Alt School

PREDICTIVE MODELLING OF HOUSE PRICES

A comprehensive report on machine learning project

In pursuit of a data science diploma desgree from Alsthool Africa, Nigeria

Prepared by: Oladokun Sunday Obasanjo

Table Of Contents

List of Figures & Visualizations

01

Project Overview

In this capstone project, I applied my data science and machine learning skills to develop a predictive model for house prices. This project involved several key steps, including data cleaning, exploratory data analysis (EDA), feature engineering, model training, and evaluation.

By leveraging a real-world dataset, I built a model designed to accurately predict house prices based on a range of features. The goal was to create a robust predictive model that provides valuable insights into the factors influencing property values.

02

Problem Statement

In the real estate market, a lot of factors can affect property values, making it tough to accurately predict house prices. These factors are inclusive of the physical elements relating to the property such as its size, location and state as well as external variables that comprise market trends and economic conditions. The complexity stems from these variables, which interact with one another differently and affect property prices in different proportions. For example, the number of bedrooms in a house may adjust its price range significantly along with other elements like the overall condition of the house in question or neighborhood facilities.

Accurate predictions of house prices can offer substantial benefits across multiple stakeholders in the real estate sector. For buyers, precise price predictions can facilitate more informed purchasing decisions, ensuring that they are not overpaying for a property. Sellers can benefit from understanding the fair market value of their property, which helps in setting a competitive price that can attract potential buyers while maximizing their return. Real estate agents gain from accurate price predictions by being able to provide better advice to their clients and streamline the buying and selling process. Financial institutions and investors also rely on accurate property valuations to make sound investment decisions and assess the risk associated with mortgage loans.

This project aims to address the challenge of accurate property valuation by developing a robust machine learning model for predicting house prices. By leveraging various features of the houses, such as their physical characteristics, location, and condition, the project seeks to build a model that can effectively capture the complex relationships between these features and the sale price. The model will be trained and evaluated using a comprehensive dataset, enabling it to learn patterns and make accurate predictions based on historical data. Through this approach, the project

intends to provide a reliable tool for stakeholders in the real estate market to better understand and predict property values.

03 Project Objectives

3.1: General Objective:

The primary objective of this project is to accurately predict house prices using machine learning models and identify the most effective model that can be deployed as a solution for stakeholders in the real estate industry.

3.2: Specific Objectives:

· Data Cleaning and Exploratory Data Analysis (EDA): Prepare and understand the dataset through thorough cleaning and analysis.

· Feature Engineering and Encoding: Enhance the model's predictive capabilities by engineering features and encoding variables appropriately.

· Model Training, Evaluation, and Optimization: Train, evaluate, and optimize multiple machine learning models to determine the best-performing one for predicting house prices.

04 Project Methodology

4.1: Data Collection

The dataset utilized for this project is titled "Housing Sales: Factors Influencing Sale Prices" from Kaggle. It is a comprehensive collection of housing sales data organized in a CSV (Comma-Separated Values) format.

Each row represents a unique property sale, while the columns provide extensive information on various property attributes. These attributes include lot size, building type, house style, condition ratings, year of construction, amenities, and sale prices.

This dataset is particularly useful for analysing real estate market trends, understanding how property characteristics and location influence sale prices, and developing predictive models for housing sales. The dataset can be accessed here.

The size of the dataset is 18 features and 2,413 observations. Out of the 18 features contained in the dataset, 14 are numerical while 4 are categorical. The features are described below:

1. **Lot_Frontage: Linear feet of street connected to the property.**
2. **Lot_Area: Lot size in square feet.**
3. **Bldg_Type: Type of building (e.g., single-family, multi-family).**
4. **House_Style: Style of the house (e.g., ranch, two-story).**
5. **Overall_Cond: Overall condition rating of the house.**
6. **Year_Built: Year the house was built.**
7. **Exter_Cond: Exterior condition rating of the house.**
8. **Total_Bsmt_SF: Total square feet of basement area.**
9. **First_Flr_SF: First-floor square feet.**

10. **Second_Flr_SF: Second-floor square feet.**
11. **Full_Bath: Number of full bathrooms.**
12. **Half_Bath: Number of half bathrooms.**
13. **Bedroom_AbvGr: Number of bedrooms above ground.**
14. **Kitchen_AbvGr: Number of kitchens above ground.**
15. **Fireplaces: Number of fireplaces.**
16. **Longitude: Longitude coordinates of the property location.**
17. **Latitude: Latitude coordinates of the property location.**
18. **Sale_Price: Sale price of the property (target feature for prediction).**

	0	1	2	3	4
Lot_Frontage	141	80	81	93	74
Lot_Area	31770	11622	14267	11160	13830
Bldg_Type	OneFam	OneFam	OneFam	OneFam	OneFam
House_Style	One_Story	One_Story	One_Story	One_Story	Two_Story
Overall_Cond	Average	Above_Average	Above_Average	Average	Average
Year_Built	1960	1961	1958	1968	1997
Exter_Cond	Typical	Typical	Typical	Typical	Typical
Total_Bsmt_SF	1080	882	1329	2110	928
First_Flr_SF	1656	896	1329	2110	928
Second_Flr_SF	0	0	0	0	701
Full_Bath	1	1	1	2	2
Half_Bath	0	0	1	1	1
Bedroom_AbvGr	3	2	3	3	3
Kitchen_AbvGr	1	1	1	1	1
Fireplaces	2	0	0	2	1
Longitude	-93.619754	-93.619756	-93.619387	-93.61732	-93.638933
Latitude	42.054035	42.053014	42.052659	42.051245	42.060899
Sale_Price	215000	105000	172000	244000	189900

Figure 4.1: The dataset showing the first five rows

```
# Check the size of the dataset
data_size = df.shape

print(f'The training set has {data_size[0]} rows (observations) and {data_size[-1]} columns (features)')
```

✓ 0.0s

```
The training set has 2413 rows (observations) and 18 columns (features)
```

Figure 4.2: Size of the dataset

4.2: Data Cleaning

Data Cleaning is an essential part of machine learning because what the model is fed consequently affects the output of the model.

The data cleaning was carried out by doing the following:

1. Typographical Error Correction: I inspected categorical features ('Bldg_Type', 'House_Style', 'Overall_Cond', 'Exter_Cond') by extracting unique values from the dataset using pandas and cross-checking them with the data description on Kaggle. All values were found to be in order.

```
# Extract categorical features as a list
categorical_features = df.select_dtypes(include=['object', 'category']).columns.tolist()

# Extract numerical features as a list
numerical_features = df.select_dtypes(include=['number']).columns.tolist()

# Print the different categories as a list
print("Categorical features:", categorical_features)
print("Numerical features:", numerical_features)
```

✓ 0.0s

```
Categorical features: ['Bldg_Type', 'House_Style', 'Overall_Cond', 'Exter_Cond']
Numerical features: ['Lot_Frontage', 'Lot_Area', 'Year_Built', 'Total_Bsmt_SF', 'First_Flr_SF', 'Second_Flr_SF', 'Full_Bath', 'Half_Bath', 'Bedroom_AbvGr', 'Kitchen_AbvGr', 'Fireplaces', 'Longitude',
'Latitude', 'Sale_Price']
```

Figure 4.3: Creation of a list of the numerical and categorical features

```
# Subset the object type features for inspection
object_data_df = df[categorical_features]

# Print the first five rows of the subsetted features
object_data_df.head()
```

