A CASE STUDY REPORT SUBMITTED AS A PART OF

EXPERIENTIAL LEARNING ON

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

HOUSE PRICE PREDICTION

"MULYAMAAN: A MACHINE LEARNING-BASED HOUSE PRICE PREDICTION SYSTEM WITH INTERACTIVE GUI"

SUBMITTED BY:

GROUP:6

S.no.	NAME	REGD.NO	CRANES REGD.NO
1.	Sudeep Swarankar	2201020138	CL2025010601885234
2.	Arpita priyadarshinee swain	2201020174	CL2025010601879097
3.	Sadiya Mutalib Baksh	2201020242	CL20250106018808114
4.	Ritika Kumari	2201020283	CL2025010601876572
5.	Alisha Kumul	2201020635	CL20250106018814120

ACKNOWLEDGEMENT

We, the members of our project group, would like to extend our sincere gratitude to all those who contributed to the successful completion of this project.

We are especially grateful to our teacher whose expert guidance, valuable feedback, and continuous support were instrumental throughout the duration of this work. Their mentorship provided us with a clear direction and encouraged us to approach challenges with confidence.

Our appreciation extends to each team member for their dedication, cooperation, and collective effort, which were essential to completing this project on time and to a high standard. The collaboration and sense of responsibility shared among us significantly contributed to the success of this endeavor.

Finally, we acknowledge the support and encouragement from our peers, friends, and families, whose understanding helped us remain focused and motivated.

ABSTRACT

The valuation of residential properties is a complex task involving various structural and locational features. Traditional pricing methods often rely on subjective assessment, which can lead to inconsistent results. This paper presents a robust machine learning approach for predicting house prices using multiple regression techniques, supported by thorough preprocessing, feature engineering, and statistical evaluation. Furthermore, the project introduces **MULYAMAAN**, an interactive graphical user interface (GUI) that facilitates real-time price estimation for end-users. The model's performance is evaluated across multiple regression techniques, including Linear, Ridge, Lasso, ElasticNet, and Polynomial Regression, with Ridge Regression yielding the best performance.

CONTENT

INTRODUCTION

The prediction of house prices is a critical task in the real estate industry, where accurate valuation directly impacts property investment, buying decisions, and financial planning. Traditionally, estimating the price of a house involved manual assessments based on experience and market comparisons. However, with the increasing availability of real estate data and advancements in machine learning, predictive modeling has become a powerful and more objective alternative to traditional methods.

In this project, we explore the use of machine learning algorithms to predict housing prices based on various features such as area, number of bedrooms and bathrooms, number of floors, and the presence of amenities like air conditioning, guest rooms, and more. The dataset used in this study contains a mix of numerical and categorical variables, providing a realistic representation of the housing market.

The primary goal is to develop a regression model that can predict house prices with high accuracy. To achieve this, we first perform extensive data exploration and preprocessing, including handling missing values, encoding categorical features, and normalizing numerical data. We then apply feature selection techniques to identify the most relevant predictors for the target variable—house price.

Real estate pricing plays a pivotal role in economic planning and property investment. With the availability of structured housing data, machine learning techniques offer a data-driven approach to estimate house prices more accurately. This paper explores an end-to-end solution for house price prediction that includes data preprocessing, exploratory data analysis, model building, and GUI deployment.

PROBLEM STATEMENT

The real estate market is one of the most dynamic and influential sectors in the global economy, where accurate property valuation plays a crucial role for buyers, sellers, investors, and financial institutions. However, traditional methods of estimating house prices often rely on manual assessments, subjective judgments, and historical trends, which may lead to inconsistencies, inaccuracies, and inefficiencies.

With the increasing availability of large-scale housing datasets and advancements in machine learning, there exists a significant opportunity to enhance the precision and reliability of house price predictions. Nevertheless, the challenge lies in building a model that can effectively capture the complex relationships between various factors influencing house prices—such as location, area, number of rooms, age of the property, nearby amenities, and current market trends.

This project aims to address this challenge by developing a data-driven, predictive model for estimating house prices using machine learning techniques. The proposed solution will analyze historical housing data, extract meaningful features, and apply suitable regression algorithms to predict property values with improved accuracy. The model will also be evaluated on the basis of its performance, scalability, and potential for real-world application.

By leveraging predictive analytics, this project seeks to provide a more objective, automated, and insightful approach to house price estimation, ultimately aiding stakeholders in making more informed and timely decisions.

PROJECT DESCRIPTION

Dataset Overview:

- Dataset Used: Housing.csv
- Total Samples: 2180
- Target Variable: price (Market value of houses in INR)
- Input Features (23):
    - Numerical: area
    - Categorical/Binary (encoded):
        - Structural: bedrooms, bathrooms, stories, parking
        - Amenities: mainroad, guestroom, basement, hotwaterheating, airconditioning, prefarea
        - Furnishing: furnishingstatus

1	price	area	bedrooms	bathroom:	stories	mainroad	guestroor	basement	hotwaterh	airconditic	parking	prefarea	furnishingstatus
2	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
3	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
4	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
5	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
6	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished
7	10850000	7500	3	3	1	yes	no	yes	no	yes	2	yes	semi-furnished
8	10150000	8580	4	3	4	yes	no	no	no	yes	2	yes	semi-furnished
9	10150000	16200	5	3	2	yes	no	no	no	no	0	no	unfurnished
10	9870000	8100	4	1	2	yes	yes	yes	no	yes	2	yes	furnished
11	9800000	5750	3	2	4	yes	yes	no	no	yes	1	yes	unfurnished
12	9800000	13200	3	1	2	yes	no	yes	no	yes	2	yes	furnished
13	9681000	6000	4	3	2	yes	yes	yes	yes	no	2	no	semi-furnished
14	9310000	6550	4	2	2	yes	no	no	no	yes	1	yes	semi-furnished
15	9240000	3500	4	2	2	yes	no	no	yes	no	2	no	furnished
16	9240000	7800	3	2	2	yes	no	no	no	no	0	yes	semi-furnished
17	9100000	6000	4	1	2	yes	no	yes	no	no	2	no	semi-furnished
18	9100000	6600	4	2	2	yes	yes	yes	no	yes	1	yes	unfurnished
19	8960000	8500	3	2	4	yes	no	no	no	yes	2	no	furnished
20	8890000	4600	3	2	2	yes	yes	no	no	yes	2	no	furnished

