

**Predicting House Prices using Linear Regression,
Random Forest and XGBoost**



**Cluster Innovation Centre
University of Delhi**

Submitted by
Anhad Mehrotra
Roll no.- 152209
Email ID- anhad01mehrotra@gmail.com

April 2025

For the subject
Artificial Intelligence

ABSTRACT

The project is centered around forecasting house prices using three machine learning algorithms: Linear Regression, Random Forest, and XGBoost. With a dataset acquired from Kaggle, the project uses ample training data as well as test data to model and predict trends of house pricing according to several features.

The primary goal of the project is to investigate, apply, and compare various regression algorithms for efficient forecasting of houses' prices. By carrying out a comparative analysis of the effectiveness of various regression algorithms, the project aims to determine the most efficient model that can be applied to real-life applications where exact price forecasting is imperative for buyers, sellers, as well as real estate experts.

During the project, extensive data preprocessing was performed, which involved missing values handling, outlier removal, encoding of categorical features into numeric types, and data scaling to provide consistency. Upon preprocessing, the dataset was split into training data as well as testing data. The three chosen models were subsequently trained against the testing data. Their performance was quantified through statistical measures like Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the R-squared score (R^2). Additional fine-tuning was performed with hyperparameter adjustment through GridSearchCV, accompanied by a reassessment of model performance.

The importance of this project is that it adopts a holistic view of model performance and behavior within a real estate application like house price estimation. By computing a range of algorithms using a form of loose parameter search, the project identifies the best model while helping to create more robust, efficient, and applicable machine learning solutions for the real estate market as well as other similar domains.

1. INTRODUCTION

Property price forecasting has been a pivotal problem for the real estate sector because there are a number of variables that affect prices. Buyers and sellers need reliable estimates to proceed with decisions, while mistakes have huge financial consequences. Conventional approaches to assessing real estate prices fail to handle the complexity of considerations as well as the number of variables involved, providing a rationale for data-driven, machine learning-based approaches.

The importance of solving the problem is the direct effect it can have on the housing market, as well as associated industries. Each of these groups can be aided by a sound forecasting model. It helps individual buyers and sellers as much as it can real estate developers, financial institutions, and government policymakers. It aids market analysis, investment decision making, and risk reduction. It is of great importance, therefore, to construct a sound and effective model for forecasting house prices.

The goal of this project is to build and compare machine learning models that are effective at predicting home prices with high accuracy. It uses Linear Regression, Random Forest, and XGBoost algorithms as specific methods to compare their respective performance. By thorough evaluation as well as optimization, the project seeks to determine the optimal performing model that can be suggested for real-world applications.

The data for the project was gathered from Kaggle with a total of 3280 rows and 80 columns of different features of the properties. Thorough preprocessing was applied to the dataset, where missing data was handled, outliers were removed, categorical variables were encoded into numeric forms, and the data was scaled. All these were aimed at making the dataset clean for efficient training of the regression models.

2. LITERATURE REVIEW

Predicting house prices has been a long-standing challenge in the field of real estate analytics. Traditionally, statistical methods like linear regression have been widely employed for modeling housing prices due to their simplicity and interpretability. However, the growing complexity of real estate data, influenced by numerous factors such as location, amenities, environmental conditions, and market trends, has necessitated the adoption of more advanced machine learning techniques.

Several studies have demonstrated the effectiveness of ensemble methods like Random Forest and Gradient Boosting techniques (such as XGBoost) in handling high-dimensional and structured datasets. These models are capable of capturing complex non-linear relationships between features and the target variable, making them suitable for real-world predictive tasks like house price estimation. Research suggests that ensemble models generally outperform traditional regression models by reducing variance and bias, leading to more accurate and stable predictions.

The availability of extensive datasets, such as the "House Prices - Advanced Regression Techniques" competition on Kaggle, has provided a standard benchmark for evaluating the performance of different algorithms on real-world data. Studies based on this dataset have shown that preprocessing steps — including missing value handling, feature encoding, outlier removal, and feature scaling — significantly impact model performance. Furthermore, the application of hyperparameter tuning techniques like GridSearchCV has been proven critical for optimizing model accuracy.

Given these findings from the literature, this project focuses on implementing and comparing Linear Regression, Random Forest, and XGBoost models. The goal is to validate whether ensemble models, when properly tuned and preprocessed, can significantly outperform simple linear models in the context of house price prediction.

3. DATASET DESCRIPTION

The dataset used in this project was obtained from Kaggle's "House Prices – Advanced Regression Techniques" competition. It comprises two separate CSV files: a training set and a testing set. Both files contain 1640 rows and 80 columns, with the training set including an additional column for the SalePrice, which represents the target variable that needs to be predicted.

[4]: df.head()

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2008	WD	Normal	208500	
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5	2007	WD	Normal	181500	
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9	2008	WD	Normal	223500	
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2006	WD	Abnorml	140000	
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12	2008	WD	Normal	250000	

rows × 81 columns

+ Code

+ Markdown

[5]: df.shape

[5]: (2919, 81)

Each row in the dataset corresponds to a house located within Ames, Iowa, and the columns capture a wide range of property characteristics. These features include both numeric and categorical data types. Examples of features are lot size (LotArea), year built (YearBuilt), number of bedrooms (BedroomAbvGr), quality of materials (OverallQual), and proximity to amenities or roadways (Condition1, Condition2). Categorical features such as zoning classification (MSZoning), type of dwelling (BldgType), and style of house (HouseStyle) provide further contextual information about each property.