✓ 0.0s

	Bldg_Type	House_Style	Overall_Cond	Exter_Cond
0	OneFam	One_Story	Average	Typical
1	OneFam	One_Story	Above_Average	Typical
2	OneFam	One_Story	Above_Average	Typical
3	OneFam	One_Story	Average	Typical
4	OneFam	Two_Story	Average	Typical

Figure 4.4: Creating subsets of numerical features and categorical features dataframes

```
# View the unique entries in each feature to take a look at each entry for inspection against misspellings, wrong characters and typographical errors

# Loop through each feature (column) in the DataFrame containing object (categorical) data
for feature in object_data_df.columns:

    # Get the unique entries for the current feature
    unique_entries = object_data_df[feature].unique()

    # Print out the feature name and its unique entries as a list
    print(f'The unique entries in {feature} are {unique_entries}')
```

✓ 0.0s

```
The unique entries in Bldg_Type are ['OneFam' 'TwnhsE' 'Twnhs' 'Duplex' 'TwoFmCon']
The unique entries in House_Style are ['One_Story' 'Two_Story' 'One_and_Half_Fin' 'SLvl' 'SFoyer'
 'Two_and_Half_Unf' 'One_and_Half_Unf' 'Two_and_Half_Fin']
The unique entries in Overall_Cond are ['Average' 'Above_Average' 'Good' 'Very_Good' 'Poor' 'Below_Average'
 'Excellent' 'Fair' 'Very_Poor']
The unique entries in Exter_Cond are ['Typical' 'Good' 'Fair' 'Poor' 'Excellent']
```

Figure 4.5: For loop to check the unique entries in each of the categorical features

2. Numerical Feature Validation: I performed a similar check for numerical features(Just as the categorical features) ('Lot_Frontage', 'Lot_Area', 'Year_Built', 'Total_Bsmt_SF', 'First_Flr_SF', 'Second_Flr_SF', 'Full_Bath', 'Half_Bath', 'Bedroom_AbvGr', 'Kitchen_AbvGr', 'Fireplaces', 'Longitude', 'Latitude', 'Sale_Price') to identify any unusual characters. These features were also found to be clean, indicating that the Kaggle dataset was relatively well-maintained.

```
# Subset the numerical type features for inspection
numerical_data_df = df[numerical_features]

# Print the first five rows of the subset
numerical_data_df.head()
```

✓ 0.0s

	Lot_Frontage	Lot_Area	Year_Built	Total_Bsmt_SF	First_Flr_SF	Second_Flr_SF	Full_Bath	Half_Bath	Bedroom_AbvGr	Kitchen_AbvGr	Fireplaces	Longitude	Latitude	Sale_Price
0	141	31770	1960	1080	1656	0	1	0	3	1	2	-93.619754	42.054035	215000
1	80	11622	1961	882	896	0	1	0	2	1	0	-93.619756	42.053014	105000
2	81	14267	1958	1329	1329	0	1	1	3	1	0	-93.619387	42.052659	172000
3	93	11160	1968	2110	2110	0	2	1	3	1	2	-93.617320	42.051245	244000
4	74	13830	1997	928	928	701	2	1	3	1	1	-93.638933	42.060899	189900

Figure 4.6: Subset of numerical features

3. Handling "Year Built": The "Year Built"(Year the house was built) feature is an integer and it would be useful later to engineer new features such as age of the house. Converting it to a datetime format is unnecessary as it would default to January 1st of each year, which lacks precision. Instead, if needed, I can calculate the age of the houses by subtracting the year from 2024 and adding this information to the dataset later.

```
# Inspecting Year Built- the only feature that has to do with time
df[['Year_Built']]
```

✓ 0.0s

	Year_Built
0	1960
1	1961
2	1958
3	1968
4	1997
...	...
2408	1984
2409	1983
2410	1992
2411	1974
2412	1993

2413 rows × 1 columns

Figure 4.7: Subset of the year built feature

4. Data Quality Issues: There were no missing values or duplicate records, but I identified 561 outliers, which constitute a significant portion of the dataset.