Data Preprocessing:

Data Cleaning:-

- Verified column data types using df.info().
- Checked for duplicates and removed them.
- Missing values: Verified using df.isnull().sum() — none found.

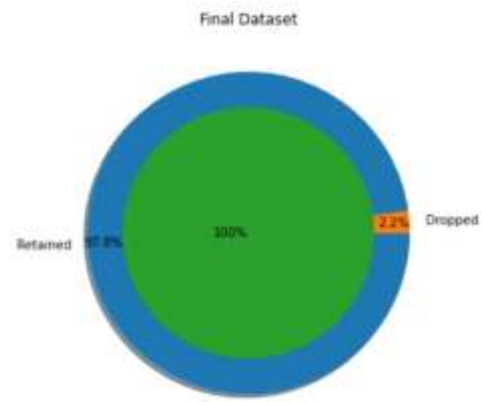
Categorical Encoding:-

- One-Hot Encoding: For binary categorical features (e.g., mainroad, guestroom)
- Dummy Variable Encoding: For multi-class categorical features (e.g., furnishingstatus)

```
One-Hot Encoding on features:
hotwaterheating
basement
guestroom
mainroad
prefarea
airconditioning

Dummy Encoding on features:
furnishingstatus
bathrooms
stories
parking
bedrooms
```

Outlier Detection & Removal:-



- Used Interquartile Range (IQR) method for numerical features.
- Approximately ~2.2% data was dropped to improve model quality.

Feature Standardization:-

- Applied StandardScaler from sklearn.preprocessing to normalize feature values.

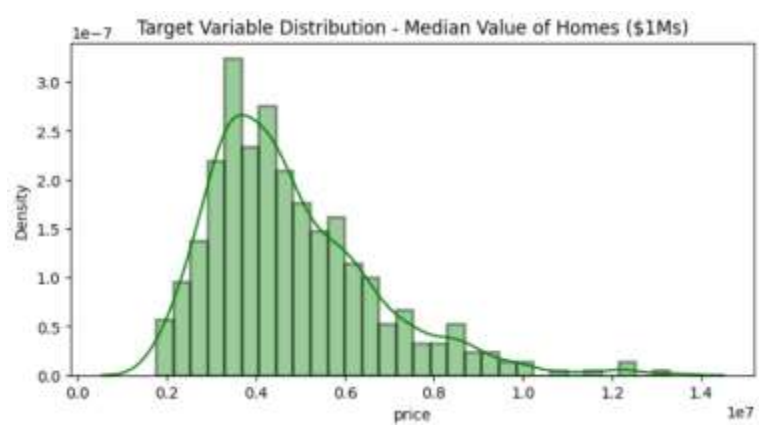
```
RangeIndex: 533 entries, 0 to 532
Data columns (total 24 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   price                            533 non-null    int64
 1   area                             533 non-null    int64
 2   mainroad                         533 non-null    bool
 3   guestroom                        533 non-null    bool
 4   basement                         533 non-null    bool
 5   hotwaterheating                  533 non-null    bool
 6   airconditioning                  533 non-null    bool
 7   prefarea                         533 non-null    bool
 8   furnishingstatus_semi-furnished  533 non-null    bool
 9   furnishingstatus_unfurnished     533 non-null    bool
 10  bathrooms_2                      533 non-null    bool
 11  bathrooms_3                      533 non-null    bool
 12  bathrooms_4                      533 non-null    bool
 13  stories_2                        533 non-null    bool
 14  stories_3                        533 non-null    bool
 15  stories_4                        533 non-null    bool
 16  parking_1                        533 non-null    bool
 17  parking_2                        533 non-null    bool
 18  parking_3                        533 non-null    bool
 19  bedrooms_2                       533 non-null    bool
 20  bedrooms_3                       533 non-null    bool
 21  bedrooms_4                       533 non-null    bool
 22  bedrooms_5                       533 non-null    bool
 23  bedrooms_6                       533 non-null    bool
dtypes: bool(22), int64(2)
memory usage: 19.9 KB
```

All columns after data preprocessing and Standardization

Exploratory Data Analysis (EDA):

