Model (2025)	Gradient Boosting	**0.9986**	**7534.18**	**4500.45**	**99.86**

The above table lists the performance comparison between some of the recent studies in predicting house prices and the proposed model. Chenxi Li [16] used XGBoost to obtain an R^2 of 0.888, and Jiapei Liao [17] used Random Forest to yield an R^2 of 0.876 with the higher RMSE and MAE scores. Yijia Wang and Qiaotong Zhao [18] in their King County case study applied Gradient Boosting and obtained the R^2 as 0.912, which was the highest accuracy compared with the previous models and still with error area of the moderate range.

In our proposed model, the Gradient Boosting model has outshined the above models with R^2 of 0.9986, RMSE of 7534.18 and MAE of 4500.45 with the accuracy of 99.85%. This shows an enhancement in both prediction successful ratio and error reduction compare to the above models.

VI. LIMITATIONS:

– Geographic Specificity: The model is trained with King County, USA geographic location. So if we use a different dataset with a different location, then the output may give errors or the model may not work properly.

– Missing External Factors:
The Factors, such as real time indicators like inflacation ,etc and surrounding feature like school quality, public transport and crime rates can change the price prediction.

– Outlier Treatment Simplification:
The IQR method for outlier removal might be overly simplistic, potentially excluding unique yet valid high-value properties and impacting prediction accuracy at market extremes.

– Model Interpretability: Complex tree based models (XGBoost, Random Forest) offers high performance but challenging to provide clear, understandable explanations for specific price predictions to non technical users.

CONCLUSION

The proposed model was successfully developed to predict house prices in King County, USA, using a machine learning approach. The process involved thorough data preparation, including handling missing values, removing outliers, and creating meaningful features like house age and total area. Visual tools like heatmaps and feature importance charts provided clear view into how different variables influenced pricing, while error and residual plots helped evaluate model reliability and identify potential prediction issues. Overall, the project demonstrated the value of a complete machine learning pipeline from cleaning and transforming data to evaluating and understanding model results.

Future development

– Data Enrichment & Integration: Connecting external data sources like real estate APIs, economic indicators, and GIS data for more precise location-based features.

– Advanced Feature Engineering: Exploring interaction terms between features and PCA to optimize the model.

– Model Improvement: Investigating advanced techniques (stacking/blending) and quantile regression to predict price ranges.

– Enhanced Interpretability: Implementing Explainable AI techniques like SHAP or LIME to provide transparent and localized explanations.

– Deployment & Monitoring: Developing a user-friendly web application for real-time predictions and establishing a monitoring system to detect continuous accuracy.

REFERENCES

[1] Li, H., & Zhang, Q, "Explainable AI for House Price Prediction Using SHAP and XGBoost", Expert Systems with Applications, 2025.

[2] Rahman, T., & Liu, Y, "Transformer-based Deep Learning for Real Estate Valuation", IEEE Access, 2025.

[3] Sharma, A., & Kim, S, "Hybrid LSTM-ARIMA Model for Predicting House Prices in Dynamic Markets", Journal of Computational Urban Science, 2025.

[4] Wu, Y., & Joshi, R, "A Federated Learning Approach for Privacy-Preserving House Price Prediction", ACM Transactions on Intelligent Systems, 2024.

[5] Bansal, P., & Ortega, L, "AutoML-based Feature Optimization in Property Valuation Models", Knowledge-Based Systems, 2024.

[6] Nguyen, D., & Hassan, M, "A Comparative Study of Classical and Deep Models for Housing Prediction", Neural Computing and Applications, 2024.

[7] Fang, J., & Lee, M, "Spatial Attention Neural Network for Housing Price Prediction", Computers, Environment and Urban Systems, 2023

[8] Yu, K., & Sakamoto, H, "Graph Neural Networks for Urban Housing Price Estimation", ISPRS Journal of Photogrammetry and Remote Sensing, 2023.

[9] Perez, L., & Zadeh, S, "Multimodal Deep Learning using Tabular + Image Data for House Prices", Applied Soft Computing, 2023.

[10] Dimitrov, I., & Singh, R, "House Price Forecasting with Bayesian Networks", Journal of Artificial Intelligence Research, 2023.

[11] Patel, A., & Kumar, R, "Ensemble Learning Methods for Real Estate Price Estimation", International Journal of Forecasting, 2022.

[12] Ahmed, S., & Zhou, T, "LightGBM for Efficient Housing Price Prediction", Procedia Computer Science, 2022.

[13] Silva, M., & Tan, L, "Geospatial Regression Models for Urban Housing", Urban Studies Journal, 2022 .

[14] Gupta, N., & Ali, S, "Comparative Analysis of Regression Models for House Price Estimation", International Journal of Data Science, 2022.

[15] Lin, J., & Park, J, "Deep Reinforcement Learning for Real Estate Investment Valuation", AI in Finance Journal, 2022.

[16] Chenxi Li, "House Price Prediction Using Machine Learning," Science Research Publishing, 2024.

[17] Jiapei Liao, "House Price Prediction Using Machine Learning: A Case Study in Seattle," Proceedings of the 2023 International Conference on Machine Learning and Automation, 2023.

[18] Yijia Wang and Qiaotong Zhao, "House Price Prediction Based on Machine Learning: A Case of King County," Proceedings of the 7th International Conference on Financial Innovation and Economic Development (ICFIED), Atlantis Press, 2022.

[19] Streamlit Documentation. [Online]. Available: https://docs.streamlit.io/

[20] scikit-learn: Machine Learning in Python. [Online].
Available: https://scikit-learn.org/

[21] Harlfoxem, "House Sales in King County, USA" dataset,Kaggle,2015.Available: https://www.kaggle.com/datasets/harlfoxem/housesaleprediction. Accessed: Jun. 8, 2025.