The dataset offers a rich mix of variables, ranging from physical property attributes to environmental conditions and neighborhood information. For instance, features like Neighborhood, Alley, LandSlope, RoofStyle, and Exterior1st capture the external and environmental factors affecting house prices. Quality and condition ratings are given for different parts of the house, such as basements, garages, kitchens, and heating systems. Additionally, several variables pertain to the sale itself, including SaleType and SaleCondition.

```
[14]: df.describe()
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDeckSF	OpenPorchSF	EnclosedPorch	3Ssnf
count	2919.000000	2919.000000	2433.000000	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000	2896.000000	2918.000000	...	2919.000000	2919.000000	2919.000000	2919.000000
mean	1460.000000	57.137718	69.305795	10168.114080	6.089072	5.564577	1971.312778	1984.264474	102.201312	441.423235	...	93.709832	47.486811	23.098321	2.600000
std	842.787043	42.517628	23.344905	7886.996359	1.409947	1.113131	30.291442	20.894344	179.334253	455.610826	...	126.526589	67.575493	64.244246	25.160000
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000
25%	730.500000	20.000000	59.000000	7478.000000	5.000000	5.000000	1953.500000	1965.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000
50%	1460.000000	50.000000	68.000000	9453.000000	6.000000	5.000000	1973.000000	1993.000000	0.000000	368.500000	...	0.000000	26.000000	0.000000	0.000000
75%	2189.500000	70.000000	80.000000	11570.000000	7.000000	6.000000	2001.000000	2004.000000	164.000000	733.000000	...	168.000000	70.000000	0.000000	0.000000
max	2919.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...	1424.000000	742.000000	1012.000000	508.000000

8 rows × 38 columns

Prior to applying any machine learning models, comprehensive preprocessing was performed on the dataset. This included handling missing values appropriately, removing outliers to ensure data integrity, encoding categorical variables into numerical formats to make them suitable for model ingestion, and scaling the numerical features to bring them onto a comparable scale. These preprocessing steps helped in preparing a clean, structured, and model-ready dataset that significantly improved the performance and reliability of the regression models used in the project.

4. METHODOLOGY

2.1) Algorithm Selection

For the purposes of this project, three machine learning algorithms were chosen: **Linear Regression**, **Random Forest**, and **XGBoost**. They were selected because they have been shown to work effectively with regression issues, especially where there are structured data with a high number of features, as with the house price data.

Linear Regression was chosen as a baseline model. It is a basic regression method that defines a linear relationship between independent variables and the dependent variable. While it makes a linear assumption and does not necessarily identify sophisticated patterns within the data, it serves as a valid benchmark against which to compare the performance gains of more sophisticated models.

Random Forest was selected because it is a method of ensemble learning that builds multiple decision trees and combines their predictions to refine accuracy and reduce overfitting. It is strong against non-linear relationships and is outlier- and noise-resistant, making it ideal for complex datasets such as this one.

XGBoost was chosen because of its exceptional performance in various machine learning competitions as well as practical applications. XGBoost is a gradient boosting algorithm that is optimized to build successive models, each of which tries to fix the mistakes of its predecessor. XGBoost is efficient, fast, as well as being very accurate. It is a perfect fit for structured data tasks like predicting the price of houses.

2.2) Tools Used

The project was implemented through the utilization of **Python**, within the framework of Kaggle using **Jupyter Notebook**. Various core libraries were employed to conduct data processing, visualization, model construction, as well as model evaluation.

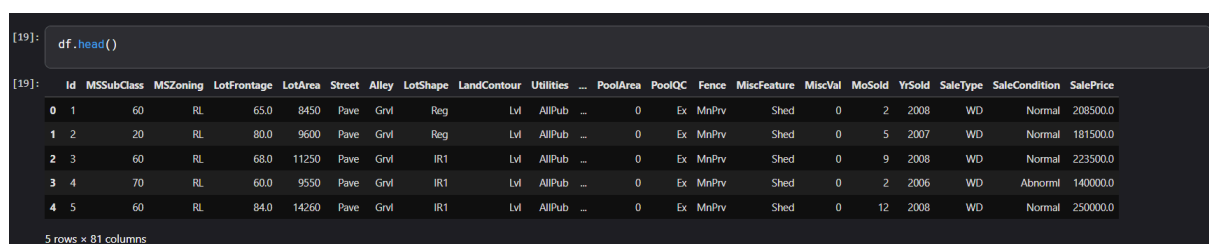
Pandas and **NumPy** were utilized for data manipulation as well as mathematical operations for efficient handling of huge datasets. **Matplotlib** and **Seaborn** were used for data visualization for creating meaningful graphs and plots to analyze data distributions as well as relationships.

Scikit-learn was the main machine learning library, performing operations of data splitting with `train_test_split`, feature scaling using `StandardScaler`, as well as model evaluation using several performance measures such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), as well as R^2 score. `PolynomialFeatures` from `sklearn.preprocessing` was employed to investigate polynomial regression.

For model training, I have implemented **LinearRegression**, **RandomForestRegressor**, and **XGBClassifier** from the XGBoost library. For model performance optimization through hyperparameter adjustment, **GridSearchCV** was utilized alongside `RandomizedSearchCV`. `Accuracy_score` and `classification_report` were included as evaluation measures as well, even though the main emphasis was placed on regression performance measures.

2.3) Preprocessing

The preprocessing step had a number of important steps to clean the dataset so it was ready for training machine learning models. First, missing values within the dataset were addressed systematically. For the numerical columns, missing values were imputed with the median of the corresponding column to avoid biasing the imputation with the presence of outliers. For the categorical columns, the missing values were imputed with the most frequent category (mode) to keep the data's distribution and meaning intact.

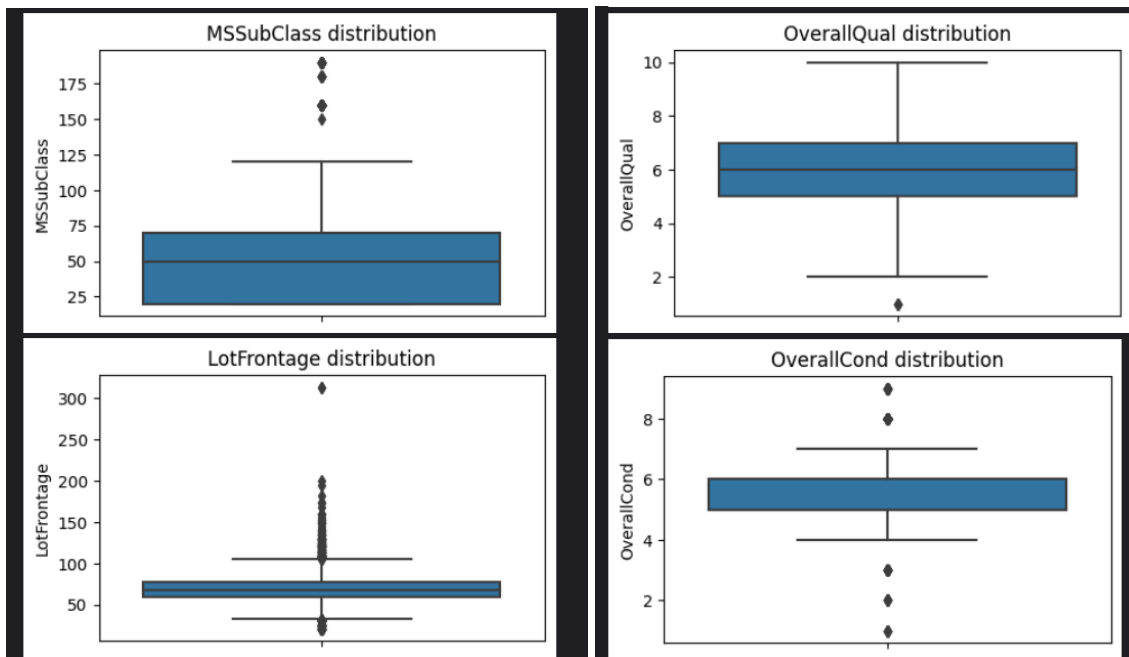


```
[19]: df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
0	1	60	RL	65.0	8450	Pave	Grvl	Reg	Lvl	AllPub	0	Ex	MnPrv	Shed	0	2	2008	WD	Normal	208500.0
1	2	20	RL	80.0	9600	Pave	Grvl	Reg	Lvl	AllPub	0	Ex	MnPrv	Shed	0	5	2007	WD	Normal	181500.0
2	3	60	RL	68.0	11250	Pave	Grvl	IR1	Lvl	AllPub	0	Ex	MnPrv	Shed	0	9	2008	WD	Normal	223500.0
3	4	70	RL	60.0	9550	Pave	Grvl	IR1	Lvl	AllPub	0	Ex	MnPrv	Shed	0	2	2006	WD	Abnorml	140000.0
4	5	60	RL	84.0	14260	Pave	Grvl	IR1	Lvl	AllPub	0	Ex	MnPrv	Shed	0	12	2008	WD	Normal	250000.0

