CAPSTONE PROJECT

HOUSE PRICE PREDICTION

By Divya Vikram

PGDSBA.O.JAN23

Contents

List of Figures

List of Tables

Introduction

The problem statement for this machine learning project involves creating a model to predict the value of a house based on various features. The goal is to develop an algorithm that accurately estimates the market value of a property by considering factors such as neighborhood characteristics, property size, condition, upgrades, and other relevant features.

The dataset utilized in this study is derived from house sales in Washington state, USA. Data has been gathered between May 2014 and May 2015 and contains 21613 entries on 23 different house attributes.
Property dataset includes property's numerical attributes like ID, pricing, room details, living space, lot size, floors, construction details, location, and measurements in 2015.

Features also include categorical insights into waterfront view, viewing frequency, property condition, quality, location concentration, and furnishing preferences. The dataset reflects diverse attributes influencing property preferences and market trends. Refer to Columns description is in the **Appendix A**

- There are 12 float type, 4 integer and 7 object type variables in the dataset some of which are misclassified owing to some junk values in the column
- Several variables exhibit missing values
- Several variables exhibit junk values and have been classified as object type variables because of junk values, though some of these are numerical values.
- There are no duplicate entries in the dataset although CID which is property ID is not unique. There are multiple entries for same CID, which means that same property has been sold multiple times.

Need of solving

Comprehensive Property Valuation:
Existing methods often rely on limited factors like location and size. This project addresses the need for a more comprehensive approach to property valuation, taking into account a broader set of features that contribute to a house's value.
Accurate Pricing for Sellers:
Many homeowners struggle to set an appropriate selling price when putting their house on the market. This project aims to provide a data-driven solution, enabling sellers to make more informed decisions and avoid pricing their property too low or too high.
Enhanced Real Estate Decision-Making:
Real estate professionals can benefit from a more sophisticated valuation tool. This project can contribute to the real estate industry by providing a model that can assist agents, brokers, and appraisers in their decision-making processes.
Improved Market Transparency:
The project contributes to creating a more transparent real estate market by offering a model that considers a diverse range of factors. This can help reduce information asymmetry and empower both buyers and sellers with better insights into property values.

Exploratory Data Analysis (EDA)

Data summary of variables:

Features	Count of Non Null Values	Mean	Standard Deviation	Minumum Value	25.00%	50.00%	75.00%	Maximum Value
CID	21613	580301521	287656571	1000102	2.123E+09	3.905E+09	7.309E+09	9900000190
PRICE	21613	540182.16	367362.23	75000	321950	450000	645000	7700000
ROOM BED	21505	3.37	0.93	0	3	3	4	33
ROOM BATH	21505	2.12	0.77	0	1.75	2.25	2.5	8
LIVING MEASURE	21596	2079.86	918.5	290	1429.25	1910	2550	13540
LOT MEASURE	21571	15104.58	41423.62	520	5040	7618	10684.5	1651359
SIGHT	21556	0.23	0.77	0	0	0	0	4
QUALITY	21612	7.66	1.18	1	7	7	8	13
CEIL MEASURE	21612	1788.37	828.1	290	1190	1560	2210	9410
BASEMENT	21612	291.52	442.58	0	0	0	560	4820
YR RENOVATED	21613	84.4	401.68	0	0	0	0	2015
ZIPCODE	21613	98077.94	53.51	98001	98033	98065	98118	98199
LAT	21613	47.56	0.14	47.16	47.47	47.57	47.68	47.78
LIVING MEASURE15	21447	1987.07	685.52	399	1490	1840	2360	6210
LOT MEASURE15	21584	12766.54	27286.99	651	5100	7620	10087	871200
FURNISHED	21584	0.2	0.4	0	0	0	0	1
	Count of Non Null		No of Unique Values		Frequently		Frequency	
DAYHOURS	21613		372		20140623T000000		142	
CEIL	21571		7		1		10647	
COAST	21612		3		0		21421	
CONDITION	21556		6		3		13978	
YR BUILT	21612		117		2014		559	
LONG	21613		753		-122.29		116	
TOTAL AREA	21584		11145		$		39	

Table 1 Data summary of variables

All variables are on different scales. Several variables exhibit missing values: 'room_bed' , 'room_bath', 'living_measure' , 'lot_measure' , 'ceil' , 'coast', 'sight' and 'condition', 'quality' , 'ceil_measure' , 'basement', 'yr_built' , 'living_measure15' , 'lot_measure15,' 'furnished,' and 'total_area' as seen from count of Nulls

Uni-variate Analysis:

The CID variable is not relevant to our analysis. We need it to check for duplicate CID entries, of which there are 176. These indicate that the same property has had several sales in the past. We can remove the CID variable from your analysis because it is merely a property ID and has no utility in model building. 'Dayhours' capturing the Property Sold date has only 2 values for Year Sold 2014 and 2015, meaning that properties were sold in year 2014 and 2015, with majority sold in year 2014.

