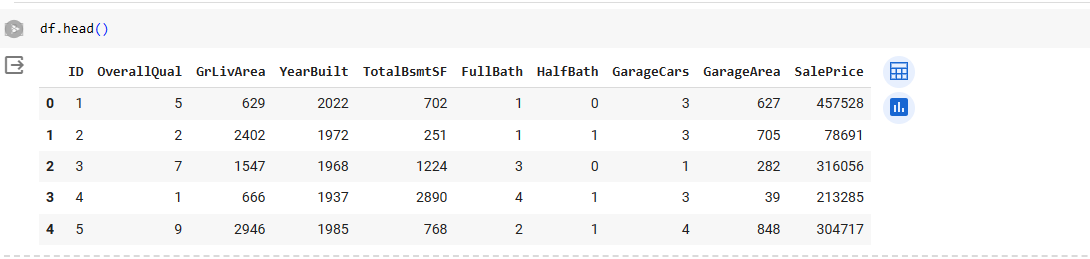
# **House Price Prediction: Exploring Regression Models for Real Estate Valuation**

**Task 1: Data Preprocessing**  
  
A CSV dataset containing both numerical and categorical features. Tasks are to:  
  
Load the dataset using Python.  
Perform data cleaning and preprocessing, including handling missing values and encoding categorical variables.  
Split the data into training and testing sets.

**Task 2: Regression**  
  
Given a dataset of house prices, build a regression model to predict house prices. You should:  
  
Choose an appropriate regression algorithm and justify your choice.  
Train the model on the training data.  
Evaluate the model's performance using relevant metrics.  
Provide a brief explanation of your model's predictions and suggest improvements.



|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Contents** | **Page No.** |
| 1. | Introduction And Objectives | 3 |
| 2. | About The Dataset | 4 |
| 3. | Task 1 Description | 4 |
|  | 3.1 Libraries Used | 5 |
|  | 3.2 Data Cleansing And Pre-processing | 6 |
|  | 3.2 Data Splitting | 8 |
| 4. | Task 2 Description | 10 |
|  | 4.1 Model Building | 11 |
|  | 4.2 Choose Best Model | 12 |
|  | 4.3 Train The Model And Use Proper Evaluation Metrics | 12 |
| 5. | Explanation Of Model Prediction | 14 |
|  | 5.1 Suggest Improvements | 14 |

INDEX

Introduction:-

House price prediction is a fundamental task in real estate and finance industries, as well as in academic research. It involves predicting the selling price of residential properties based on various factors such as location, size, amenities, and market trends. Accurate house price prediction is crucial for homeowners, real estate agents, investors, and policymakers to make informed decisions regarding buying, selling, and investing in properties.

In recent years, the availability of large-scale housing datasets, coupled with advancements in machine learning and data analytics techniques, has led to the development of sophisticated predictive models for house price estimation. These models leverage diverse features such as property characteristics, neighborhood attributes, economic indicators, and historical sales data to generate accurate price predictions.

House price prediction models employ a range of regression techniques, including linear regression, decision trees, random forests, support vector regression, and gradient boosting, among others. These models are trained on historical housing data with known prices and corresponding features, and then validated and fine-tuned to ensure robust performance on unseen data.

The predictive accuracy of house price prediction models depends on several factors, including the quality and relevance of input features, the size and representativeness of the training data, the choice of modeling algorithms, and the effectiveness of model evaluation and validation strategies. Additionally, domain knowledge, expertise in data preprocessing, feature engineering, and model interpretation play essential roles in building reliable and interpretable predictive models.

House price prediction has numerous practical applications, including assisting homebuyers and sellers in pricing negotiations, guiding real estate investments, assessing property valuation for mortgage lending, and informing urban planning and housing policy decisions. As the real estate market continues to evolve, accurate and reliable house price prediction models will remain invaluable tools for stakeholders across the housing ecosystem.

Objectives:-

* Load the dataset using Python.
* Perform data cleaning and preprocessing, including handling missing values and encoding categorical variables.
* Split the data into training and testing sets.
* Build a regression model to predict house prices.
* Choose an appropriate regression algorithm and justify your choice.
* Train the model on the training data.
* Evaluate the model's performance using relevant metrics.
* Provide a brief explanation of your model's predictions and suggest improvements.