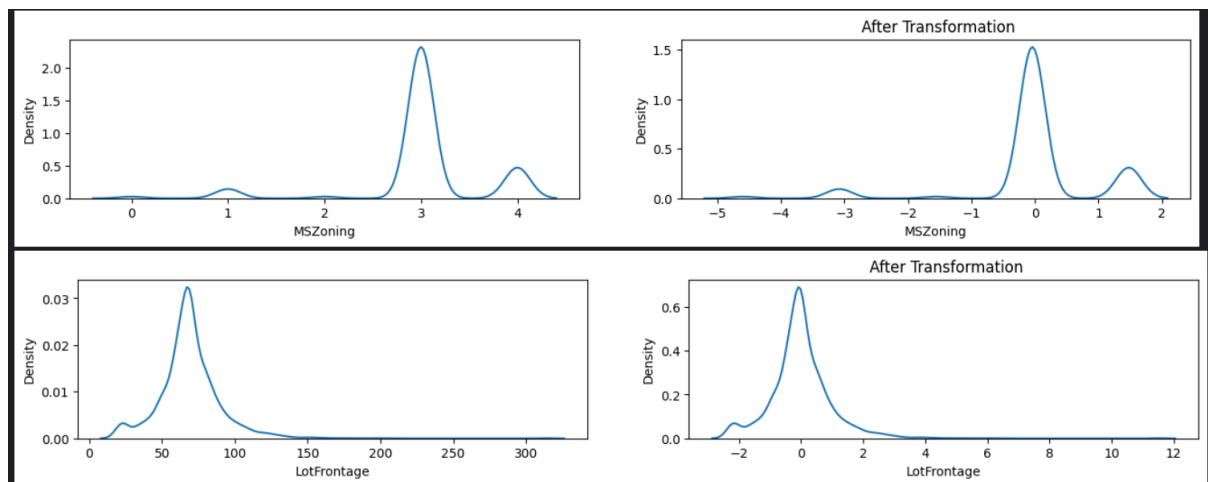
5 rows × 21 columns

Then, outliers were removed. Boxplots were employed as a visual means of identifying anomalies in the numeric attributes. Outliers, potentially causing model training and predictions to be skewed, were subsequently removed for the sake of data integrity as well as to enhance model performance.



Upon outlier removal, categorical attributes were encoded as numeric. Label Encoding was utilized, where a distinct integer was given to each distinct category within a feature. This was a necessity, as a vast majority of machine learning algorithms need data to be numeric.

Lastly, feature scaling was conducted to normalize the range of independent variables. While using `StandardScaler`, the features were scaled so that their mean was zero and standard deviation was one. This scaling of data ensured equal contribution of all features towards model training as well as facilitated quicker convergence of gradient-based algorithms.



3. IMPLEMENTATION DETAILS

The implementation of the models followed a structured sequence of steps to ensure accuracy, robustness, and reproducibility.

1. Data Splitting

- The dataset was divided into **training** and **testing** subsets using an **80-20 split**.
- A **random state of 2** was set to ensure that the split was **reproducible**.
- This resulted in:
 - **2335 samples** for training
 - **584 samples** for testing
- The shapes and structure of the original dataset were preserved to maintain representativeness.

2. Model Implementation

- **Three different regression models** were selected for implementation:
 - **Linear Regression**: Used as a **baseline model** to establish a basic comparison.
 - **Random Forest Regressor**: Chosen for its ability to **handle non-linear relationships** and **resist overfitting**.
 - **XGBoost Regressor**: Selected for its **high performance**, **speed**, and **robustness** in structured data problems.
- Each model was:
 - Trained on the **training dataset**.
 - Used to make predictions on both **training** and **testing datasets**.

3. Model Evaluation (Before Tuning)

- The models were initially evaluated using standard regression metrics:
 - **Mean Absolute Error (MAE)**
 - **Mean Squared Error (MSE)**

- **Root Mean Squared Error (RMSE)**
- **R-squared (R^2) Score**
- These metrics provided a **comprehensive overview** of each model's predictive accuracy and error distribution.

4. Hyperparameter Tuning

- **GridSearchCV** with **5-fold cross-validation** was employed for hyperparameter optimization.
- Specific hyperparameter grids were defined for each model:
 - **Linear Regression**: Focused on intercept fitting and parallel processing settings.
 - **Random Forest**: Included variations in the number of trees, depth of trees, minimum samples for splits, and feature selection methods.
 - **XGBoost**: Tuned on parameters like learning rate, depth, number of estimators, column subsampling, gamma, and regularization terms.
- After tuning:
 - Each model was **re-trained** with the **best hyperparameters**.
 - Predictions were made again on the **testing dataset** for comparison.

5. Final Evaluation (After Tuning)

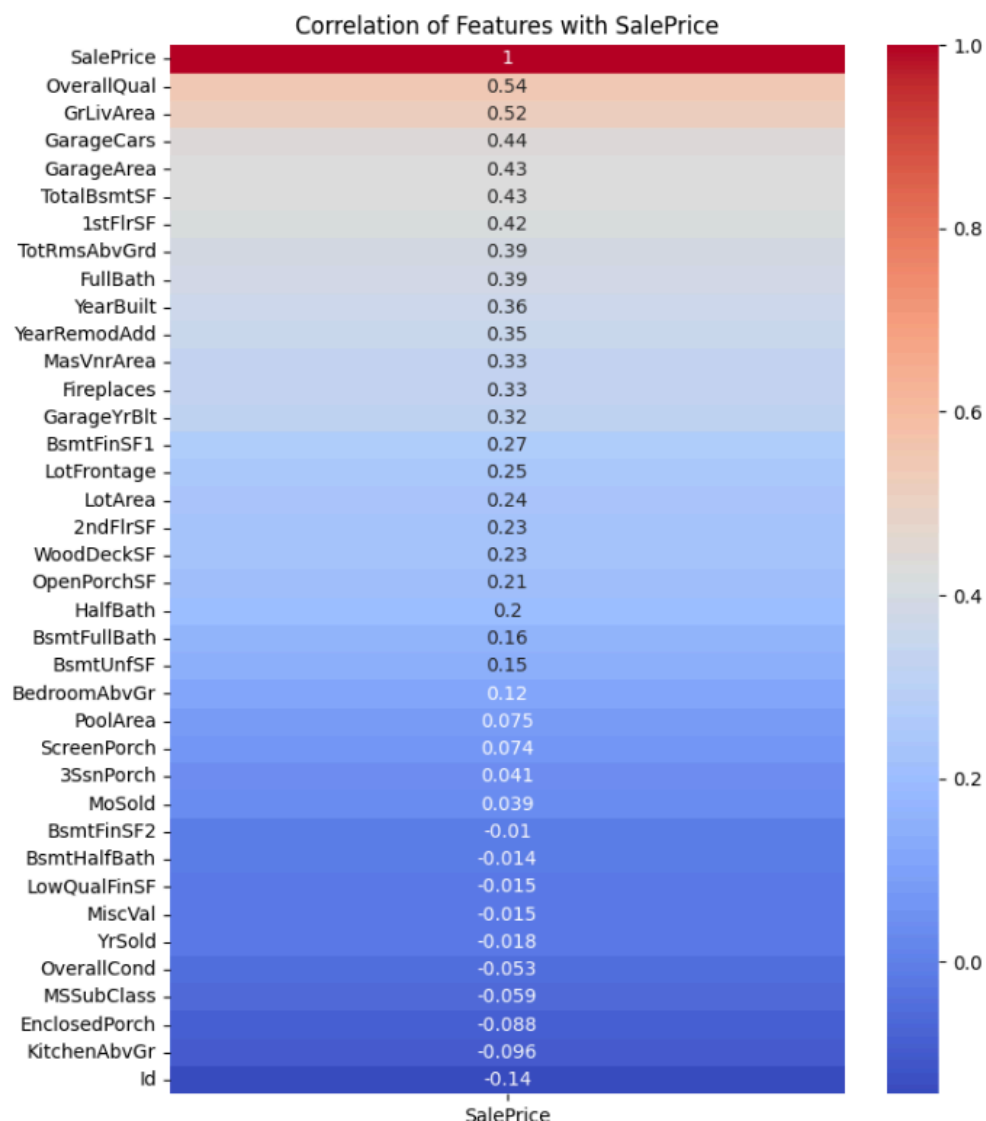
- The re-trained models were evaluated using the **same performance metrics** (MAE, MSE, RMSE, R^2).
- The **improvements** observed were then used to decide the **best performing model** for the problem.