Analysis of Price variable -Price is the target variable. Let's plot this variable and analyze the variable

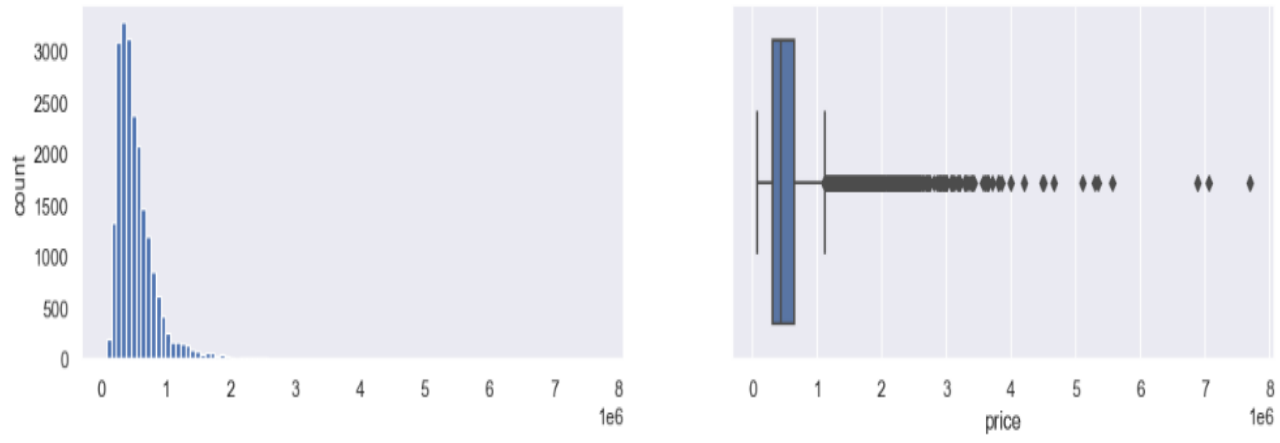


Figure 1 Distribution of Price Variable

Price Variable has a right skewed distribution with skewness of 4.02 and Kurtosis of 34.5 and from boxplot we see that there are many outliers in this variable. It has a mean of $540188.16 and standard deviation of 367362.23. Minimum Price is $75000 and Maximum value is $7700000. Median Price is $450000. Majority of the properties are in the range $30000-$50000 followed by $50000-$80000. Least number of properties are between $50000-$100000 and above $ 500000.