About The Dataset :-

* ID: A unique identifier for each observation.
* OverallQual: Overall material and finish quality rating of the house, rated on a scale from 1 to 10.
* GrLivArea: Above ground living area in square feet.
* YearBuilt: The year the house was built.
* TotalBsmtSF: Total basement area in square feet.
* FullBath: Number of full bathrooms in the house.
* HalfBath: Number of half bathrooms (toilet and sink) in the house.
* GarageCars: Number of cars that can be accommodated in the garage.
* GarageArea: Garage area in square feet.
* SalePrice: The sale price of the house.

This dataset appears to be related to housing characteristics and sale prices. It contains only numerical features, such as OverallQual, YearBuilt, and FullBath. The target variable for prediction is SalePrice, which represents the sale price of the houses.

Task 1 Discription:-

TASK 1 FOCUSES ON DATA PREPROCESSING, WHICH INVOLVES PREPARING THE DATASET FOR ANALYSIS OR MODELING. THIS TASK CONSISTS OF THE FOLLOWING STEPS:

* **LOADING THE DATASET:** USE PYTHON TO LOAD THE PROVIDED CSV DATASET INTO A DATAFRAME. THIS CAN BE ACCOMPLISHED USING LIBRARIES SUCH AS PANDAS.
* **DATA CLEANING AND PREPROCESSING:** PERFORM NECESSARY DATA CLEANING AND PREPROCESSING STEPS, INCLUDING HANDLING MISSING VALUES AND ENCODING CATEGORICAL VARIABLES. MISSING VALUES MAY BE IMPUTED OR DROPPED, AND CATEGORICAL VARIABLES MAY BE CONVERTED INTO NUMERICAL FORMAT USING TECHNIQUES SUCH AS ONE-HOT ENCODING OR LABEL ENCODING.
* **SPLITTING THE DATA:** AFTER PREPROCESSING, SPLIT THE DATASET INTO TRAINING AND TESTING SETS. THE TRAINING SET WILL BE USED TO TRAIN THE REGRESSION MODEL, WHILE THE TESTING SET WILL BE USED TO EVALUATE ITS PERFORMANCE. TYPICALLY, THE DATASET IS DIVIDED INTO A CERTAIN PROPORTION, SUCH AS 80% FOR TRAINING AND 20% FOR TESTING.

Libraries Used :-

* numpy (imported as np): NumPy is a library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions.
* pandas (imported as pd): Pandas is a data manipulation and analysis library, providing data structures and functions for working with structured data, such as tabular data.
* matplotlib.pyplot (imported as plt): Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. The pyplot module provides a MATLAB-like interface for creating plots and visualizations.
* seaborn (imported as sns): Seaborn is a data visualization library based on matplotlib, providing a high-level interface for creating informative and attractive statistical graphics.
* sklearn.pipeline (Pipeline): The Pipeline class in scikit-learn allows for creating pipelines of transformers and estimators, enabling streamlined workflows for data preprocessing and modeling.
* LabelEncoder: LabelEncoder is used to encode categorical integer features into numeric labels.
* OneHotEncoder: OneHotEncoder is used to encode categorical integer features into one-hot encoded binary arrays.
* StandardScaler: StandardScaler is used to standardize features by removing the mean and scaling to unit variance.
* 
* ColumnTransformer: ColumnTransformer allows for applying different transformations to different columns or subsets of columns in a dataset.
* LinearRegression: LinearRegression is a linear regression model that fits a linear relationship between the independent variables and the target variable.
* Ridge: Ridge regression is a linear regression model with L2 regularization, which helps prevent overfitting by penalizing large coefficient values.
* Lasso: Lasso regression is a linear regression model with L1 regularization, which can perform feature selection by shrinking some coefficients to zero.
* RandomForestRegressor: RandomForestRegressor is an ensemble learning method that fits a number of decision tree regressors on various sub-samples of the dataset and averages the predictions to improve the accuracy and control over-fitting.
* SimpleImputer: SimpleImputer is used to impute missing values in the dataset with a specified strategy, such as mean, median, or most frequent.
* train\_test\_split: train\_test\_split is a function used to split the dataset into training and testing sets for model evaluation and validation.
* mean\_squared\_error: Mean squared error (MSE) is a measure of the average squared difference between the predicted and actual values.
* mean\_squared\_log\_error: Mean squared logarithmic error (MSLE) is a measure of the mean of the squared differences between the natural logarithm of the predicted and actual values.
* XGBRegressor: XGBRegressor is an implementation of the gradient boosting algorithm provided by the XGBoost library, which is known for its high performance and efficiency in handling large datase