5. RESULTS AND COMPARATIVE ANALYSIS

The performance of the three selected regression models — Linear Regression, Random Forest, and XGBoost — was thoroughly evaluated using multiple statistical metrics. The models were first assessed with their default parameters to establish a baseline performance. Following this, hyperparameter tuning was conducted to optimize each model and improve their predictive accuracy. The evaluation focused on key performance indicators including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the R-squared (R^2) score, offering a comprehensive view of each model's effectiveness in predicting house prices.

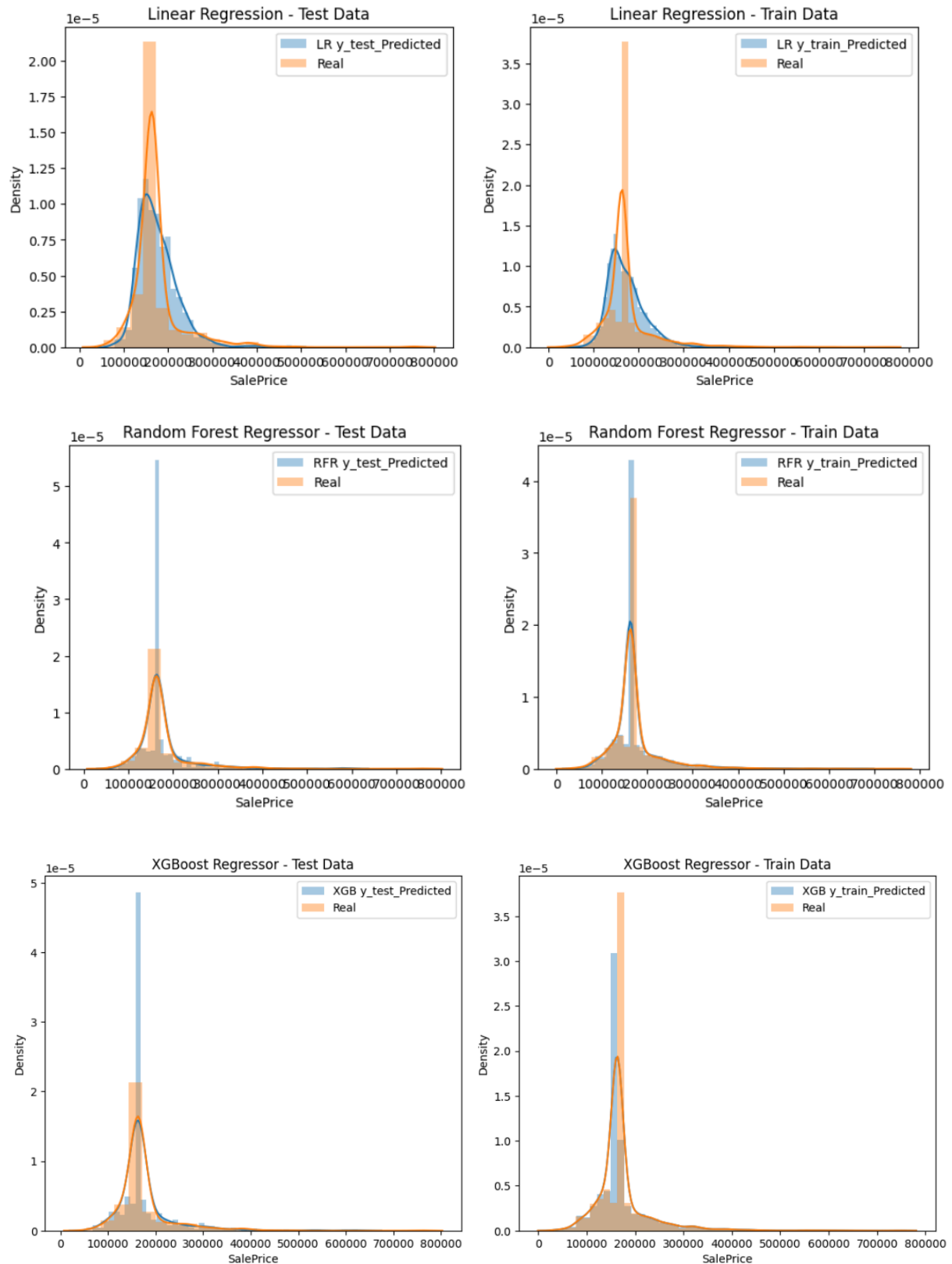
3.1) Correlation Matrix

To understand the relationships between the target variable (SalePrice) and the other features, a correlation matrix was generated, highlighting the features most strongly correlated with house prices.