Analysis of Continuous Numerical variables

Click on Hyperlink on each Feature to See the detailed Visual

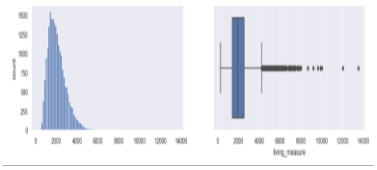
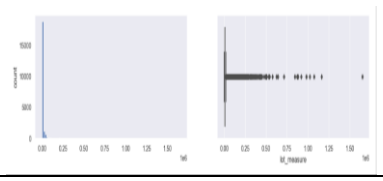
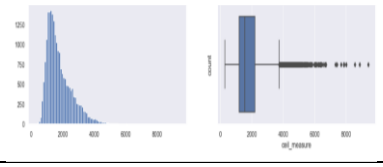
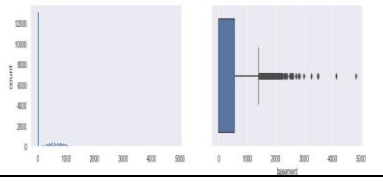
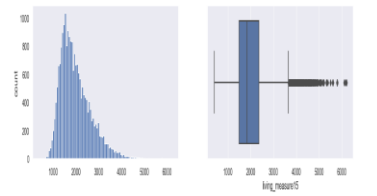
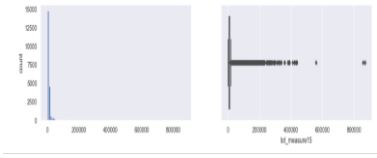
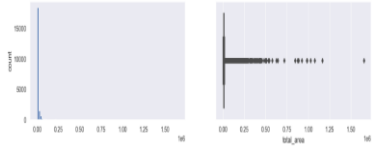
Feature	Distribution	Insight
Living Measure		Distribution is right skewed with skewness of 1.47 and Kurtosis of 5.24 with many outliers in this variable. It has a mean of 2079.90 sq ft and standard deviation of 918.44 sq ft. Minimum is 290 sq ft and Maximum value is 13540 sq ft. Median is 1910 sq ft.
Lot Measure		Distribution is right skewed with skewness of 13.07 with many outliers in this variable. It has a mean of 15106.97 sq ft and standard deviation of 41420.51 sq ft. Minimum is 520 sq ft and Maximum value is 1651359 sq ft. Median is 7618 sq ft.
Ceil Measure		Distribution is right skewed with skewness of 1.45 and Kurtosis of 3.4 with outliers in this variable. It has a mean of 1788.37 sq ft and standard deviation of 828.1 sq ft. Minimum is 290 sq ft and Maximum value is 9410 sq ft. Median is 1560 sq ft.
Basement		Distribution is right skewed with skewness of 1.58 with outliers in this variable. Kurtosis is not available because there are 0 values in dataset It has a mean of 291.52 sq ft and standard deviation of 442.58 sq ft. Minimum is 0 sq ft and Maximum value is 4820 sq ft. Median is 0 sq ft meaning 50% of the properties have basement measure 0.
Living Measure 15		Distribution is right skewed with skewness of 1.11 and Kurtosis of 1.62 and many outliers in this variable. It has a mean of 1987.07 sq ft and standard deviation of 685.52 sq ft. Minimum is 399 sq ft and Maximum value is 6210 sq ft. Median is 1840 sq ft meaning 50% of the properties have Living measure15 value below 1840sqft.
Lot Measure 15		Distribution is right skewed with skewness of 9.52 and Kurtosis of 150 with outliers in this variable. It has a mean of 12766.54 sq ft and standard deviation of 27286.99 sq ft. Minimum is 651 sq ft and Maximum value is 871200 sq ft. Median is 7620 sq ft meaning 50% of the properties have value below 7620 sqft.
Total area		Distribution is right skewed with skewness of 12.96 and Kurtosis of 281.15 with many It has a mean of 17186.87 sq ft and standard deviation of 41589.08 sq ft. Minimum Total_area is 1423 sq ft and Maximum Total_area value is 1652659 sq ft. Median Total_area is 9575 sq ft meaning 50% of the properties have Total_area value below 9575 sqft.

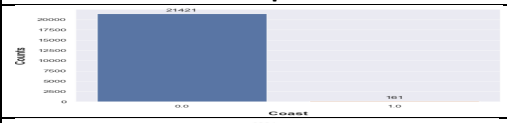
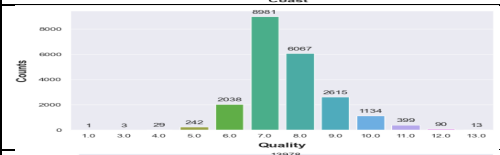
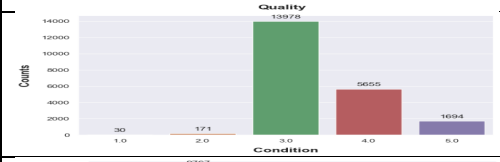
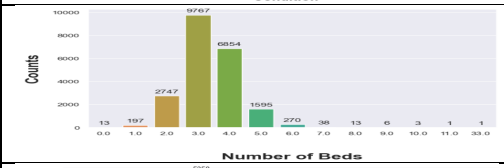
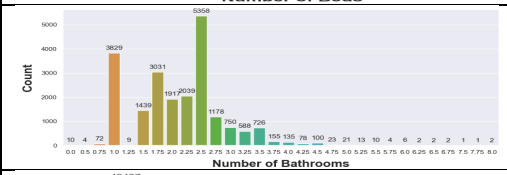
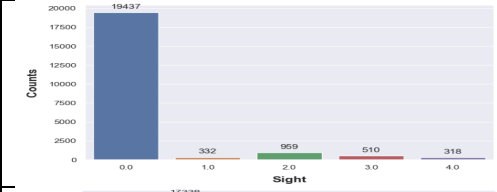
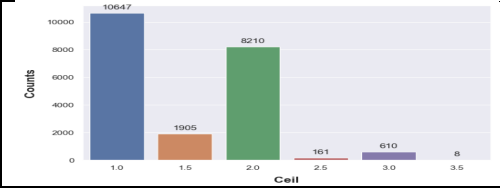
Figure 2 Summary of all Continuous Numerical Variable