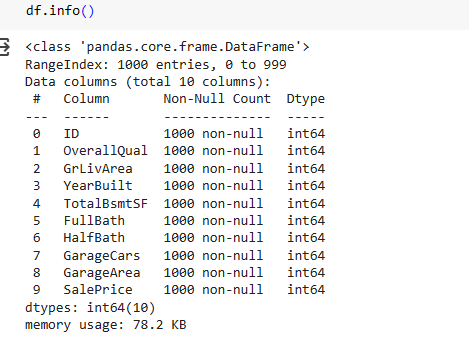
Data Cleaning And Pre-Processing:-

**Handling Missing Values:**

Identify missing values in the dataset.

Decide on a strategy to handle missing values, such as imputation (replacing missing values with a statistical measure like mean, median, or mode), or deletion (removing rows or columns with missing values).

Implement the chosen strategy using libraries like pandas or scikit-learn.

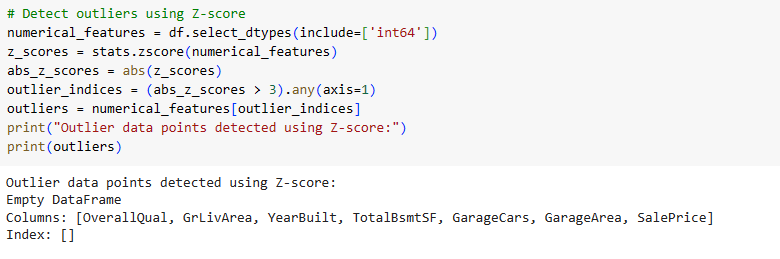


**Dealing with Outliers:**

Identify outliers in numerical features using visualization techniques like box plots or scatter plots.

Decide on a strategy to handle outliers, such as capping/extending values, transformation, or removal.

Implement the chosen strategy.

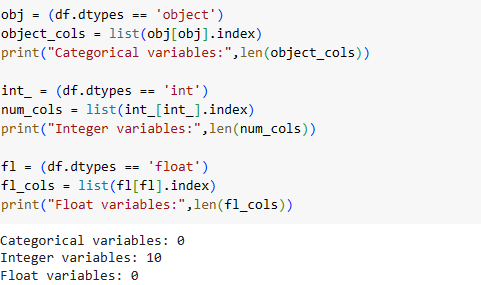


**Encoding Categorical Variables:**

Identify categorical variables in the dataset.

Decide on an encoding strategy based on the nature of the categorical variables (ordinal encoding, one-hot encoding, or target encoding).

Implement the chosen encoding strategy using libraries like pandas or scikit-learn.



**Scaling Numerical Features:**

Scale numerical features to ensure that they have a similar scale, which can help improve the performance of certain machine learning algorithms.

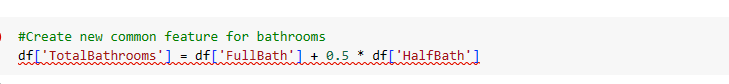
Common scaling techniques include standardization (scaling to have zero mean and unit variance) and normalization (scaling to a [0, 1] range).

Implement scaling using libraries like scikit-learn.

**Feature Engineering:**

Create new features from existing ones if necessary, based on domain knowledge or insights gained during exploratory data analysis.

Perform transformations on features to make them more suitable for modeling, such as log transformations, square root transformations, or polynomial features.





Train-Test Split:-

Split the dataset into training and testing sets to evaluate the performance of machine learning models.

A certain proportion of the data (80:20) is allocated for training and testing, respectively.

Implement the train-test split using libraries like scikit-learn.