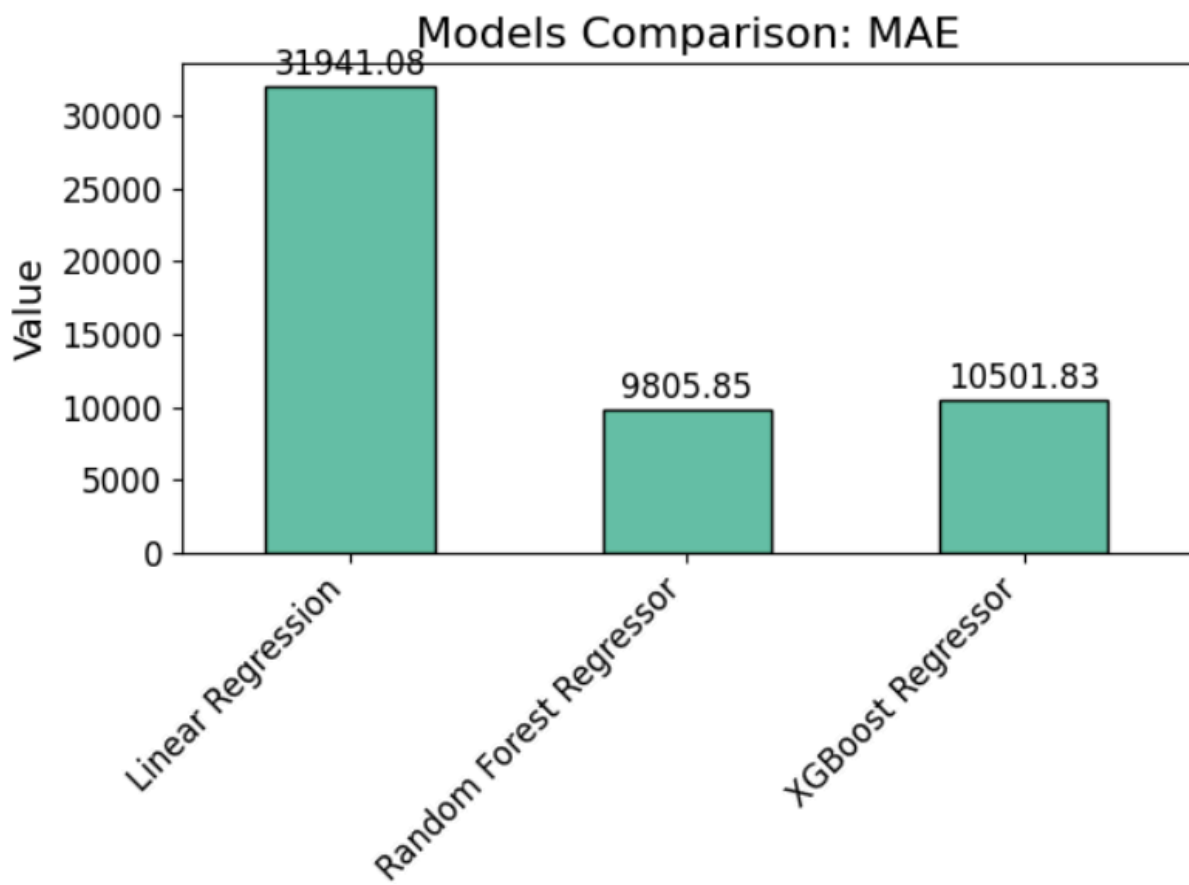


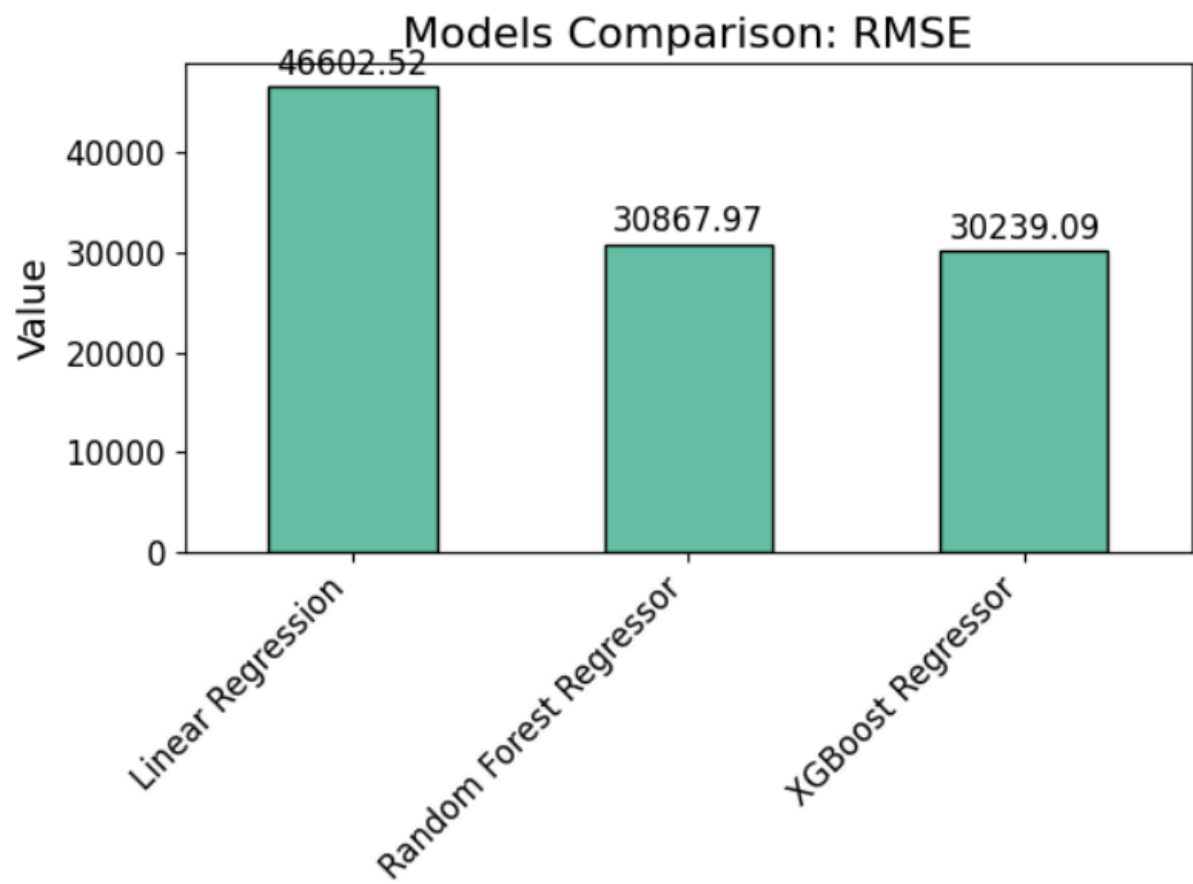
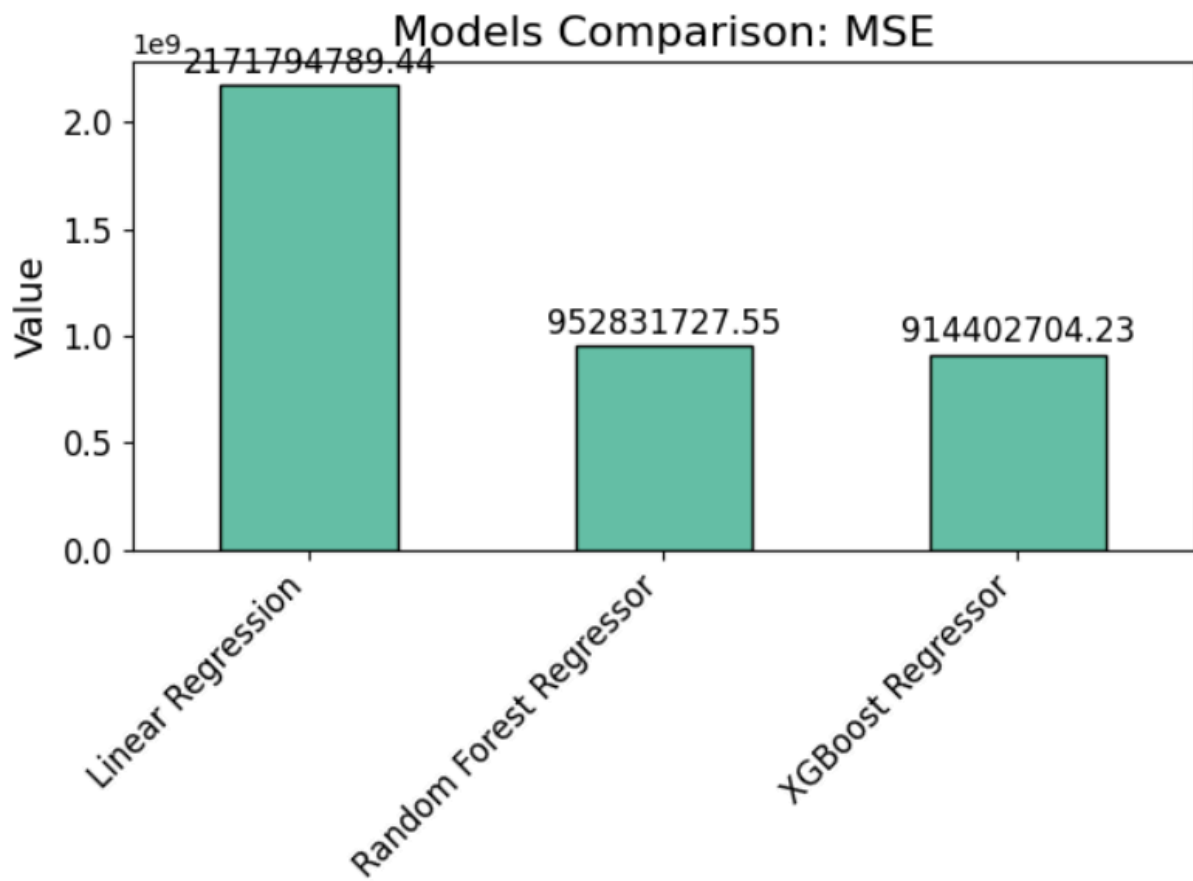
3.2) Results before Hyperparameter Tuning