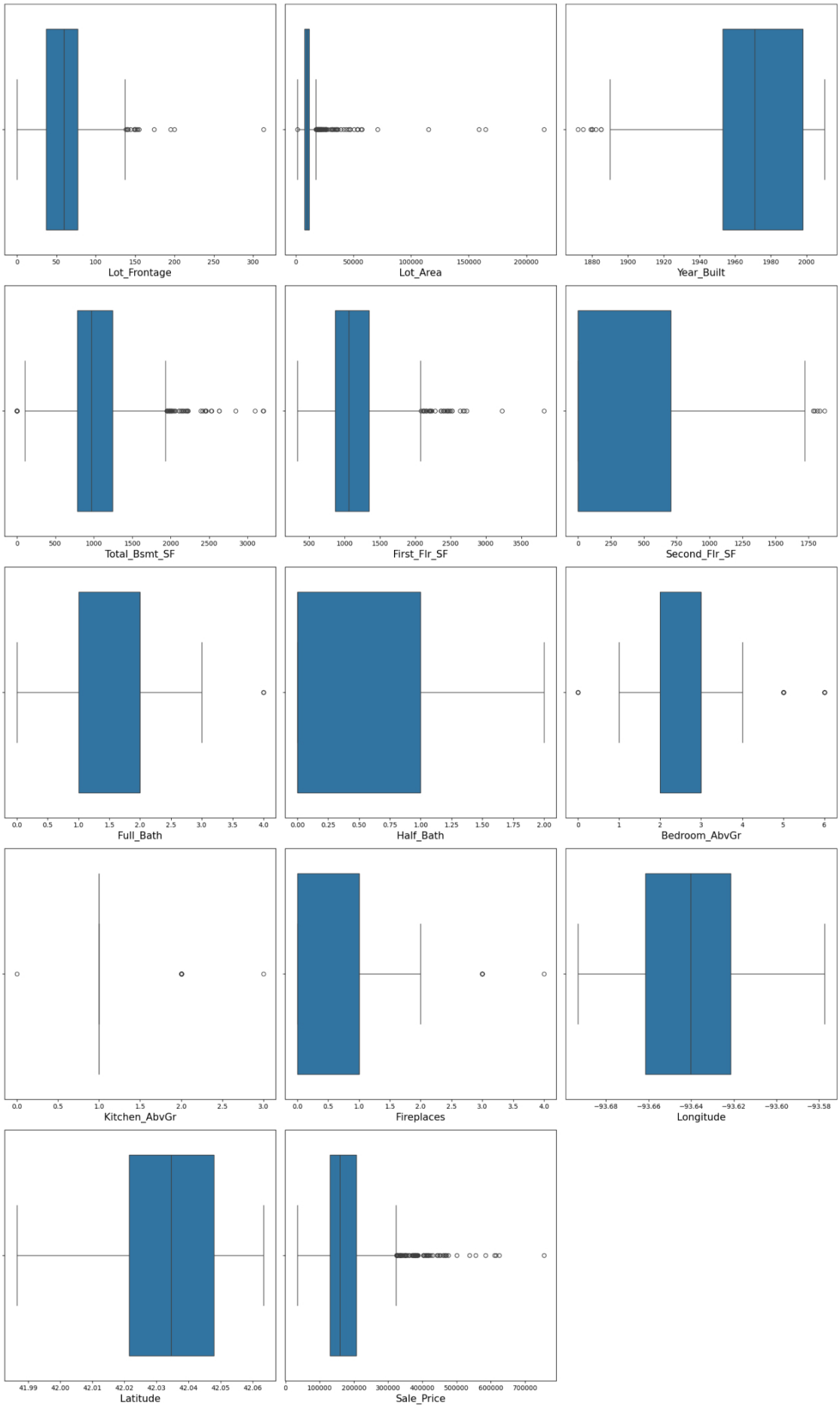


Figure 4.8: Boxplot showing outliers across the numerical features

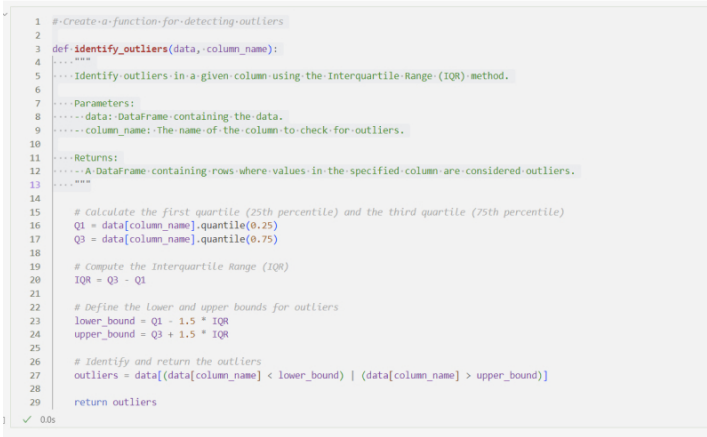
4.3: Handling Outliers (Extreme Values)

561 outliers were identified which constitute a significant portion of the dataset. To see the effect of the outliers, I made a copy of the dataset, removed outliers by dropping them and then an analysis of variance (ANOVA) was carried out between the dataframe with outliers and the one without.

The F-statistic of 17.5570 indicates a significant difference in `Sale_Price` between outliers and non-outliers. The p-value of 2.8887e-05 confirms this difference is statistically significant. The two figures suggest that outliers will impact `Sale_Price` differently than non-outliers and may affect model performance.

Despite the discovery, I decided to make the hard choice of keeping two datasets for my machine learning phase- one with outliers and the other one without outliers

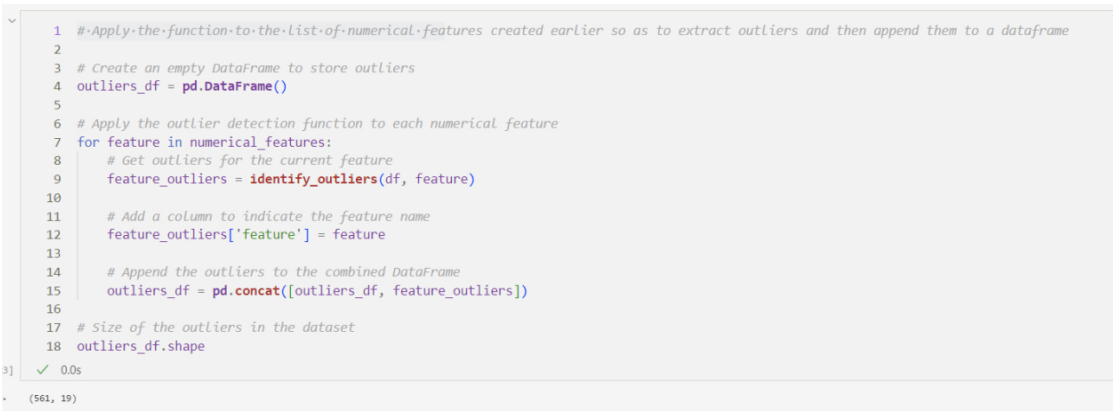
I will evaluate the performance of machine learning models of choice on both datasets.



```python
# Create a function for detecting outliers

def identify_outliers(data, column_name):
    """
    Identify outliers in a given column using the Interquartile Range (IQR) method.

    Parameters:
        data: DataFrame containing the data.
        column_name: The name of the column to check for outliers.

    Returns:
        A DataFrame containing rows where values in the specified column are considered outliers.
    """

    # Calculate the first quartile (25th percentile) and the third quartile (75th percentile)
    Q1 = data[column_name].quantile(0.25)
    Q3 = data[column_name].quantile(0.75)

    # Compute the Interquartile Range (IQR)
    IQR = Q3 - Q1

    # Define the lower and upper bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify and return the outliers
    outliers = data[(data[column_name] < lower_bound) | (data[column_name] > upper_bound)]

    return outliers
```

0.0s

Figure 4.9: Function to check and return outliers from dataset



```python
# Apply the function to the list of numerical features created earlier so as to extract outliers and then append them to a dataframe

# Create an empty DataFrame to store outliers
outliers_df = pd.DataFrame()

# Apply the outlier detection function to each numerical feature
for feature in numerical_features:
    # Get outliers for the current feature
    feature_outliers = identify_outliers(df, feature)

    # Add a column to indicate the feature name
    feature_outliers['feature'] = feature

    # Append the outliers to the combined DataFrame
    outliers_df = pd.concat([outliers_df, feature_outliers])

# Size of the outliers in the dataset
outliers_df.shape
```

0.0s

```
(561, 19)
```

Figure 4.10: Applying the function to dataset which returns the size of the outliers

4.4: Exploratory Data Analysis (EDA)

I carried out EDA on my dataframe (df) with outliers to better understand the data, the only time I will be bringing df2(without outliers) into the picture again would be during the machine learning phase.

Before I dive into really exploring my data, its important to split into train and test datasets for the following reasons:

1. Prevents data leakage: By splitting before EDA, training set is isolated from the test set, preventing any information transfer.

2. Mimics real-world scenario: this will stimulate the model to perform on unseen data

```
# Set the random seed for reproducibility
random_seed = 42

# Define the fraction of data to be used for the training set
train_fraction = 0.8

# Sample the training data
train_df = df.sample(frac=train_fraction, random_state=random_seed)

# Get the test data by dropping the training indices
test_df = df.drop(train_df.index)
```

✓ 0.0s

Figure 4.11: Splitting the data with outliers into training and test set before EDA

```
# View the training set
train_df.head().T
```

✓ 0.0s

	765	2387	2162	1833	1814
Lot_Frontage	85	54	60	79	120
Lot_Area	10200	13811	10800	9245	10356
Bldg_Type	OneFam	OneFam	OneFam	OneFam	OneFam
House_Style	One_Story	One_Story	One_and_Half_Fin	Two_Story	One_Story
Overall_Cond	Average	Above_Average	Very_Good	Average	Above_Average
Year_Built	2007	1987	1936	2006	1975
Exter_Cond	Typical	Typical	Typical	Typical	Typical
Total_Bsmt_SF	1578	1112	796	939	969
First_Flr_SF	1602	1137	1096	939	969
Second_Flr_SF	0	0	370	858	0
Full_Bath	2	2	2	2	1
Half_Bath	0	0	0	1	1
Bedroom_AbvGr	3	2	3	3	3
Kitchen_AbvGr	1	1	1	1	1
Fireplaces	1	1	1	0	0
Longitude	-93.684115	-93.646099	-93.613899	-93.684137	-93.684354
Latitude	42.016468	41.999553	42.034761	42.014823	42.021025
Sale_Price	293200	176000	170000	213500	122000
IsOutlier	False	True	True	False	True
HouseAge (Years)	17	37	88	18	49
DecadeBuilt	2000	1980	1930	2000	1970

Figure 4.12: Training set which accounts for 80%

The EDA was then carried out in 4 different phases on the train set created from the split:

1. Non-graphical univariate analysis explored frequency distributions for categorical features and statistical summaries for numerical features. Skewness and unique values were also checked.