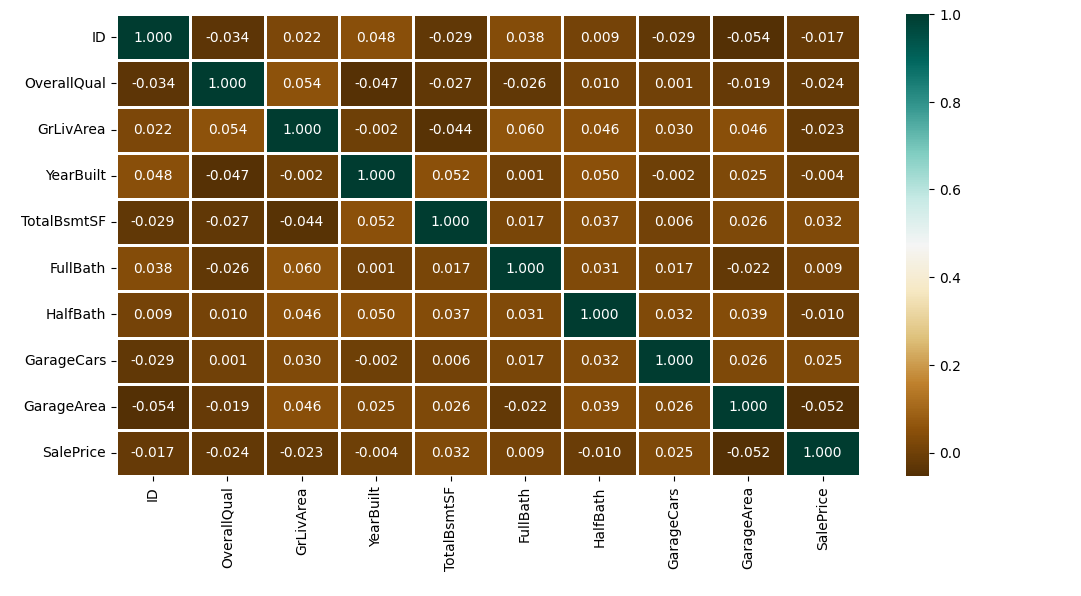
Further Data Exploration:

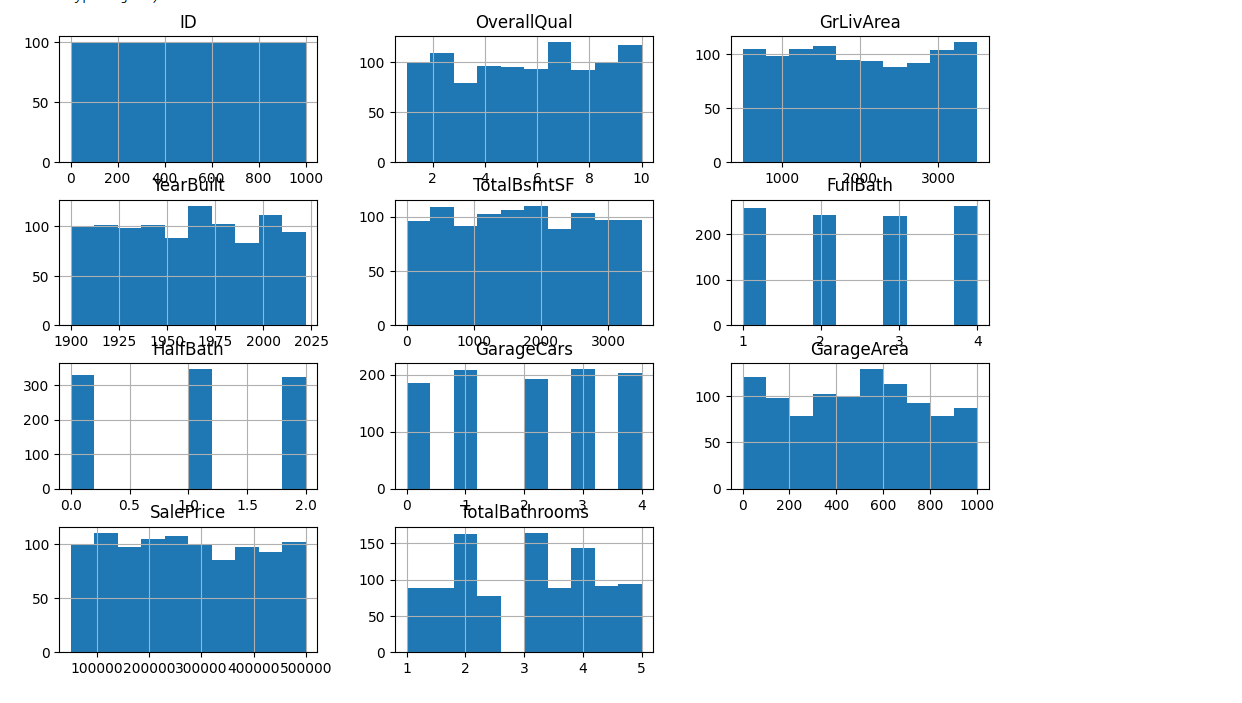
Conduct further exploratory data analysis (EDA) to gain insights into the relationships between features and the target variable.