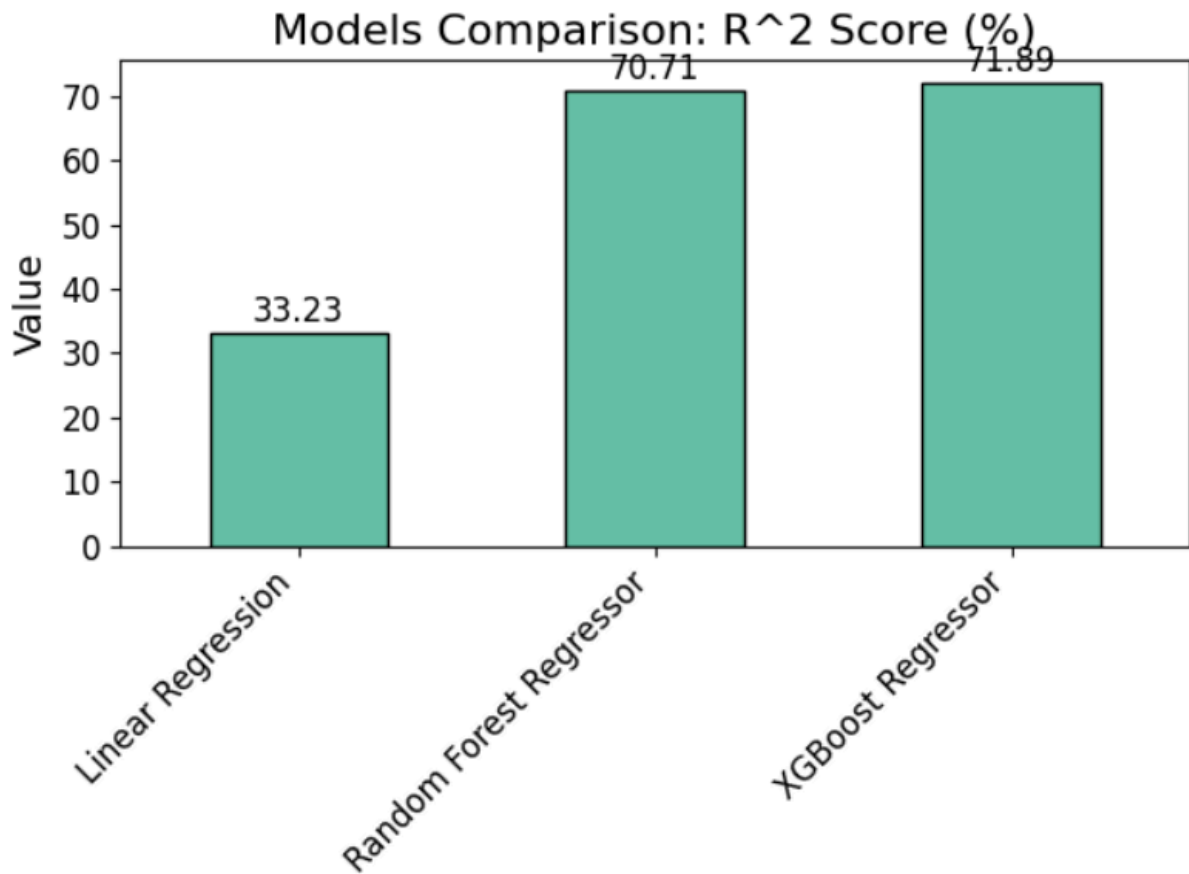
To assess the initial performance of the models, evaluation metrics such as MAE, MSE, RMSE, and R^2 were calculated before applying any hyperparameter optimization.



Model	MAE	MSE	RMSE	R ² Score
Linear Regression	31,941.08	2,171,794,789.44	46,602.52	0.3323
Random Forest	9,917.40	951,096,466.28	30,839.85	0.7076
XGBoost	10,501.83	914,402,704.23	30,239.09	0.7189