2. Graphical univariate analysis explored histograms and boxplots for numerical features and bar plots for categorical features.

3. Non-graphical multivariate analysis explored relationships between features and the target feature majorly to understand how much of a relationship exists. Correlation analysis was carried out to get the co-efficients for each of the features. Groupby analysis was also carried out and finally analysis of variance was carried out to check if the means of Sale_Price across different categories of categorical variables are significantly different.

4. Graphical multivariate analysis explored the use of pairplots to visualize the relationship between multiple features and the target variable (Sales Price). A correlation heatmap was created as well.

4.5: Feature Engineering

After the EDA, new features were created that could enhance the performance of the models that would be used later on. The features that were created are House age and Decade the house was built.

This was done for the test set and also the dataframe without outliers that would be utilized during machine learning phase.

The new features were correlated with the target feature to see which one would be useful during modelling.

4.6: Feature Selection for Machine Learning

This phase was used to select features that are necessary for my machine learning as a result of their effect on the target variable as established from the exploratory data analysis carried out.

The summary of the effect each of the features have on the target feature we are predicting (house prices) were categorised into strong, moderate, and weak relationship for numerical features while ANOVA was used for the categorical features.

The numerical features with strong or moderate relationship with the target feature were selected with the categorical features with significant impact were also selected to be used during the machine learning phase.

Strong Positive Correlations:
- Basement Area
- First Floor Area
- Full Bathrooms

Moderate Positive Correlations:
- Year Built
- Fireplaces
- Decade the house was built

Significant Impact on Sale Price:
- Building Type
- House Style
- Overall Condition
- Exterior Condition.

Finally, three datasets were generated with the features needed for machine learning phase:

1. Train df (with outliers)
2. Test df (with outliers)
3. Df without outliers

4.7: Model Selection

Linear Regression:

- Simplicity and Interpretability: Linear Regression is straightforward and provides clear insights into the relationships between features and the target variable. This makes it easy to interpret how each feature influences house prices.

- Baseline Model: It serves as a good baseline model. If the Linear Regression model performs well, it can indicate that the relationships between features and prices are relatively simple and linear.

- Efficiency: It is computationally efficient and requires less processing power compared to more complex models.

Random Forest Regressor:

- Handling Non-Linearity: Random Forest Regressor can model complex, non-linear relationships between features and the target variable, which is useful for capturing intricate patterns in housing data.

- Feature Importance: It provides insights into feature importance, helping to understand which attributes are most influential in predicting house prices.

- Robustness: It is less prone to overfitting compared to individual decision trees and can handle noisy data and missing values better.

XGBoost Regressor:

- Performance: XGBoost is known for its high predictive accuracy and efficiency. It often performs better than other models due to its advanced boosting technique.

- Handling Large Datasets: It can handle large datasets and high-dimensional features effectively, making it suitable for complex housing datasets.

- Feature Engineering: XGBoost can automatically handle various feature interactions and is robust to overfitting, which enhances model performance and generalization.

4.8: Data Preparation For Modelling

The categorical features in the datasets were converted to numerical values as follows:

The categorical ordinal features were mapped to a number scale created for them while the nominal features were encoded using one hot encoding.

4.9: Model Training and Evaluation

The three models were used to train and test the data with and without outliers, they were then evaluated using root mean square error, mean squared error, and R2 scores.

Finally, scatter plots and bar charts were created to evaluate the best model.

05 Results & Discussion

5.1: Categorical Features Analysis

The analysis of categorical variables within the dataset reveals distinct patterns and distributions among property characteristics. Bldg_Type (Type of building, e.g., single-family, multi-family) shows that "OneFam" is the predominant building type, making up 82.12% of the dataset, while "TwnhsE" follows as a distant second with 8.03%. Less common types such as "Twnhs," "Duplex," and "TwoFmCon" collectively represent a minor proportion of the dataset, underscoring the prevalence of single-family homes.

House_Style (Style of the house, e.g., ranch, two-story) further supports this observation, with "One_Story" houses comprising nearly half of the dataset at 49.64%, followed by "Two_Story" homes at 29.84%. The rarity of styles like "Two_and_Half_Fin" at just 0.21% highlights the dominance of more traditional house styles within the sample.

Regarding Overall_Cond (Overall condition rating of the house), over half of the properties are rated "Average," indicating a general trend towards moderately maintained homes. Only a small percentage fall into the "Poor" or "Very_Poor" categories, suggesting that the dataset primarily consists of homes in at least acceptable condition.

In terms of Exter_Cond (Exterior condition rating of the house), a significant majority (86.79%) of the homes are classified as "Typical," which may reflect a standardization in exterior conditions within the sample area. The minimal presence of properties in "Poor" condition (0.05%) again indicates that the dataset mainly covers well-maintained properties.