Analysis of Temporal variables

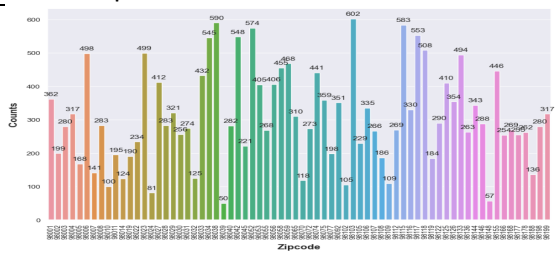
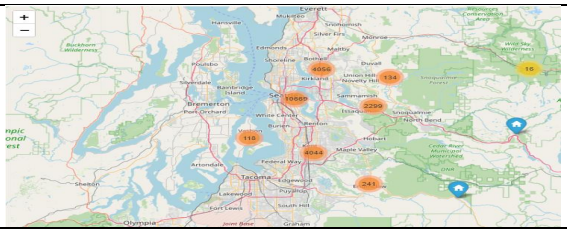
Properties have been built between 1900 and 2015. Majority of properties have Year Renovated as 0, meaning there has been no renovation. Renovations that have happened are between 1934 and 2015.

Analysis of categorical variables

Click on Hyperlink on each Feature to See the detailed Visual

Feature	Count plot	Insight
Coast		99.3 % of the properties have no waterfront view.
Quality		Quality Ranges from 1 to 13 .42 % of the properties have quality 7 and 28 % with quality 8. Only 1 property is of least quality 1 and there are 13 properties which are of best quality rating 13
Condition		Condition ranges from 1 to 5.65 % of the properties have condition 3 and 26 % with condition 4. Only 7.9% of the properties are in best condition i.e. 5
Bedrooms		Bedrooms ranges from 0 to 33 . 77% of the properties have 3 or 4 bedrooms
Bathrooms		Number of Bathrooms range from 1 to 8. 25 % of the properties have 2.5 bathrooms
Sight		90.2 % of the properties have no views and
Furnished		80% of the properties are not furnished.
Ceil		Ceil ranges from 1 to 3.5 .87% of the properties have 1 or 2 Floors .

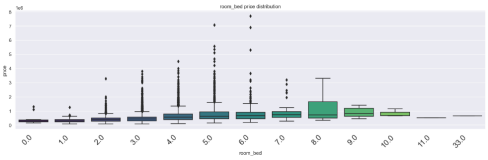
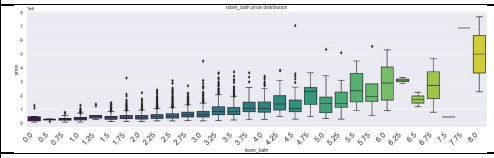
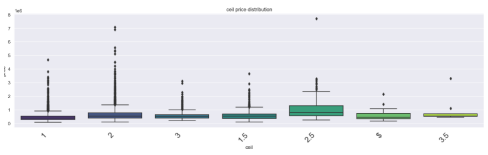
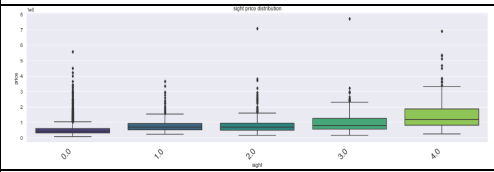
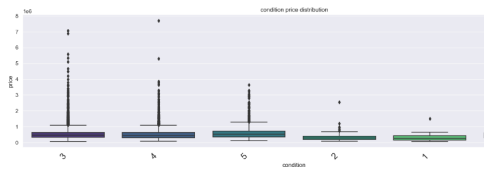
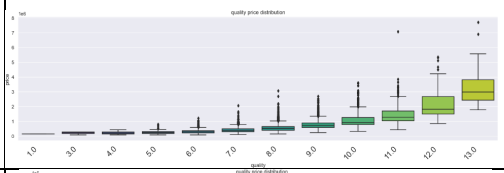
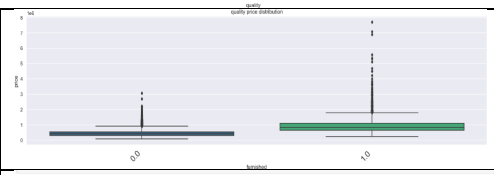
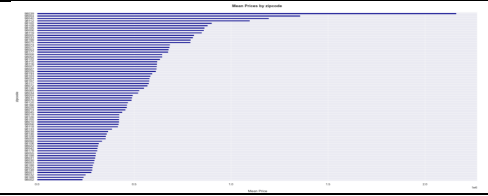
Analysis of Geographical variables