Target Variable Distribution:-

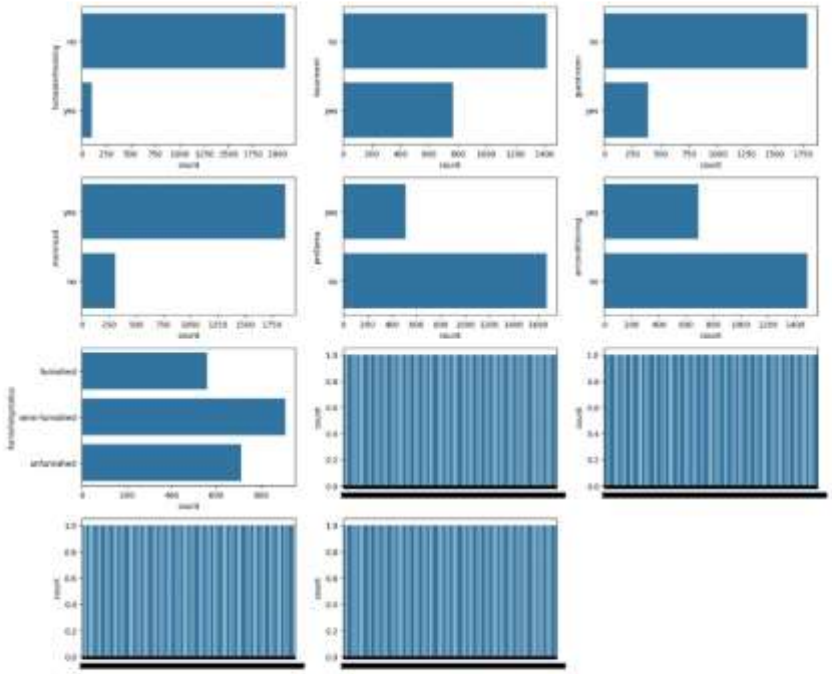
- Used sns.distplot() to plot house price distribution.
- Skewed right, indicating presence of high-valued properties.



Target vs Density graph to visualize distribution and averaging

Feature Distribution:-

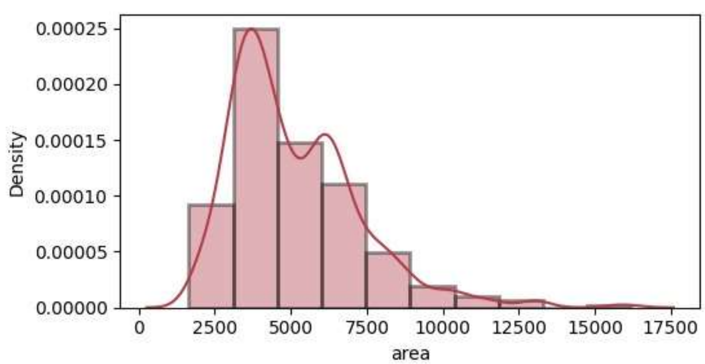
- Boxplots and histograms for all numerical features.
- Count plots for categorical features.



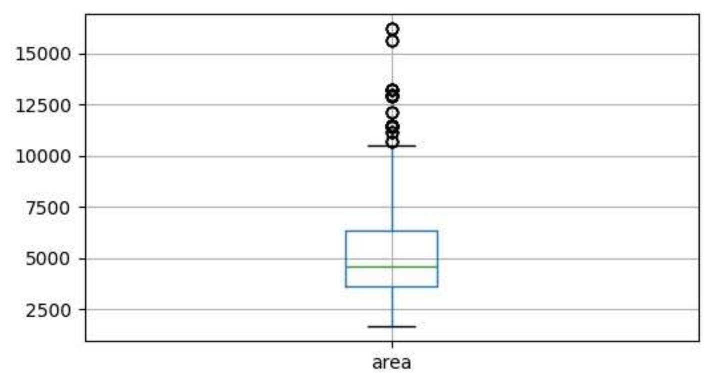
Graph for visualizing the categorical features

Feature Relationships:-

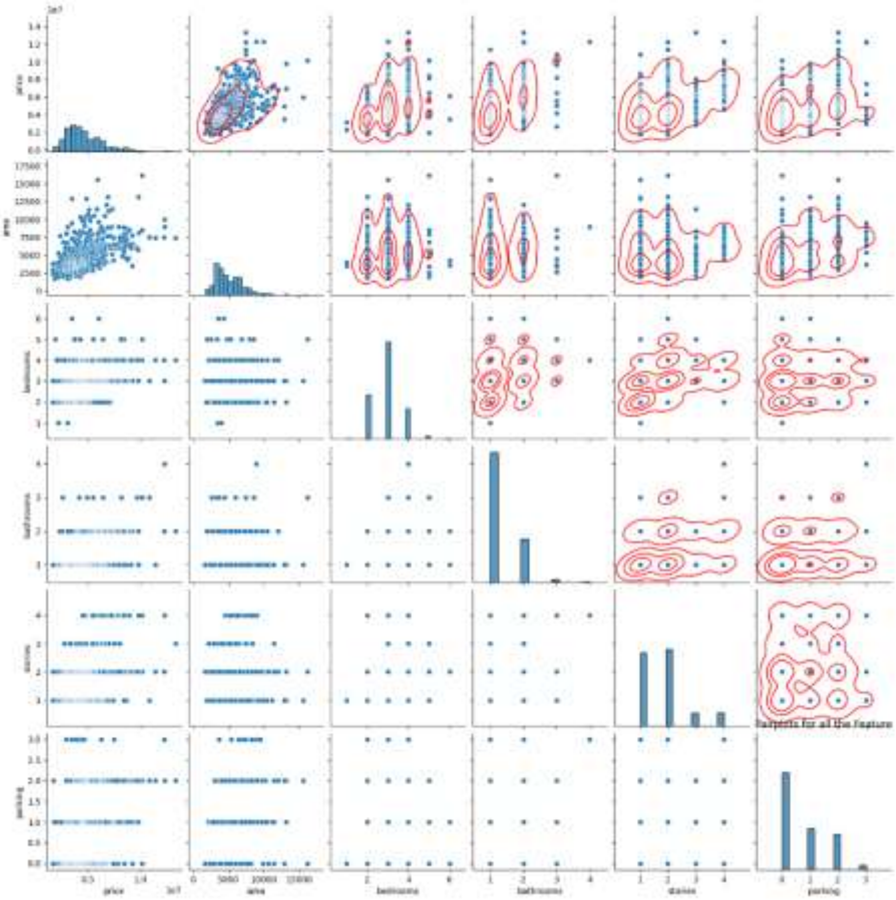
- Used sns.pairplot() to visualize multivariate relationships.
- Heatmap to analyze multicollinearity.