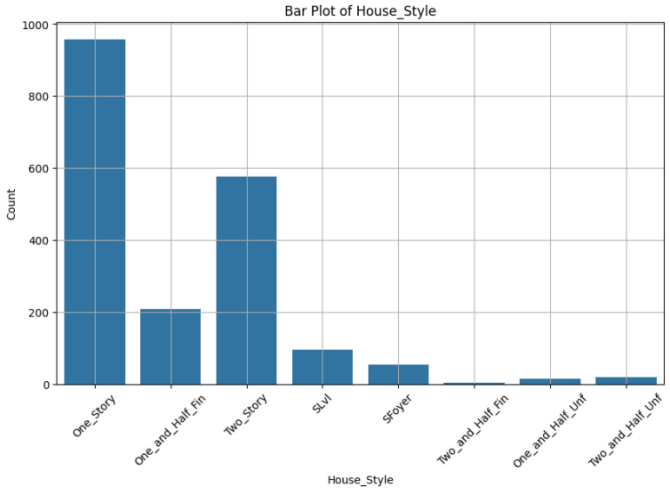


Figure 5.1: Bar plot of the different house styles

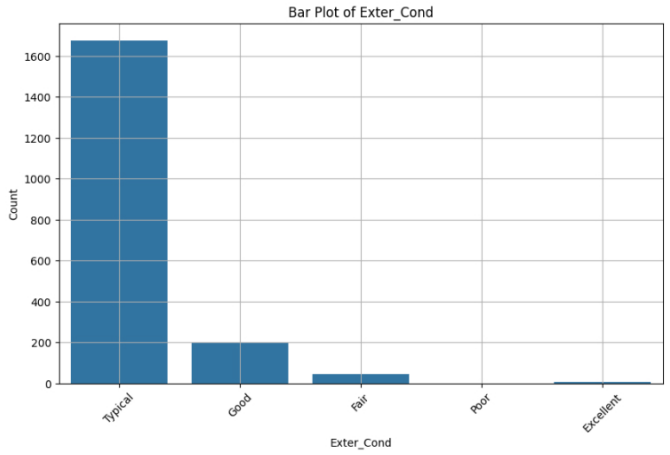


Figure 5.2: Bar plot exterior condition of the houses

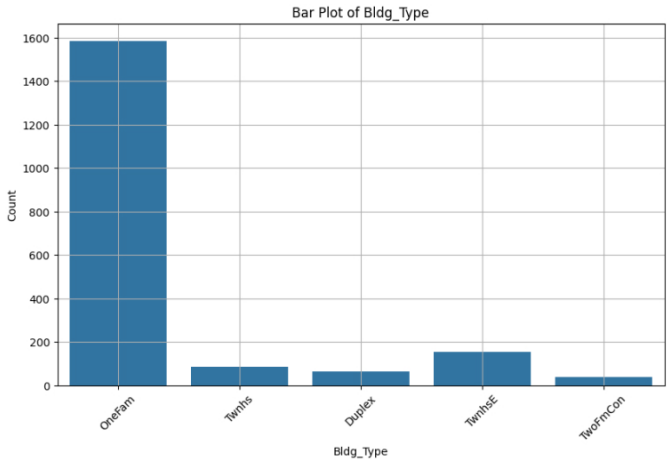


Figure 5.3: Bar plot of the different building types

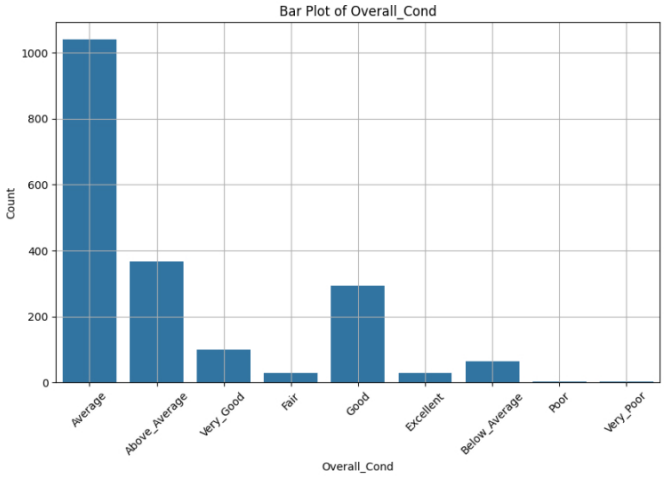


Figure 5.4: Bar plot of overall conditions of the houses

5.2: Numerical Features Analysis

Analyzing the continuous variables reveals notable trends and variations. Lot_Frontage (Linear feet of street connected to the property) averages around 55 feet, with a moderate standard deviation, indicating that most properties have similar lot widths, though some variability exists.

The Lot_Area (Lot size in square feet) shows greater disparity, with the average lot size at 10,051 square feet but a large standard deviation, signaling significant differences in property sizes.

Year_Built (Year the house was built) analysis reveals that most homes were constructed around 1970, with the data encompassing a wide historical range from 1872 to 2010. This spread suggests a mix of older and newer homes, with a median construction period in the mid-20th century.

Total_Bsmt_SF (Total square feet of basement area) and First_Flr_SF (First-floor square feet) provide insights into property sizes, with average basement and first-floor areas indicating standard residential dimensions, though the ranges and standard deviations reflect substantial variability.

The distribution of features such as Sale_Price (Sale price of the property, which is also the target feature this project aims to predict) highlights the broad range of property values within the dataset. The average sale price of $176,115, coupled with a substantial standard deviation, suggests significant economic diversity among the properties analyzed.

5.3: Distribution of Features

The distribution analyses further elucidate the underlying structure of the dataset. Variables like Lot_Area and Sale_Price are highly positively skewed, indicating a concentration of smaller or less expensive properties, with a few outliers representing very large or high-value properties.

Conversely, Year_Built and Longitude (Longitude coordinates of the property location) exhibit slight negative skewness, suggesting a tendency towards more recent construction and specific geographical clustering.

Kurtosis values, particularly for Lot_Area and Sale_Price, indicate the presence of heavy tails, reflecting extreme values that deviate significantly from the mean.

In contrast, features like Longitude and Latitude (Latitude coordinates of the property location) demonstrate nearly normal distributions, indicative of the dataset's geographical consistency.

Overall, the analysis of categorical variables and descriptive statistics provides a comprehensive overview of the dataset's characteristics, revealing significant trends and highlighting areas of variability that could influence further modeling efforts.

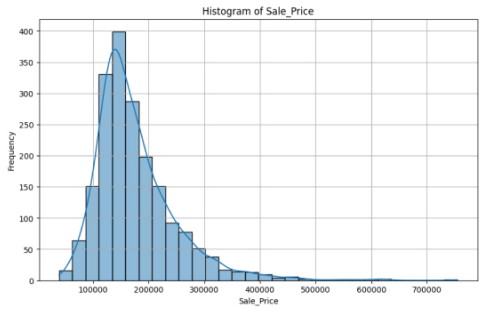


Figure 5.5: Distribution of sales prices (USD)

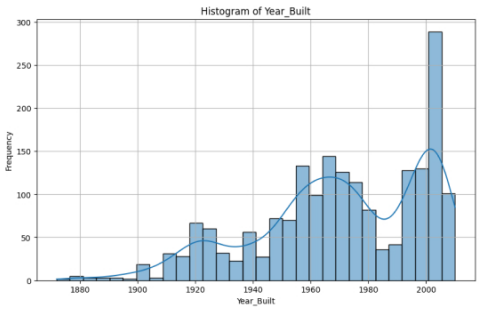


Figure 5.9: Distribution of years houses were built

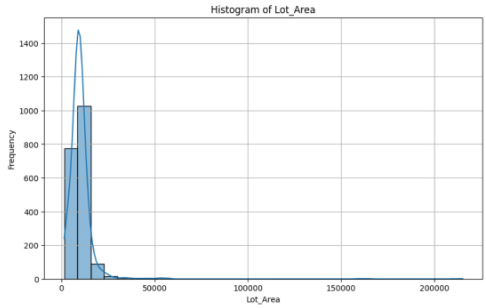


Figure 5.6: Distribution of Lot Area (ft)

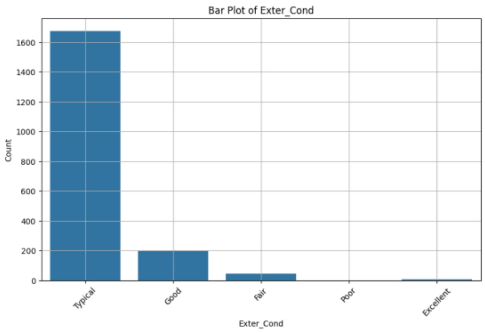


Figure 5.10: Distribution of condition of exterior condition of houses

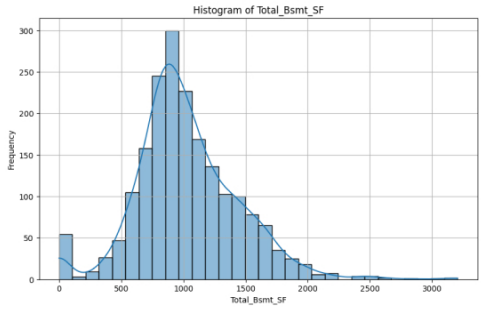


Figure 5.7: Distribution of Basement Area size(ft)

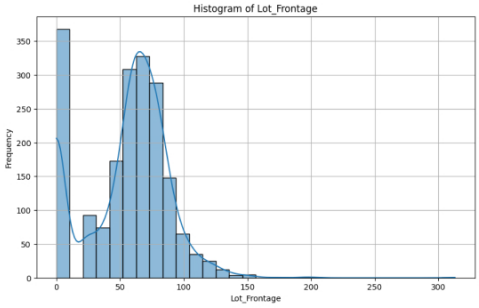


Figure 5.8: Distribution of Lot Frontage size (ft)