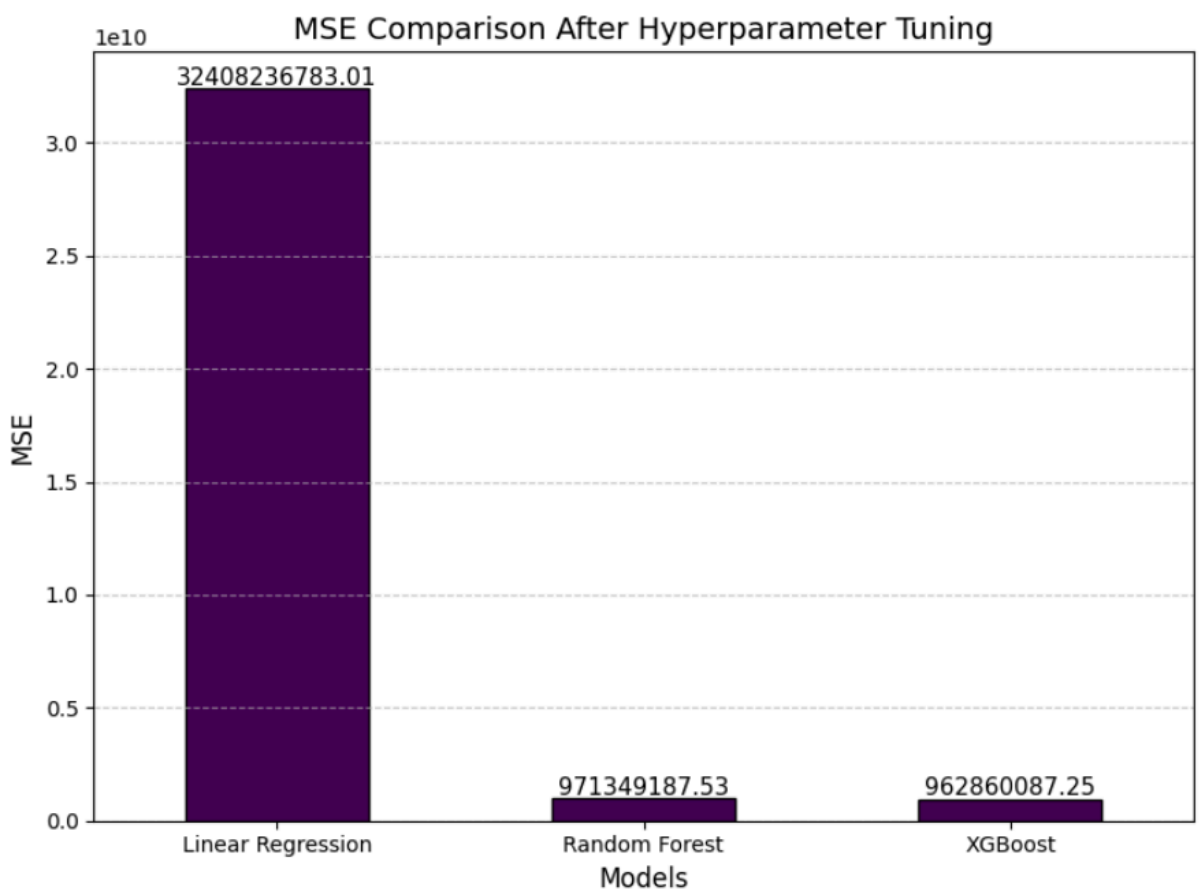
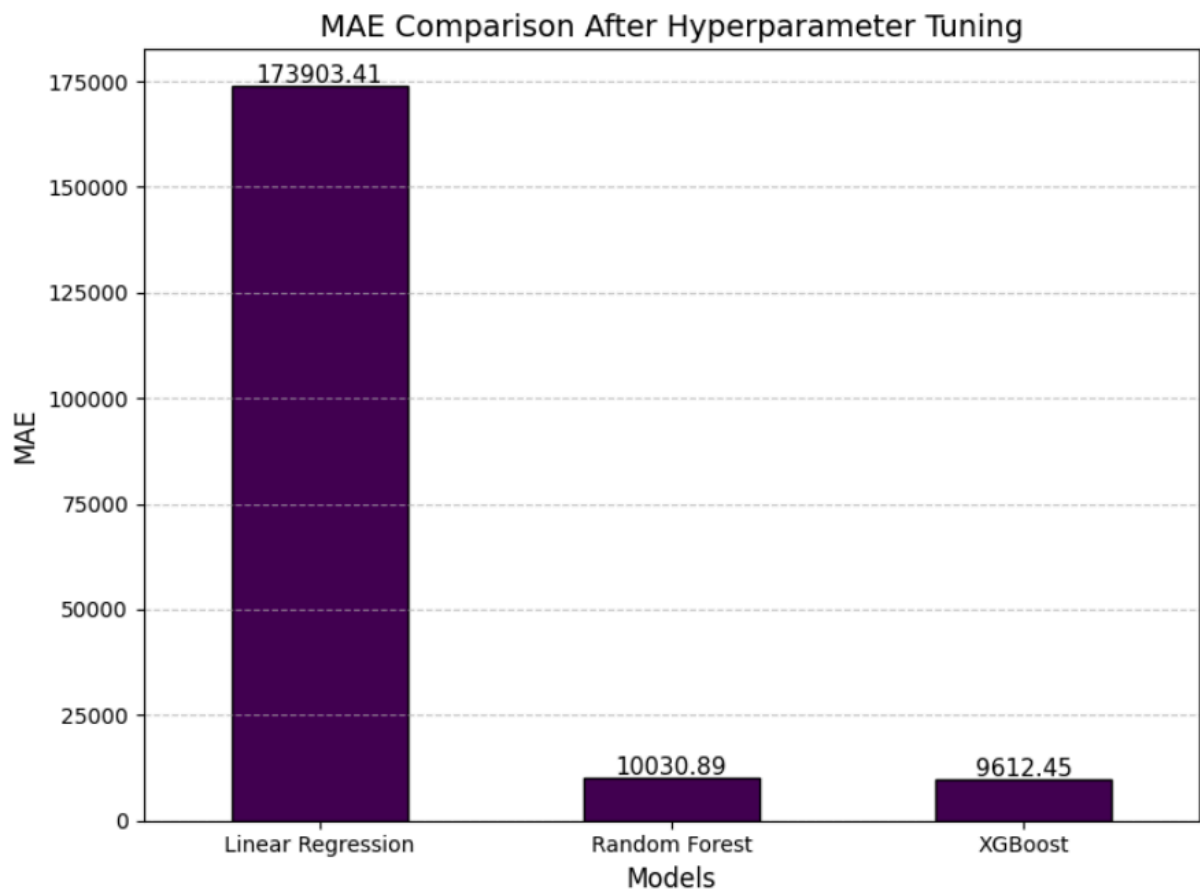


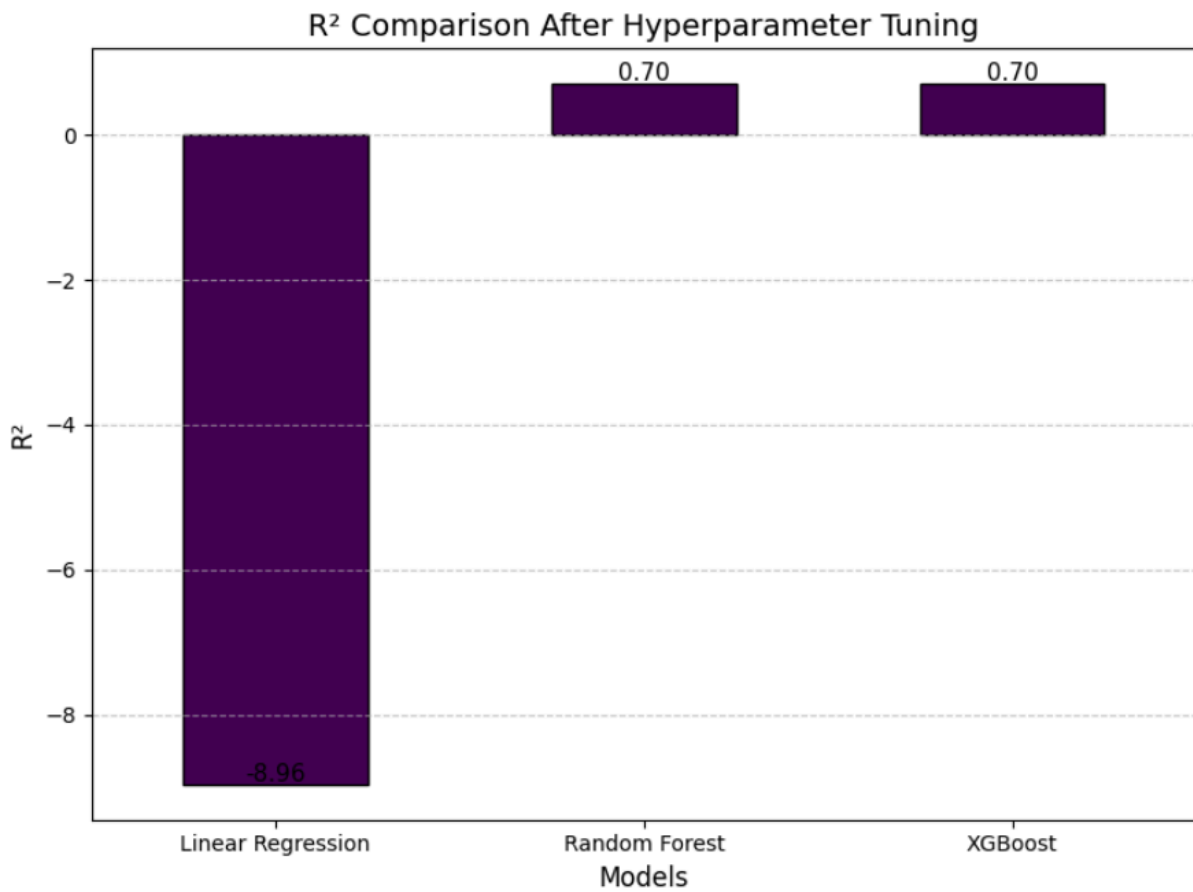
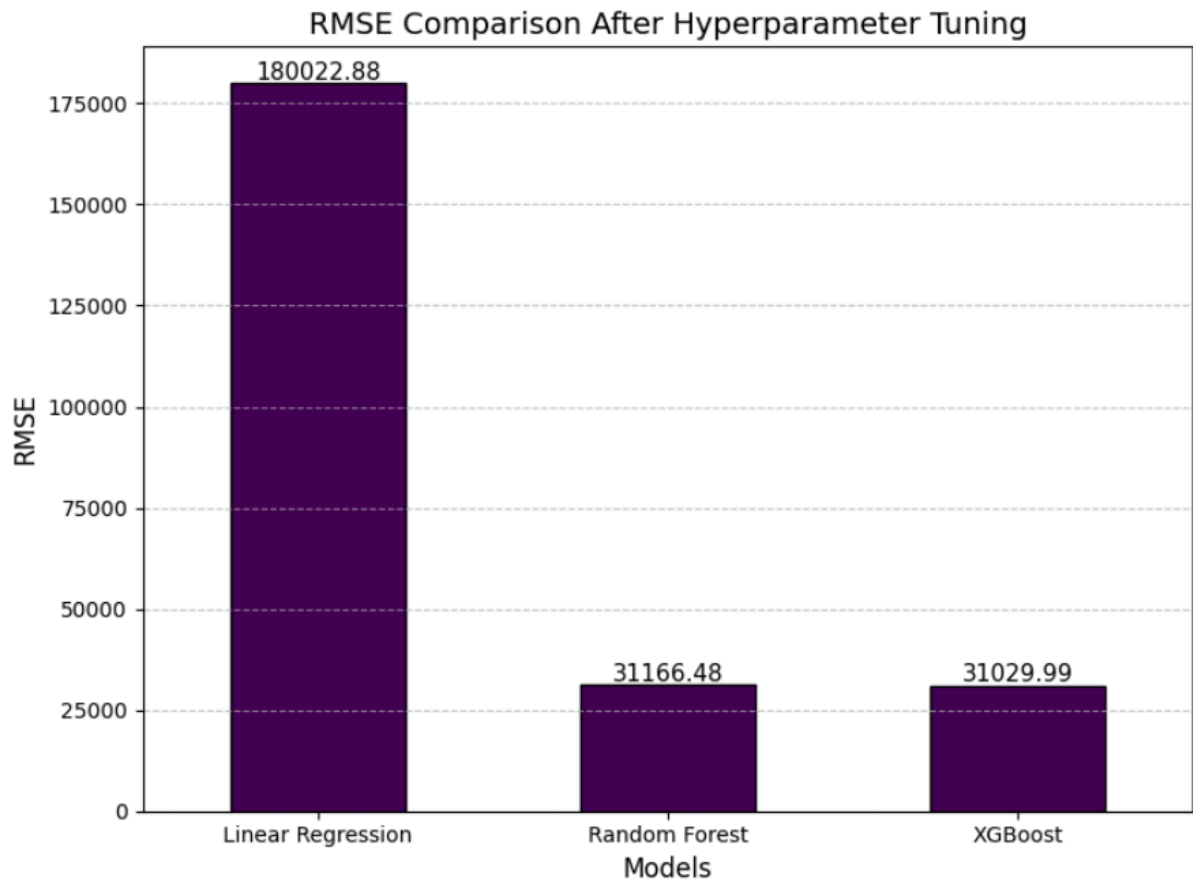