Feature	Visual Representation	Insights
Zipcode		Maximum number of the properties listed are from zip code 98103 which amount to 41.5 % properties and least number of properties are listed from zipcode 98039
Lattitude Longitude		All properties are in Kings County of US state of Washington

Bivariate Analysis

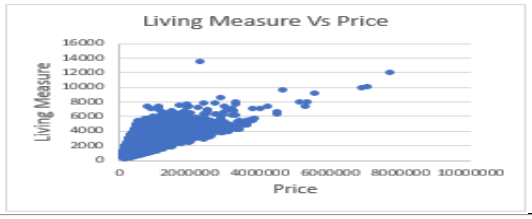
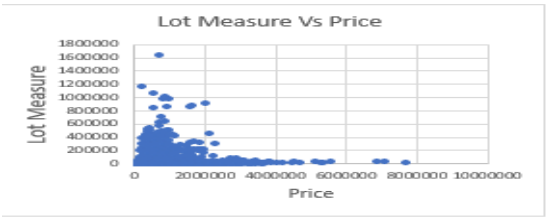
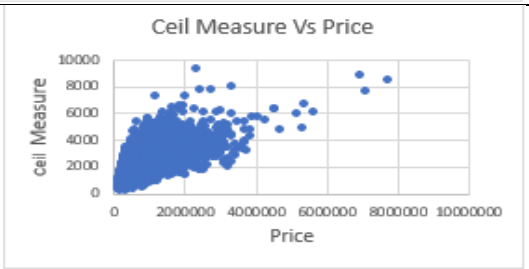
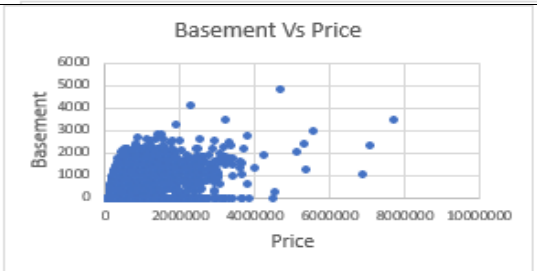
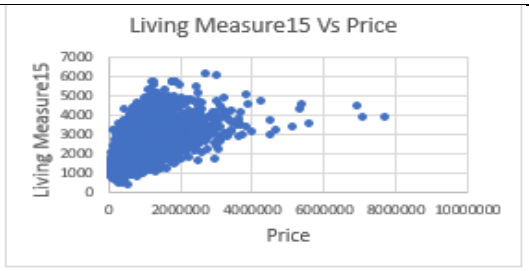
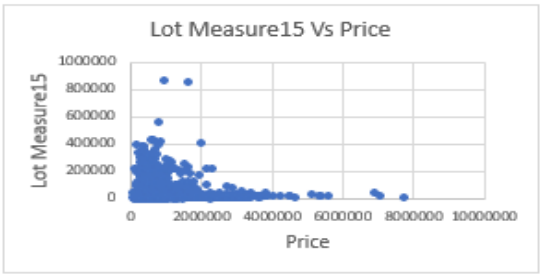
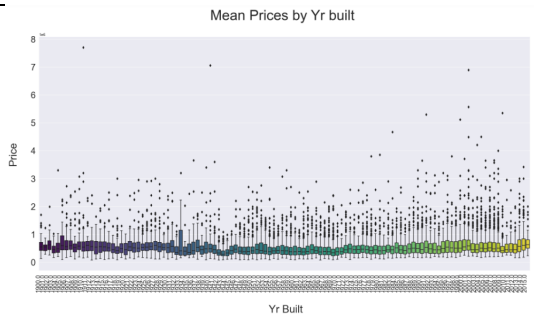
Relationship between the Price and categorical variables