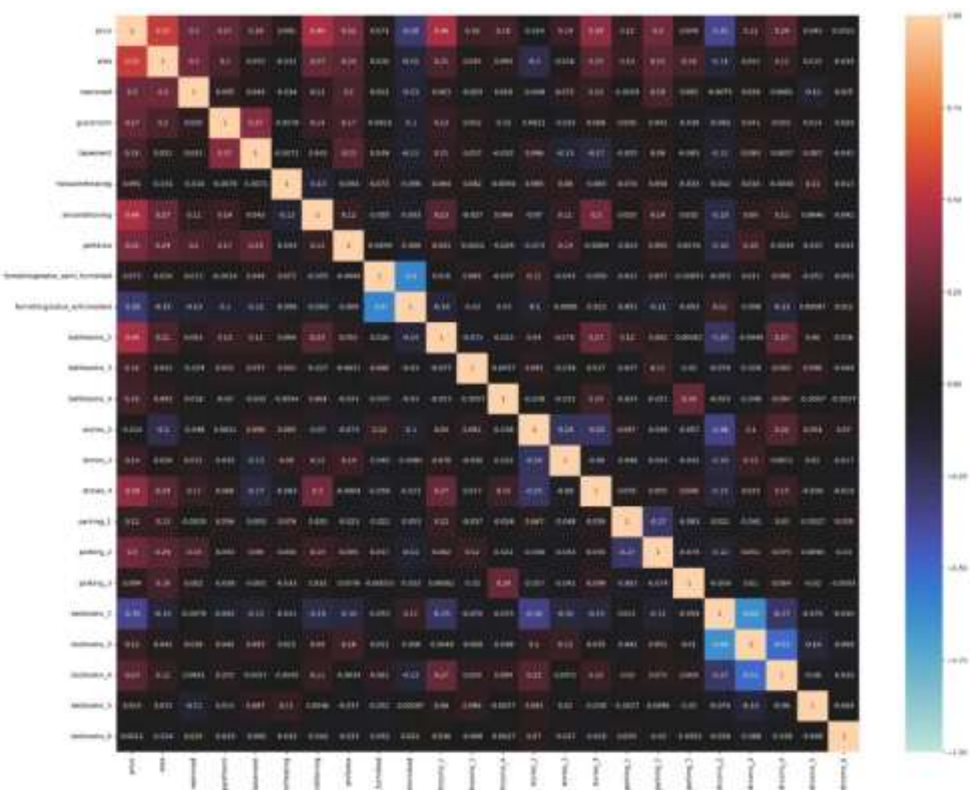
Density vs Area graph to visualize numeric features distribution



Graph for visualization of outliers



Visualization for understanding the relationship between all the features

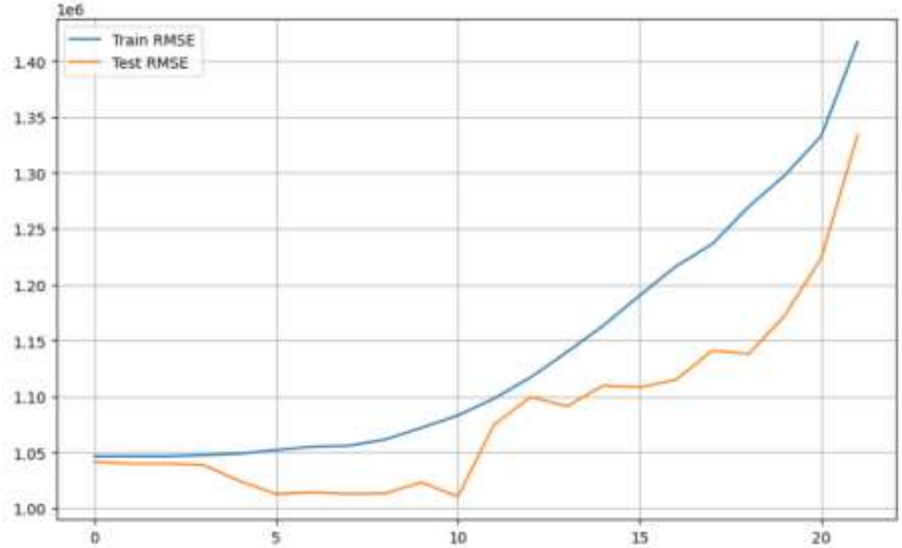


Correlation matrix

Feature Engineering:

Recursive Feature Elimination (RFE):-

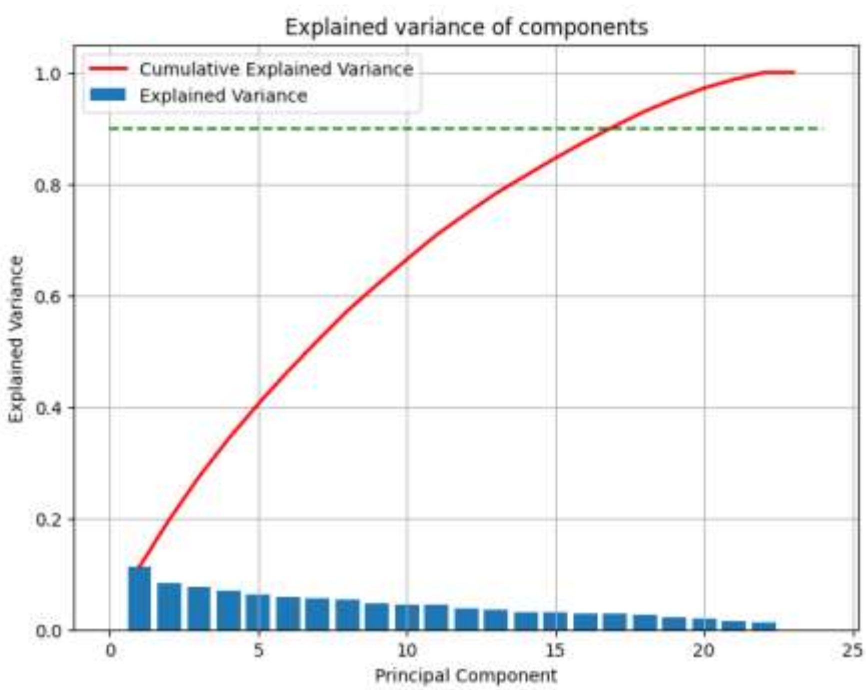
- Used to select the most significant predictors by recursively removing the least important features.