5.4:Correlation Analysis

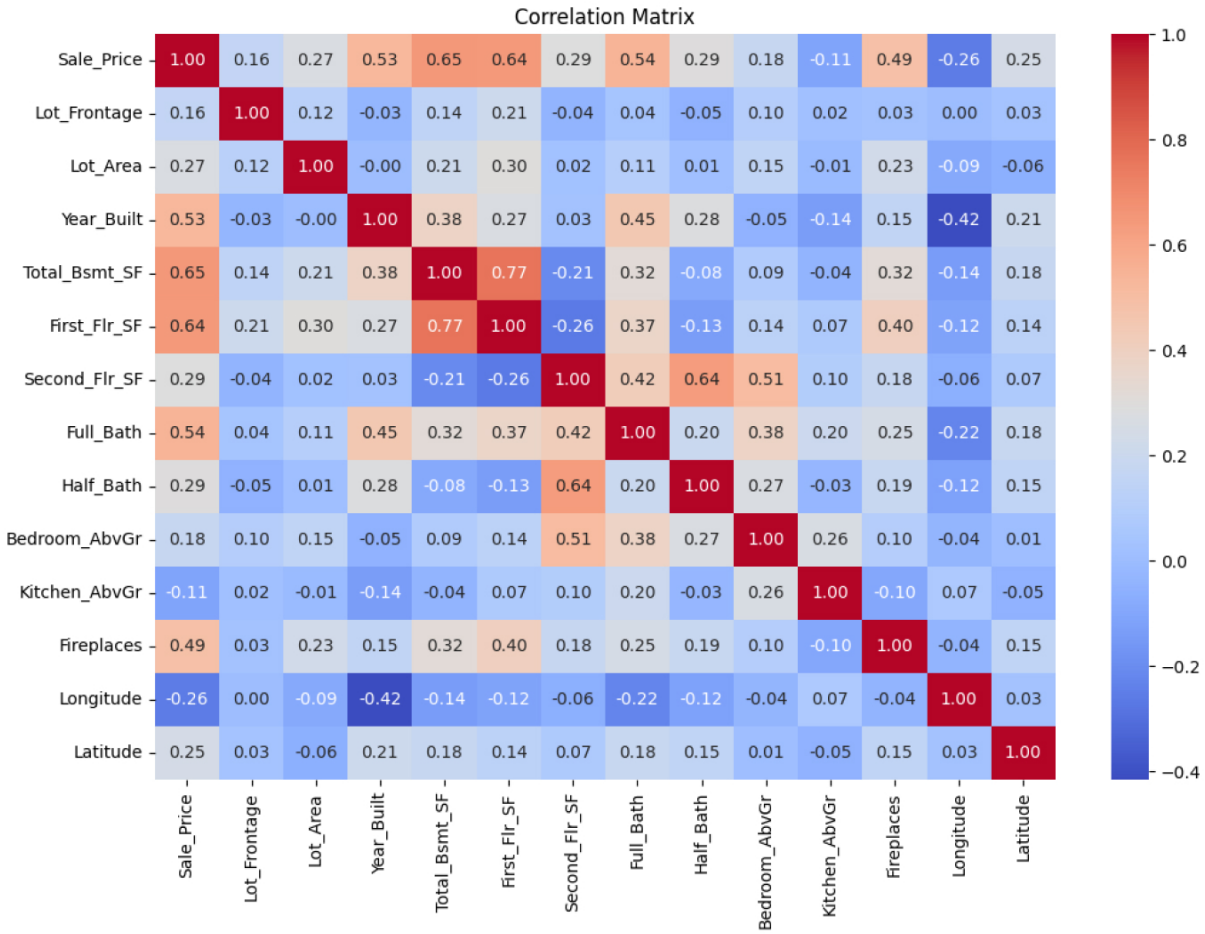


Figure 5.11: Heatmap of correlation of the different features

This section examines the relationships between each feature and the target variable, Sale_Price. The correlation analysis reveals how various features contribute to the property's value:

- Total_Bsmt_SF (0.6482): A strong positive correlation exists between the total basement area and the sale price, indicating that properties with larger basements tend to have higher values.

- First_Flr_SF (0.6448): The first-floor square footage also shows a strong positive correlation with sale price, suggesting that larger first floors are associated with higher property values.

- Full_Bath (0.5410): The number of full bathrooms has a moderate positive correlation with sale price, indicating that properties with more full bathrooms are likely to be valued higher.

- Year_Built (0.5284): The year the house was built is positively correlated with sale price, reflecting that newer homes generally command higher prices due to modern features and updates.

- Half_Bath (0.2926): The presence of half bathrooms has a weaker positive correlation with sale price. While it contributes to the property value, the impact is less significant compared to other features.

- Fireplaces (0.4897): The number of fireplaces has a moderate positive correlation with sale price, with homes featuring more fireplaces typically achieving higher sale prices.

- Second_Flr_SF (0.2904): The size of the second floor has a weak positive correlation with sale price, indicating that larger second floors are somewhat associated with higher property values.

- Lot_Area (0.2724): Lot size shows a weak positive correlation with sale price. While larger lots generally lead to higher values, the effect is less pronounced.

- Latitude (0.2455): The latitude of a property exhibits a weak positive correlation with sale price, possibly reflecting geographical variations in property values.

- Bedroom_AbvGr (0.1784): The number of bedrooms above ground has a weak positive correlation with sale price, suggesting a minor contribution to higher prices.

- Lot_Frontage (0.1631): Lot frontage has a weak positive correlation with sale price. Although larger frontages are associated with higher values, the relationship is not strong.

- Kitchen_AbvGr (-0.1076): The number of kitchens above ground shows a weak negative correlation with sale price, implying that more kitchens might slightly reduce the property's value, though the effect is minimal.

- Longitude (-0.2588): Longitude has a weak negative correlation with sale price, suggesting that properties located further east might have lower prices, although the relationship is weak.

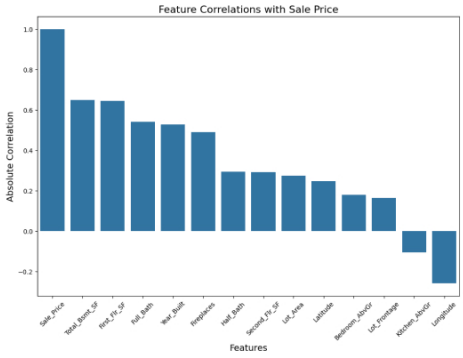


Figure 5.12: Feature correlation with house price

5.4: Analysis of Variance for Categorical Variables

- Bldg_Type: The very low p-value indicates statistically significant differences in sale prices among building types. The type of building significantly influences property values.

```
# ANOVA for Sale_Price across different Bldg_Type
f_stat, p_value = f_oneway(*[group["Sale_Price"].values for name, group in train_df.groupby("Bldg_Type")])
print(f"ANOVA F-statistic: {f_stat}, P-value: {p_value}")
```

```
ANOVA F-statistic: 15.995156967151086, P-value: 6.875397050056259e-13
```

Figure 5.13: ANOVA of sales prices vs building types

- House_Style: The extremely low p-value shows significant differences in sale prices based on house style, confirming that house style is a crucial determinant of the sale price.

```
# ANOVA for Sale_Price across different House_Style
f_stat, p_value = f_oneway(*[group["Sale_Price"].values for name, group in train_df.groupby("House_Style")])
print(f"ANOVA F-statistic: {f_stat}, P-value: {p_value}")
```

```
ANOVA F-statistic: 21.040251417889703, P-value: 1.849879289995216e-27
```