3.3) Results after Hyperparameter Tuning

Following hyperparameter tuning using GridSearchCV, the models were re-evaluated using the same set of performance metrics to measure improvements and identify the most effective model.

Model	MAE	MSE	RMSE	R ² Score
Linear Regression	173,903.41	32,408,236,783.01	180022.88	-8.9632
Random Forest	10,030.89	971,349,187.53	31166.48	0.7014
XGBoost	9,612.45	962,860,087.25	31029.99	0.7040





4. CONCLUSION

Based on the obtained results, the following conclusions were drawn:

1. Baseline Model Performance (Before Hyperparameter Tuning):

- **Linear Regression** showed the **weakest performance** among all models with a high RMSE of 46,602.52 and a low R^2 score of 0.3323.
- **Random Forest** and **XGBoost** both performed significantly better than Linear Regression:
 - Random Forest achieved a R^2 score of **0.7076**.
 - XGBoost slightly outperformed Random Forest with a R^2 score of **0.7189**.
- Overall, **XGBoost was the best model** before tuning, closely followed by Random Forest.

2. Model Performance After Hyperparameter Tuning:

- **Linear Regression** performed extremely poorly after hyperparameter tuning, resulting in a **negative R^2 score of -8.9632** and an even higher RMSE of 180,022.88, indicating that tuning made its performance significantly worse.
- **Random Forest** maintained strong performance with a slight decrease in R^2 score to **0.7014**, but overall it remained stable and reliable.
- **XGBoost** continued to perform the best even after tuning, achieving an R^2 score of **0.7040** and the lowest MAE among all models.

3. Final Model Comparison:

- **XGBoost Regressor emerged as the best performing model overall**, showing consistent and reliable results both before and after hyperparameter tuning.
- **Random Forest Regressor** also performed robustly and could serve as a strong alternative model.
- **Linear Regression** was not suitable for this problem as it failed to capture the complex relationships within the data, both before and after tuning.

4. Key Observations:

- Ensemble methods like **Random Forest** and **XGBoost** are highly effective in structured data problems such as house price prediction.
- Hyperparameter tuning had a **positive impact** on Random Forest and XGBoost, although the improvement was moderate.
- **Simple linear models are insufficient** for complex, multi-variable datasets where nonlinear interactions are present.

In conclusion, the project successfully demonstrated that advanced ensemble models, particularly **XGBoost**, are highly effective for predicting house prices, offering significantly higher accuracy compared to traditional methods like Linear Regression.

5. FUTURE WORK

While the project achieved strong results using Random Forest and XGBoost, there are several areas where future work and enhancements can be considered:

1. Exploration of Other Regression Algorithms:

- Implement and compare additional advanced regression techniques such as:
 - **LightGBM** (Light Gradient Boosting Machine)
 - **CatBoost** (especially useful for categorical features)
 - **Support Vector Regression (SVR)**
 - **K-Nearest Neighbors (KNN) Regressor**
- These models might offer even better performance depending on data characteristics.

2. Handling of Missing Values:

- Explore more sophisticated imputation methods like **KNN Imputer** or **Iterative Imputer** instead of simple median/mode filling to better preserve data patterns.

3. Model Stacking and Blending:

- Combine multiple models through **stacking** or **blending techniques** to leverage the strengths of different algorithms and achieve even better accuracy.

4. Hyperparameter Tuning Optimization:

- Use more advanced search techniques like **Bayesian Optimization** or **Optuna** instead of GridSearchCV for faster and more efficient hyperparameter tuning.

5. Dealing with Outliers More Effectively:

- Instead of simple removal, explore robust scaling methods or transformation techniques (e.g., log transformation) to handle outliers without losing valuable information.

6. REFERENCES

1. Kaggle Dataset:
House Prices - Advanced Regression Techniques, Kaggle.
Link: <https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques>
2. Scikit-learn Documentation:
Used for machine learning model implementations, preprocessing, and evaluation.
Link: <https://scikit-learn.org/stable/>
3. XGBoost Documentation:
Used for understanding and implementing the XGBoost Regressor model.
Link: <https://xgboost.readthedocs.io/en/stable/>
4. Seaborn and Matplotlib Documentation:
For data visualization tasks.
Seaborn: <https://seaborn.pydata.org/>
Matplotlib: <https://matplotlib.org/>
5. Pandas and NumPy Documentation:
For data manipulation and numerical operations.
Pandas: <https://pandas.pydata.org/>
NumPy: <https://numpy.org/>