Root mean square error(RMSE) of test and train data

Principal Component Analysis (PCA):-

- Applied to reduce dimensionality while preserving variance.
- Explained variance ratio plotted to determine ideal number of components.

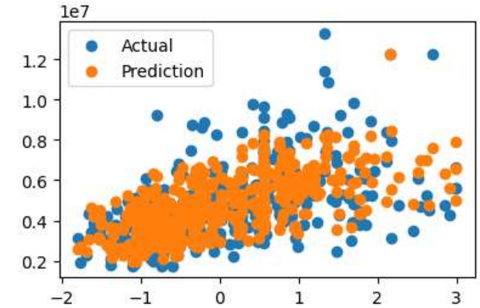


Model Building & Evaluation:

Models Implemented:-

- **Multiple Linear Regression (MLR)**

Evaluating Multiple Linear Regression Model



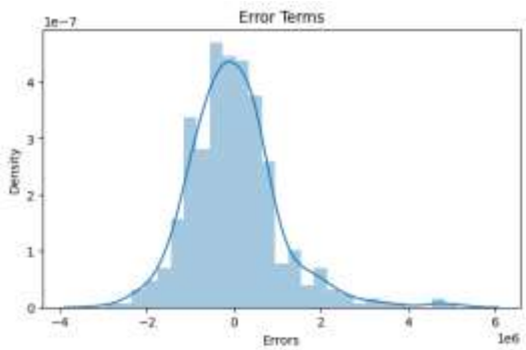
- The Intercept of the Regresion Model was found to be 4716708.779342723

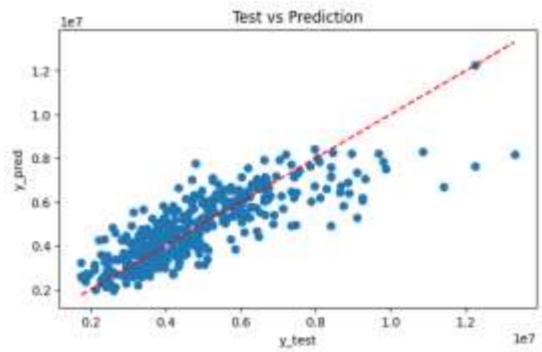
Training Set Metrics

- R2-Score on Training set ---> 0.6789097089550895
- Residual Sum of Squares (RSS) on Training set ---> 466429810296572.75
- Mean Squared Error (MSE) on Training set ---> 1094905657973.1757
- Root Mean Squared Error (RMSE) on Training set ---> 1046377.3974877209

Testing Set Metrics

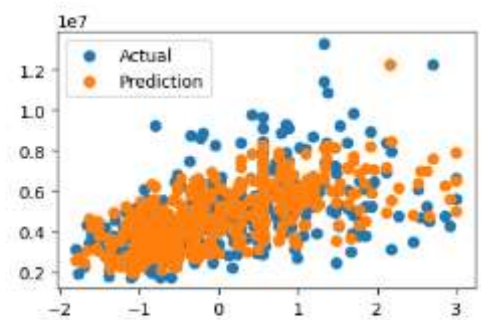
- R2-Score on Testing set ---> 0.6866794976385519
- Residual Sum of Squares (RSS) on Testing set ---> 116042808105904.88
- Mean Squared Error (MSE) on Testing set ---> 1084512225288.8306
- Root Mean Squared Error (RMSE) on Testing set ---> 1041399.1671250896





- **Ridge Regression (RLR) – L2 Regularization**

Evaluating Ridge Regression Model



- The Intercept of the Regresion Model was found to be 4716708.779342723

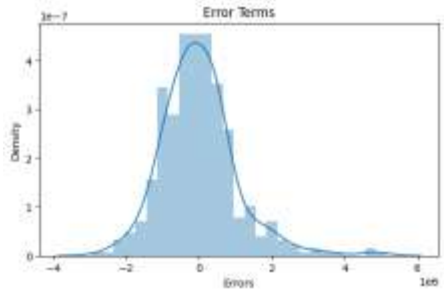
Training Set Metrics

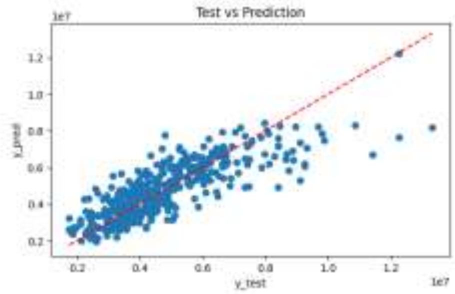
- R2-Score on Training set ---> 0.6789060979912986
- Residual Sum of Squares (RSS) on Training set ---> 466435055740620.5
- Mean Squared Error (MSE) on Training set ---> 1094917971222.1139
- Root Mean Squared Error (RMSE) on Training set ---> 1046383.28122251