Click on Hyperlink on each Feature to See the detailed Visual

Relationship	Visual Representation	Insights
Bedrooms VS Price		The number of bedrooms does not exhibit a linear relationship with price. As the number of bedrooms increases, there is no corresponding increase in mean prices.
Bathrooms VS Price		The number of bathrooms shows a clear linear relationship with price. As the number of bathrooms increases, the mean prices also show an upward trend.
Number of Floors Vs Price		The variable "Ceil" (number of floors) does not demonstrate a linear relationship with price. Therefore, it is not conclusive to assert that an increase in the number of floors leads to higher prices
Cast Vs Price		Properties with a waterfront view command higher prices compared to those without such a view.
Sight Vs Price		There is a noticeable increase in property prices with an increase in the number of views (sight). Properties with no views are priced considerably lower than those with two or more views.
Condition Vs Price		The overall condition of the property is positively correlated with its price, indicating that properties in better condition tend to have higher prices.
Quality Vs Price		There is a distinct and positive correlation between the quality (grade given to the housing unit) of the property and its price
Furnished Vs Price		Furnished properties are priced higher than their non-furnished counterparts.
Latitude/Longitude Vs Price		Among the Cities, those with waterfront view boasts the highest property prices.
Zipcode Vs Price		Among the Zipcodes, 98039 boasts the highest mean property prices and zip code 98002 which has lowest mean property prices

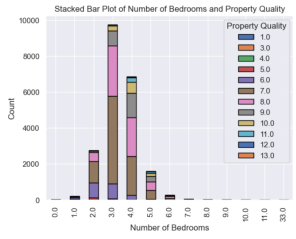
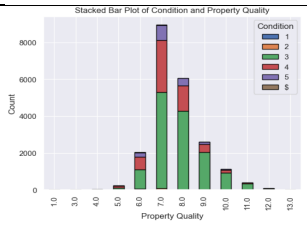
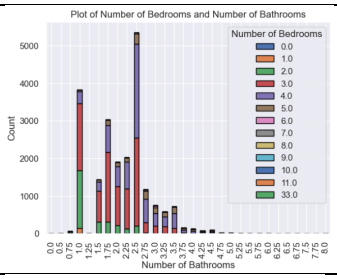
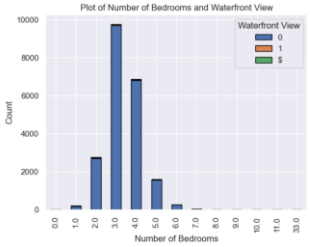
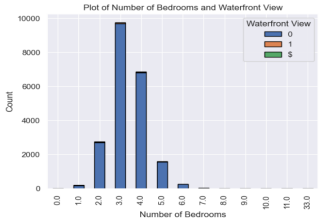
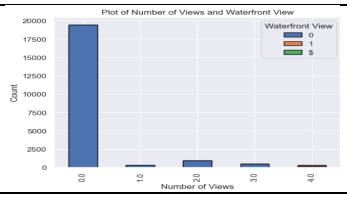
Relationship between Price and all the numerical variables

Click on Hyperlink on each Feature to See the detailed Visual

Relationship	Visual Representation	Insights
Price Vs Living Measure		A clear linear relationship exists between the price of a property and the living area's square footage.
Price Vs Lot Measure		The relationship between property price and lot measure is non-linear, as depicted in the scatter plot.
Price Vs Ceil Measure		A linear relationship is observed between the Ceil measure in a property and its price..
Price Vs Basement Measure		The scatter plot between property prices and basement measure does not reveal a distinct trend.
Price Vs Living Measure 15		The scatter plot reveals a linear relationship between property prices and living measure as of 2015.
Price Vs Lot Measure 15		There is no clear trend in the scatter plot between property prices and lot measure as of 2015.
Price Vs Year Built		No clear linear relationship is observed between the price of a property and the year it was built. This implies that the age of the property, as indicated by the year built, may not be a primary driver of price variation.