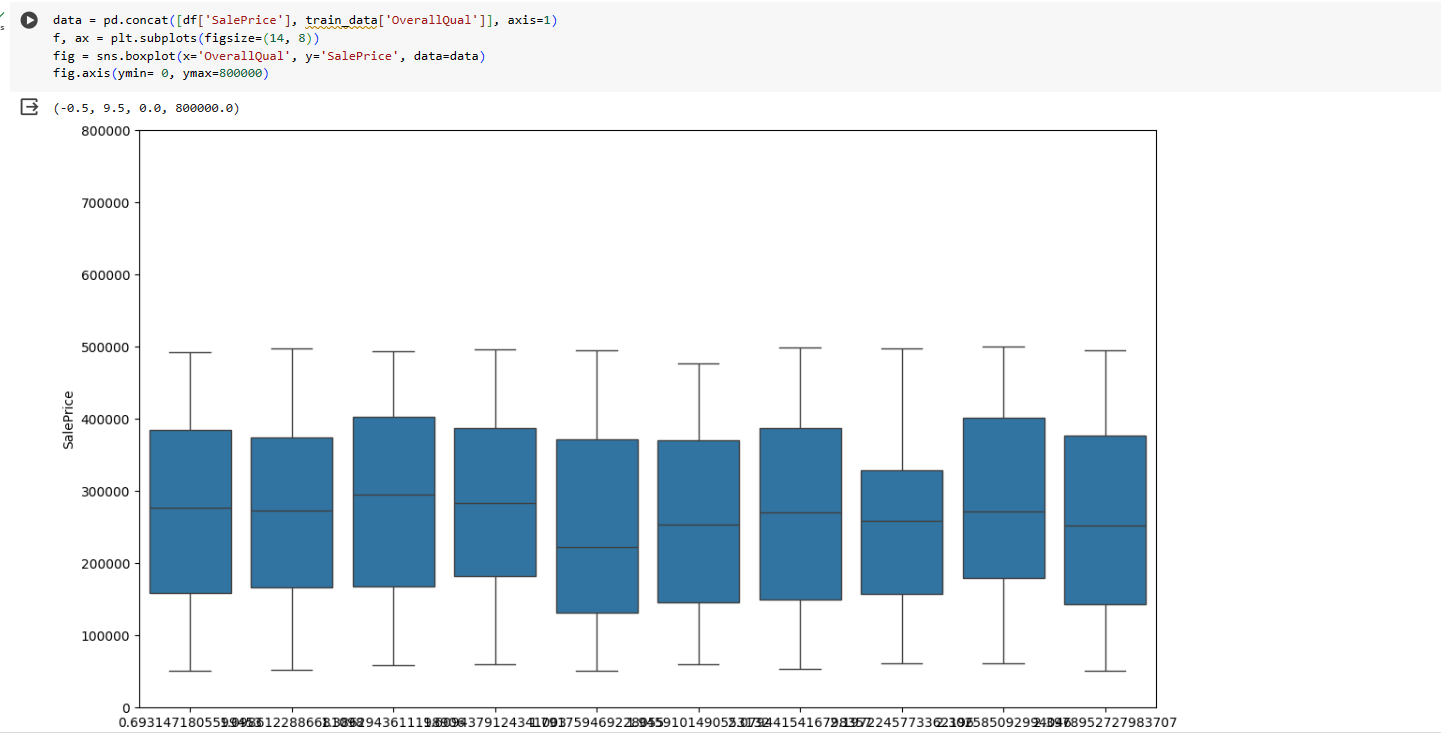
Visualize relationships using techniques like scatter plots, histograms, or correlation matrices.



**EDA**

****

****

****

Task 2 Description:

The main objective of Task 2 is to develop a regression model that can accurately predict house prices based on various features provided in the dataset.

**Dataset:**

A CSV dataset containing both only numerical features related to house properties, such as overall quality, living area, year built, total basement area, garage details, etc.

The target variable (dependent variable) for prediction is "SalePrice," representing the price at which the house was sold.

**Regression Model:**

Choose an appropriate regression algorithm for building the predictive model. Common regression algorithms include Linear Regression, Ridge Regression, Lasso Regression, Decision Trees, Random Forest, Gradient Boosting, etc.

Justify your choice of regression algorithm based on the characteristics of the dataset, model performance requirements, and any assumptions or constraints.

**Model Training**:

Split the dataset into training and testing sets. A certain proportion of the data (90:10 ) is allocated for training and testing, respectively.