Testing Set Metrics

- R2-Score on Testing set ---> 0.6868090228048058
- Residual Sum of Squares (RSS) on Testing set ---> 115994836575477.69
- Mean Squared Error (MSE) on Testing set ---> 1084063893228.7633

- Root Mean Squared Error (RMSE) on Testing set ---> 1041183.8902080473

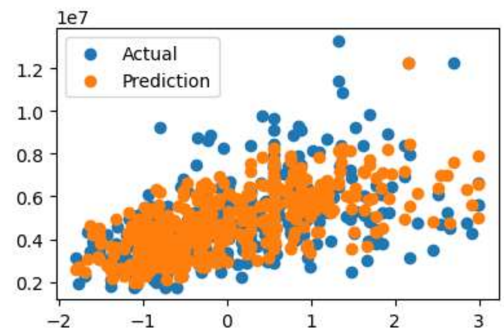




- **Lasso Regression (LLR) – L1 Regularization**

<u>Evaluating Lasso Regression Model</u>

- The Intercept of the Regresion Model was found to be 4716708.779342723

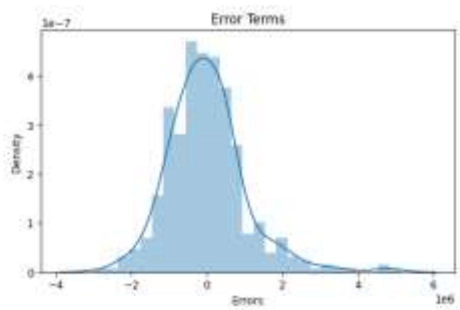


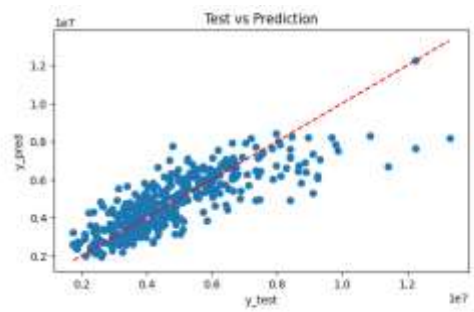
<u>Training Set Metrics</u>

- R2-Score on Training set ---> 0.6789097088318172
- Residual Sum of Squares (RSS) on Training set ---> 466429810475643.3
- Mean Squared Error (MSE) on Training set ---> 1094905658393.529
- Root Mean Squared Error (RMSE) on Training set ---> 1046377.3976885821v

<u>Testing Set Metrics</u>

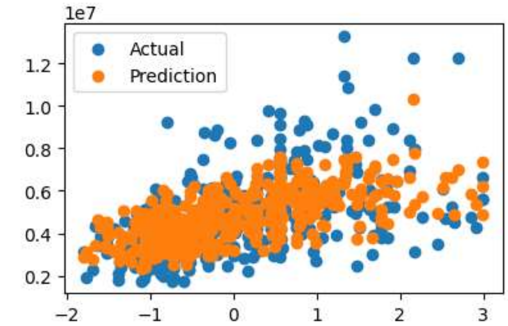
- R2-Score on Testing set ---> 0.6866804541048077
- Residual Sum of Squares (RSS) on Testing set ---> 116042453864706.62
- Mean Squared Error (MSE) on Testing set ---> 1084508914623.4265
- Root Mean Squared Error (RMSE) on Testing set ---> 1041397.5775962927





- **ElasticNet Regression (ENR) – Combination of L1 & L2**

Evaluating Elastic-Net Regression Model



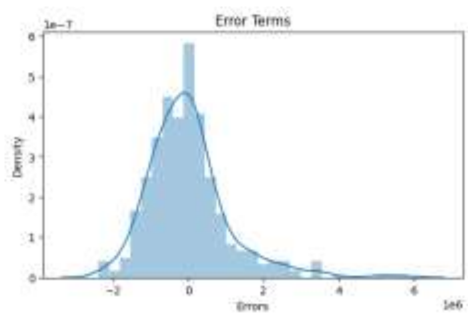
- The Intercept of the Regresion Model was found to be 4716708.779342723

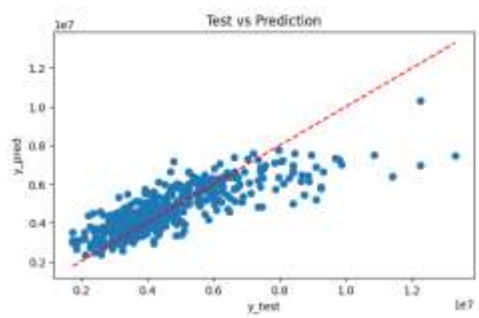
Training Set Metrics

- R2-Score on Training set ---> 0.6518476052536579
- Residual Sum of Squares (RSS) on Training set ---> 505741406591209.25
- Mean Squared Error (MSE) on Training set ---> 1187186400448.8481
- Root Mean Squared Error (RMSE) on Training set ---> 1089580.8370418637

Testing Set Metrics

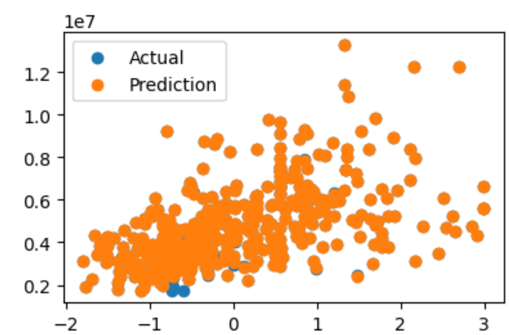
- R2-Score on Testing set ---> 0.673916682711091
- Residual Sum of Squares (RSS) on Testing set ---> 120769702363881.02
- Mean Squared Error (MSE) on Testing set ---> 1128688807139.0747
- Root Mean Squared Error (RMSE) on Testing set ---> 1062397.6690199743