Figure 5.14: ANOVA of sales prices vs building styles

- Overall_Cond: The low p-value confirms significant differences in sale prices across overall condition ratings. The condition of a house strongly impacts its sale price.

```
# ANOVA for Sale_Price across different conditions of the houses
f_stat, p_value = f_oneway(*[group["Sale_Price"].values for name, group in train_df.groupby("Overall_Cond")])
print(f"ANOVA F-statistic: {f_stat}, P-value: {p_value}")
```

```
ANOVA F-statistic: 39.90803392208425, P-value: 3.2165023584025035e-59
```

Figure 5.15: ANOVA of sales prices vs overall condition of buildings

- Exter_Cond: The very low p-value suggests significant differences in sale prices depending on exterior condition ratings, with better exterior conditions correlating with higher property values.

```
# ANOVA for Sale_Price across different conditions of the exterior conditions
f_stat, p_value = f_oneway(*[group["Sale_Price"].values for name, group in train_df.groupby("Exter_Cond")])
print(f"ANOVA F-statistic: {f_stat}, P-value: {p_value}")
```

```
ANOVA F-statistic: 8.851055671801037, P-value: 4.4291025834429006e-07
```

Figure 5.16: ANOVA of sales prices vs exterior condition of buildings

5.5: Model Training and Evaluation

The three models (Linear Regression, Random Forest Regression, XG Boost Regressor) were trained on the data with outliers and without outliers and the following evaluations were gotten:

```
# Linear Regression Scores
print(f"Linear Regression with Outliers - MSE: {mse_lr_with_outliers}, R²: {r2_lr_with_outliers}, RMSE: {rmse_lr_with_outliers}")
print(f"Linear Regression without Outliers - MSE: {mse_lr_without_outliers}, R²: {r2_lr_without_outliers}, RMSE: {rmse_lr_without_outliers}")

# Random Forest Regression Scores
print(f"Random Forest Regression with Outliers - MSE: {mse_rfr_with_outliers}, R²: {r2_rfr_with_outliers}, RMSE: {rmse_rfr_with_outliers}")
print(f"Random Forest Regression without Outliers - MSE: {mse_rfr_without_outliers}, R²: {r2_rfr_without_outliers}, RMSE: {rmse_rfr_without_outliers}")

# XGBoost Regressor Scores
print(f"XGBoost Regressor with Outliers - MSE: {mse_xgb_with_outliers}, R²: {r2_xgb_with_outliers}, RMSE: {rmse_xgb_with_outliers}")
print(f"XGBoost Regressor without Outliers - MSE: {mse_xgb_without_outliers}, R²: {r2_xgb_without_outliers}, RMSE: {rmse_xgb_without_outliers}")
```

```
Linear Regression with Outliers - MSE: 657867159.9041076, R²: 0.8420130154533794, RMSE: 25648.92122300873
Linear Regression without Outliers - MSE: 608186193.5088243, R²: 0.8500533094320352, RMSE: 24661.431294813858
Random Forest Regression with Outliers - MSE: 635382871.8485961, R²: 0.8474126235902039, RMSE: 25206.80209484329
Random Forest Regression without Outliers - MSE: 546561254.9072628, R²: 0.8767419398296685, RMSE: 23378.649552685092
XGBoost Regressor with Outliers - MSE: 635382871.8485961, R²: 0.8474126235902039, RMSE: 25206.80209484329
XGBoost Regressor without Outliers - MSE: 181252490.7402407, R²: 0.9604702409527814, RMSE: 13463.004521288727
```

Figure 5.17: Evaluation of all models trained

Model	Outliers	MSE	R^2	RMSE
Linear Regression	With Outliers	657867159.9	0.842013015	25648.92122
Linear Regression	Without Outliers	608186193.5	0.850053309	24661.43129
Random Forest Regression	With Outliers	635382871.8	0.847412624	25206.80209
Random Forest Regression	Without Outliers	546561254.9	0.87674194	23378.64955
XGBoost Regressor	With Outliers	635382871.8	0.847412624	25206.80209
XGBoost Regressor	Without Outliers	181252490.7	0.960470241	13463.00452

Figure 5.18: Table of models and their different evaluations

Linear Regression Model:

- With Outliers: This model performs reasonably well, with a high R^2 indicating that it explains 84.2% of the variance in the data. However, the presence of outliers slightly increases the MSE and RMSE, suggesting that these outliers negatively affect the model's performance by introducing more prediction errors.

- Without Outliers: Removing outliers improved the model slightly, as indicated by the lower MSE and RMSE and a marginally higher R^2. The model is better at predicting values closer to the actual data without the influence of extreme outliers.

Random Forest Regression Model:

- With Outliers: This model performs slightly better than linear regression in handling outliers, as seen by a slightly lower MSE and RMSE and a marginally higher R^2. The model is more robust to outliers, thanks to the averaging of multiple decision trees.

- Without Outliers: Removing outliers significantly improved the model's performance, as shown by a substantial reduction in MSE and RMSE and an increase in R^2. This indicates that the random forest model can achieve better accuracy when outliers are excluded from the dataset.

XG Boost Regression Model:

- With Outliers: Similar to the random forest model, XGBoost handles outliers well, as reflected in comparable MSE, R^2, and RMSE scores. The performance is stable but does not significantly improve over random forest with outliers present.

- Without Outliers: XGBoost performs exceptionally well without outliers, with the lowest MSE and RMSE and the highest R^2 among all models. This suggests that XGBoost is the most effective model in this context, particularly when outliers are removed, as it captures the underlying patterns in the data more accurately.

5.6: Selection of best model

The best performing model is the XG Boost Regression Model with the following metrics:

- **MSE (Mean Squared Error): The average squared difference between predicted and actual prices is $181,252,490.74.**
- **RMSE (Root Mean Squared Error): The average prediction error is $13,463.00, which shows how close predictions are to the actual house prices. This means the prediction is off by 7.6% (Recall that the average house price is $176,115.**
- **R^2 (R-squared): The model explains 96% of the variation in house prices, meaning it fits the data very well.**

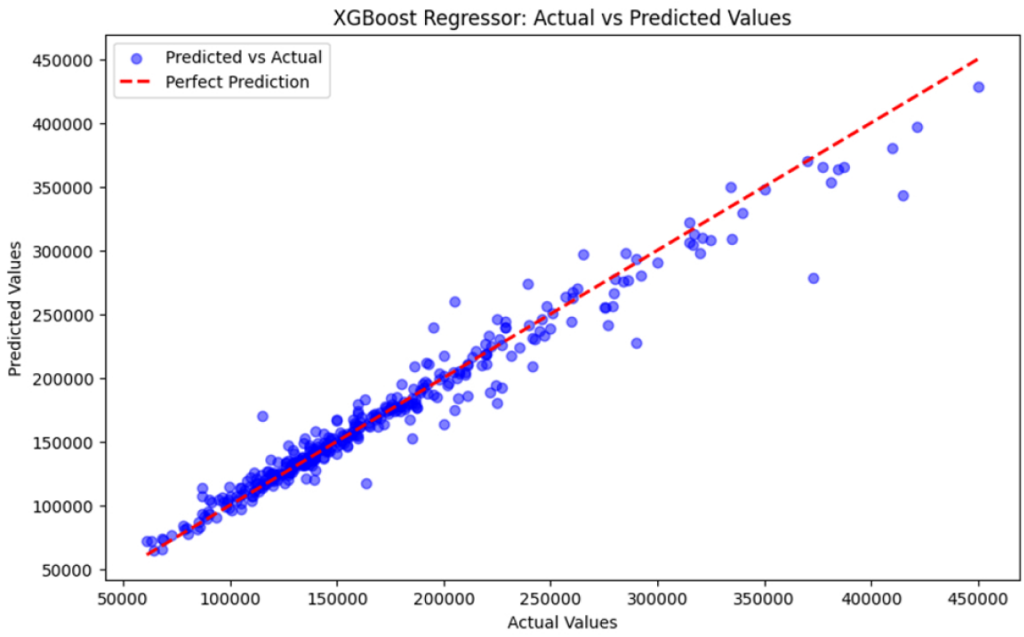


Figure 5.19: Scatter plot of predicted versus actual values for XG Boost Regressor Model