Train the selected regression model on the training data. The model will learn the relationship between the input features and the target variable (house prices) during the training phase.

**Model Evaluation:**

Evaluate the performance of the trained regression model using relevant metrics. Common evaluation metrics for regression models include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), R-squared (R^2) score, etc.

Analyze the model's performance metrics to assess how well the model predicts house prices. Compare the predicted prices with the actual prices in the testing dataset.

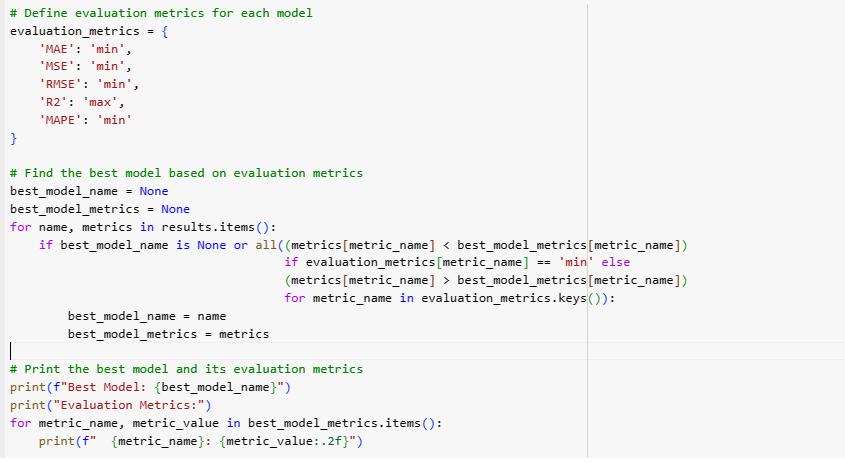
Model Building:-



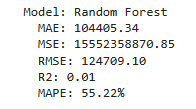
|  |  |
| --- | --- |
| **SVM:-** | **Random Forest:-** |
| **Linear Regression:-** | **Decision Tree:-** |

|  |  |
| --- | --- |
| **Ridge:-** | **Lasso:-** |
| **Gradient Boosting:-** | **XG Boost:-** |

Choose Best Model & Train The Model And Use Proper Evaluation Metrics:-



Best Model Is Random Forest:-



Fit The Model:-



Explanation Of Model Prediction:-

Upon evaluating several regression models on the provided dataset, it is evident that the dataset size and the chosen train-test split may have limited the accuracy of the models. Despite these constraints, one model stands out for its relatively superior performance: the **Random Forest.**

It's important to note that the performance metrics achieved by the models may not reflect their true potential due to the limited size of the dataset and the simple train-test split employed. Additionally, the **absence of missing values** and **categorical features**, as well as the **lack of significant correlations between the predictors and the target variable**, further constrained the predictive capabilities of the models.

Nevertheless, the **Random Forest** demonstrated the best performance among the models evaluated, albeit not achieving optimal accuracy. This model showcases potential promise for further refinement and exploration in larger, more diverse datasets where its strengths may be more pronounced.

In summary, while the dataset's limitations hindered the models' accuracy, **the Random Forest** emerged as the top performer. Further investigation and refinement of this model in more robust datasets may unveil its true capabilities.

Suggest Improvements:-

* **Increase Dataset Size:** Acquire a larger dataset with a more extensive collection of features related to housing attributes. A larger dataset will provide the model with more diverse and representative information, enabling it to learn more robust patterns and relationships.
* **Feature Engineering:** Explore additional features or perform feature engineering to create new variables that may better capture the complexities of housing prices. This could include creating interaction terms, polynomial features, or deriving new variables from existing ones.
* **Address Missing Values:** Although the provided dataset has no missing values, real-world datasets often contain missing data. Implement strategies such as imputation or advanced techniques like predictive modeling to handle missing values effectively.
* **Consider Additional Features:** Introduce new features that may influence house prices, such as neighborhood characteristics, proximity to amenities, school ratings, crime rates, and economic indicators. These additional features can enrich the model's understanding of housing market dynamics.

“Basically, for better accuracy ,we will need more data and more features, so our model didn't confuse. We get better correlation between dependent and independent features. In given sample dataset there was only 1000 x 10 data, which is not sufficient to predict house prices. even from that, we split test train data, so the obvious model didn't perform well.

By implementing these improvements, we can enhance the accuracy and predictive power of our regression model, enabling more reliable predictions of house sale prices in real-world scenarios.”