- **Polynomial Regression (PNR) – Degree 5 polynomial terms**

Evaluating Polynomial Regression Model



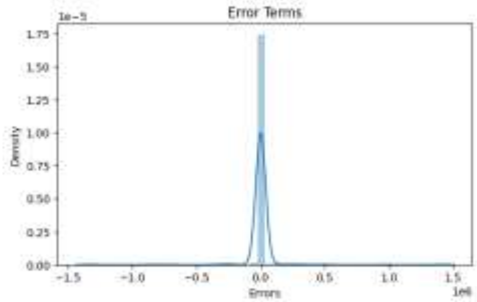
- The Intercept of the Regresion Model was found to be 4716708.779342723

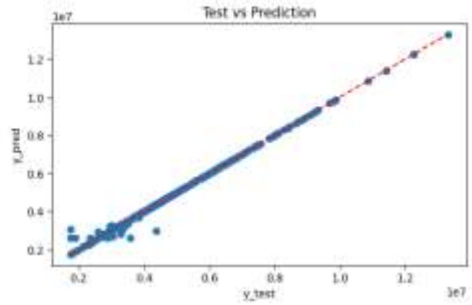
Training Set Metrics

- R2-Score on Training set ---> 0.9953406460321895
- Residual Sum of Squares (RSS) on Training set ---> 6768381504897.227
- Mean Squared Error (MSE) on Training set ---> 15888219495.063913
- Root Mean Squared Error (RMSE) on Training set ---> 126048.48073286688

Testing Set Metrics

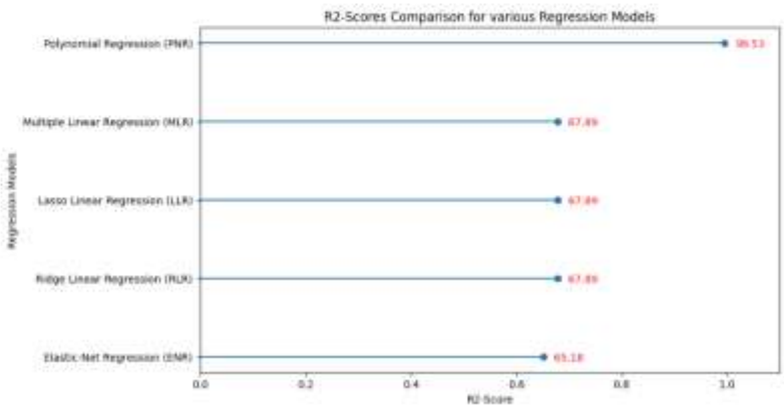
- R2-Score on Testing set ---> -1.0475402605745437e+18
- Residual Sum of Squares (RSS) on Testing set ---> 3.879717813704693e+32
- Mean Squared Error (MSE) on Testing set ---> 3.625904498789433e+30
- Root Mean Squared Error (RMSE) on Testing set ---> 1904180794669832.0

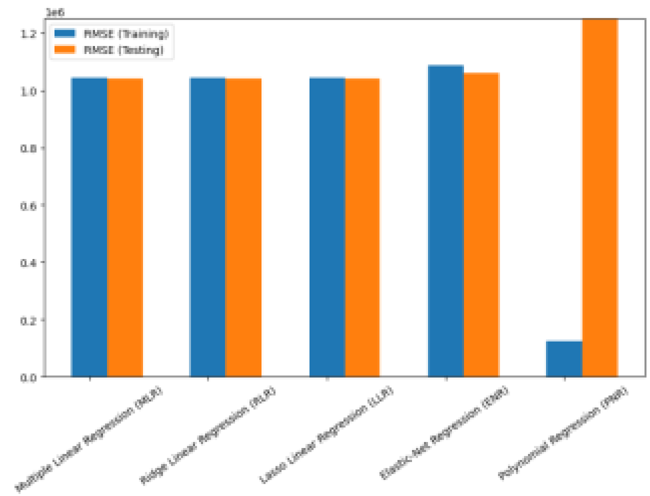




Model Metrics:-

- R^2 Score – Goodness of fit
- RSS (Residual Sum of Squares) – Unexplained variance
- MSE (Mean Squared Error) – Average squared error
- RMSE (Root Mean Squared Error) – Model's absolute error





Model	Train R^2	Test R^2	Train RMSE	Test RMSE
Linear Regression (MLR)	~0.66	~0.64	Moderate	Moderate
Ridge Regression (RLR)	Higher	Higher	Lower	Lower
Lasso Regression (LLR)	Slightly Lower	Moderate	Moderate	Moderate
ElasticNet (ENR)	Similar to Lasso	Similar	Moderate	Moderate
Polynomial Regression	Highest	Good	Lowest	Slight overfitting

GUI Development – MULYAMAAN

Purpose

To offer a real-time house price prediction interface to users without requiring programming knowledge.