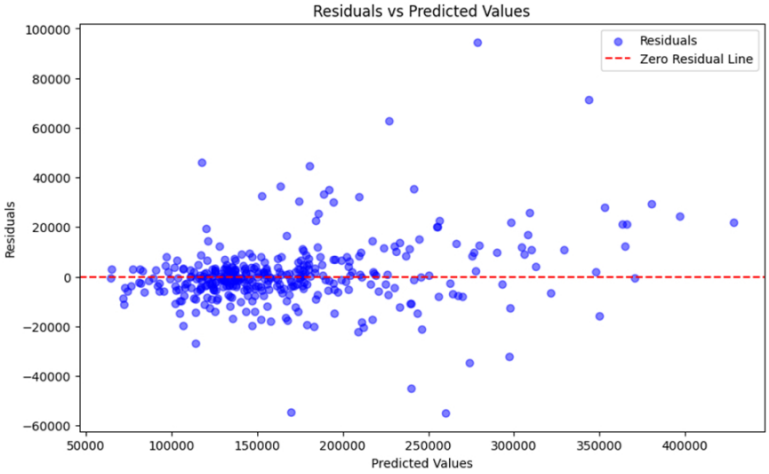


Figure 5.20: Residual vs Predicted values for XG Boost Regressor Model

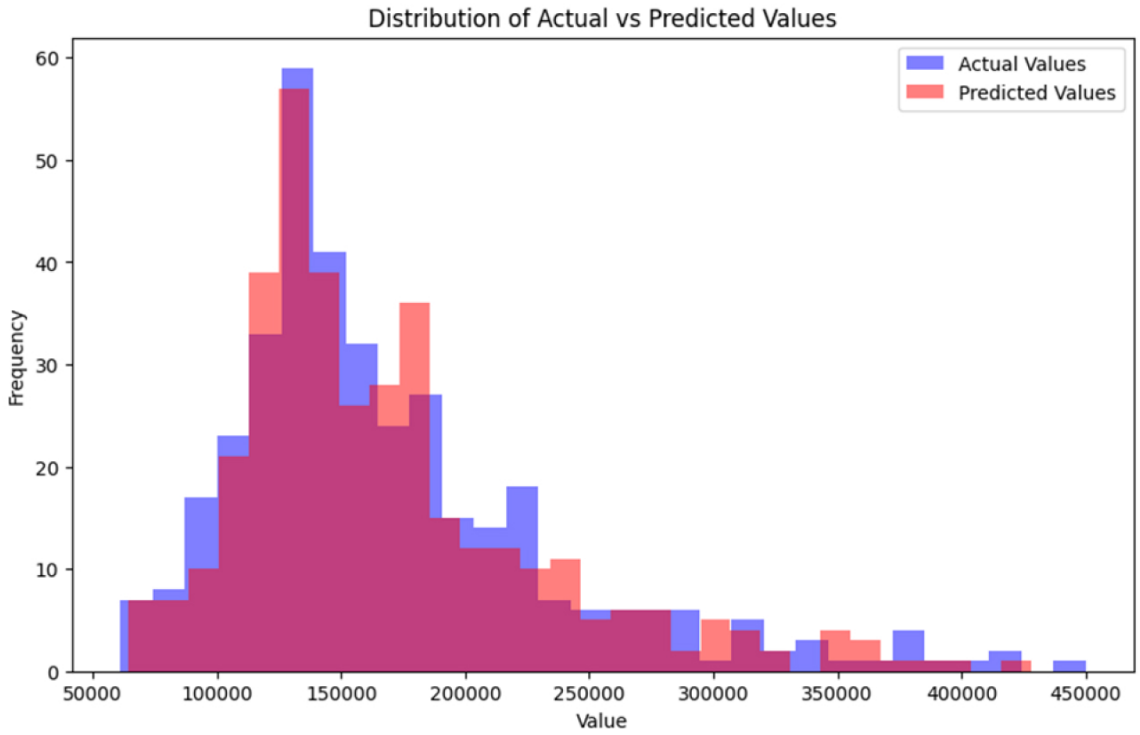


Figure 5.21: Distribution of Actual vs Predicted Values for XG Boost Regressor Model

06 Conclusion

The XGBoost Regressor emerged as the most effective model, particularly when trained on the dataset without outliers. This model achieved the lowest Mean Squared Error (MSE) of 181,252,490.74 and the lowest Root Mean Squared Error (RMSE) of 13,463.00, alongside a remarkably high R-squared (R^2) value of 0.960. These metrics indicate that the XGBoost Regressor not only minimized prediction errors but also explained a substantial proportion of the variance in the target variable, making it the most reliable model for this task.

The impact of outliers on model performance was significant across all models. The XGBoost Regressor, in particular, showed a dramatic improvement when outliers were removed, with its MSE dropping from 635,382,871.85 to 181,252,490.74 and its R^2 increasing from 0.847 to 0.960. This sharp contrast highlights the model's sensitivity to outliers and its potential to perform exceptionally well when these anomalies are effectively managed.

Similarly, the Random Forest Regression model also benefited from the removal of outliers, with its MSE decreasing from 635,382,871.85 to 546,561,254.91 and its R^2 improving from 0.847 to 0.877. Although the improvement was less dramatic than that of the XGBoost, it still underscores the importance of outlier management. The RMSE for Random Forest also reduced significantly, from 25,206.80 to 23,378.65, indicating better prediction accuracy after outlier removal.

On the other hand, Linear Regression demonstrated the most vulnerability to outliers. With outliers present, it produced an MSE of 657,867,159.90 and an R^2 of 0.842. Removing outliers led to a noticeable improvement, with the MSE reducing to 608,186,193.51 and the R^2 increasing to 0.850.

However, even with these gains, Linear Regression remained the least effective model in our analysis, particularly in comparison to the non-linear models like Random Forest and XGBoost.

Overall, the analysis reveals that outlier management is crucial for achieving accurate predictions across all models. The XGBoost Regressor's performance, especially its sharp improvement post-outlier removal, suggests that this model is highly sensitive to the presence of outliers but also highly capable when such data points are properly addressed.

In contrast, while Random Forest showed some resilience to outliers, it too benefited from their removal. Linear Regression, however, was the most affected by outliers, underscoring its limitations in handling such data and further emphasizing the superiority of more sophisticated models like XGBoost and Random Forest in complex predictive tasks.

07

Further Research

- Feature Engineering: Explore advanced techniques to create or transform features for better predictive accuracy.
- Model Tuning: Optimize hyperparameters across models to improve performance further.
- External Data Integration: Incorporate additional data sources like economic indicators or market trends to enhance model accuracy.
- Time-Series Analysis: Consider a time-series approach to analyze trends in house prices over time.

08 Project Links & Contact

Github Link:

https://github.com/Sunday-Oladokun/Project--Third-Semester-/blob/main/Predictive%20Modelling%20Of%20House%20Prices.ipynb

Contact Address

- Location: Abuja, Nigeria
- Email Address: anthonyoladokun@gmail.com
- Phone Number: +234 810 5757 307