Tools Used

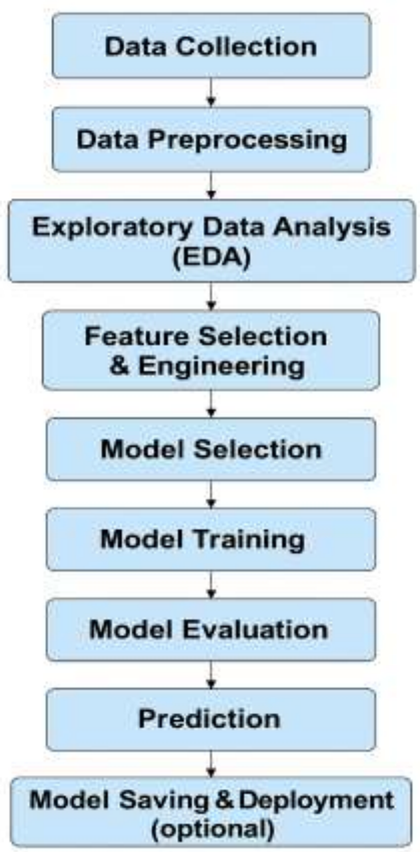
- Tkinter: GUI library in Python
- Joblib: For loading trained model and scaler
- Pandas: To format input features for model prediction

Features

- Tabbed Interface: Divided into "Basic Features" and "Amenities"
- Modern Gradient Styling: Custom gradient background using canvas
- Input Controls:
    - Entry boxes, Spinboxes, Radio Buttons
    - All fields strictly validated
- Prediction Animation: Result displayed with live counter effect

Application Flow

1. **User Inputs** property attributes.
2. **Inputs transformed** into a dataframe with matching feature order.
3. **Feature Scaling** applied using saved StandardScaler.
4. **Prediction made** using saved LinearRegression model.
5. **Result displayed** in GUI.



Results Snapshot

- Visual comparison of actual vs predicted prices
- Distribution of errors plotted
- GUI screenshots for input and result

Model Persistence

- Finalized model and scaler saved as:
    - House_prediction.pkl
    - scaler.pkl
- Loaded into GUI for prediction at runtime.

CHALLENGES FACED

1. Data Quality and Preprocessing

The initial dataset contained missing values, inconsistent entries, and irrelevant features, which posed significant challenges during the data cleaning and preprocessing phase. Ensuring data completeness and correctness was essential for building a reliable model.

2. Feature Selection and Engineering

Identifying which features had the most impact on house prices required careful analysis. Creating new features (feature engineering) and reducing dimensionality without losing critical information were complex and time-consuming processes.

3. Model Selection and Tuning

Choosing the appropriate regression algorithms (e.g., Linear Regression, Decision Tree, Random Forest, XGBoost) involved multiple trials and evaluations. Hyperparameter tuning for optimizing performance added to the complexity.

4. Handling Multicollinearity

Some independent variables were found to be highly correlated, which could skew the predictions. Techniques such as correlation analysis and Variance Inflation Factor (VIF) were needed to manage multicollinearity.

5. Overfitting and Underfitting

Balancing the model to perform well on both training and testing data was a major challenge. Several models initially overfit the data, necessitating regularization techniques and cross-validation.

6. Interpretability vs. Accuracy

More complex models often offered higher accuracy but at the cost of interpretability. Striking the right balance between explainability and predictive power was essential, especially for real-world applications.

7. Limited Computational Resources

Training some models, especially ensemble methods, was computationally expensive and time-consuming. Working within limited hardware constraints required efficient resource management.

FUTURE ENHANCEMENT

1. Feature Selection and Engineering
Identifying which features had the most impact on house prices required careful analysis. Creating new features (feature engineering) and reducing dimensionality without losing critical information were complex and time-consuming processes.

2. Model Selection and Tuning
Choosing the appropriate regression algorithms (e.g., Linear Regression, Decision Tree, Random Forest, XGBoost) involved multiple trials and evaluations. Hyperparameter tuning for optimizing performance added to the complexity.

3. Handling Multicollinearity
Some independent variables were found to be highly correlated, which could skew the predictions. Techniques such as correlation analysis and Variance Inflation Factor (VIF) were needed to manage multicollinearity.

4. Overfitting and Underfitting
Balancing the model to perform well on both training and testing data was a major challenge. Several models included model serialization, API setup, and ensuring responsiveness of the deployed system.initially overfit the data, necessitating regularization techniques and cross-validation.

5. Interpretability vs. Accuracy
More complex models often offered higher accuracy but at the cost of interpretability. Striking the right balance between explainability and predictive power was essential, especially for real-world applications.

6. Limited Computational Resources
Training some models, especially ensemble methods, was computationally expensive and time-consuming. Working within limited hardware constraints required efficient resource management.

7. Deployment and Integration (if applicable)
If the model was integrated into a user interface or web application, challenges included model serialization, API setup, and ensuring responsiveness of the deployed system

CONCLUSION

The House Price Prediction project demonstrates the effective use of machine learning techniques to estimate property prices based on a variety of influential factors. Through detailed data exploration, preprocessing, and feature selection, we ensured the dataset was clean, consistent, and ready for model training. Various regression algorithms—including Linear Regression, Ridge, Lasso, ElasticNet, and ensemble methods—were implemented and compared using appropriate performance metrics such as R^2 score, Mean Absolute Error (MAE), and Mean Squared Error (MSE).

The results revealed that regularized models and ensemble techniques generally offered improved accuracy and robustness over basic linear regression. This emphasizes the importance of selecting the right algorithm and fine-tuning it for the specific dataset and use case.

Overall, this project highlights how data-driven approaches can significantly enhance the accuracy and reliability of real estate price predictions. Such models can serve as practical decision-support tools for home buyers, sellers, real estate agents, and financial institutions. Future work can involve incorporating real-time market data, geographic mapping, and advanced deep learning models to further refine prediction accuracy and usability.

REFERENCES

- Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research.*
- James, G., et al. (2013). *An Introduction to Statistical Learning.* Springer.
- Python Software Foundation. Tkinter documentation.
- Housing dataset - Source (local collection)
- Seaborn, Matplotlib, and Pandas official documentation.
- https://scikit-learn.org/stable/
- https://scikit-learn.org/stable/
- https://docs.python.org/3/library/tkinter.html
- https://pandas.pydata.org/docs/

SOURCE CODE LINK:-

https://github.com/SudeepSwarankar/HOUSE-PRICE-PREDICTION-APP/tree/main/house%20priceing