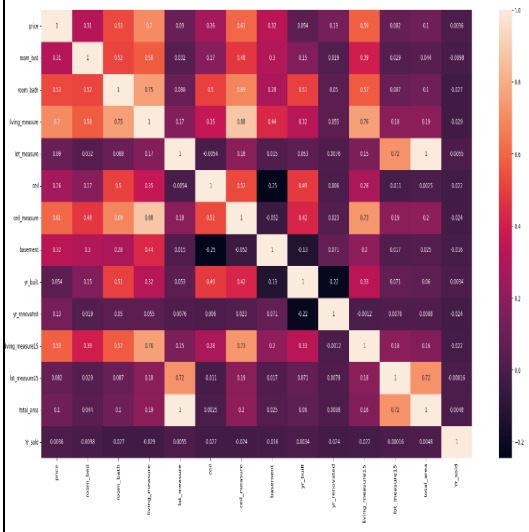
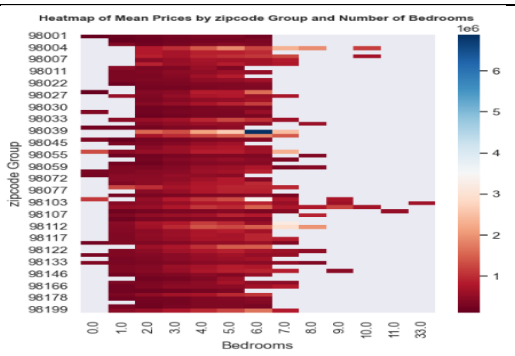
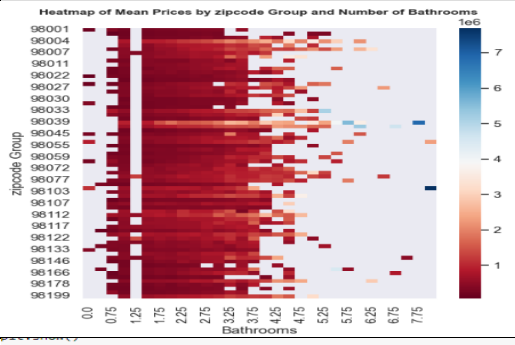
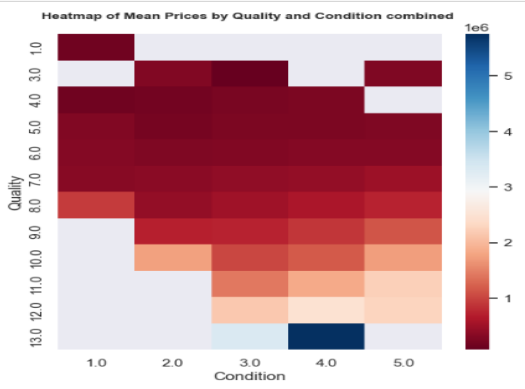
Relationship between the categorical variables

Click on Hyperlink on each Feature to See the detailed Visual

Relationship	Visual Representation	Insights
Bedrooms Vs Quality		3 or 4 bedrooms correlate with higher quality; extreme counts differ.
Quality VS Condition		Higher quality often pairs with better conditions; correlation observed.
Bedrooms Vs Bathrooms		4-bed, 2.5-bath most common; counts decrease with higher bedrooms or bathrooms.
Bedrooms Vs Furnished		Furnished less common; 3 or 4 bedrooms prevail, decreasing with more.
Bedrooms Vs Coast		3-bedroom common across views; decreasing counts with more bedrooms observed.
Bedrooms Vs Coast		3-bedroom prevalent, decreasing with more; waterfront views less common.
Sight Vs Coast		No waterfront has higher "Number_of_Views" counts overall.

Multi-variate Analysis:

We can investigate interactions and dependencies among multiple variables by utilizing heatmaps to display correlation matrices for multiple variables. Let's see the Correlation Plot between Numerical variables
Click on Hyperlink on each Feature to See the detailed Visual

Relationship	Visual Representation(Heatmap)	Insights
Heatmap of All Numerical Variables		Property price strongly correlates with living measure, room bath, ceil measure, and living measure 15. Bedrooms show strong correlation with room bath, ceil measure; room bath correlates with living measure. Ceil exhibits moderate correlation with ceil measure; lot measure aligns with living measure 15
Zipcode Vs Price Vs Bedrooms		There is no marked increase in prices in any zip code with increase in Bedrooms.
Zipcode Vs Price Vs Bathrooms		There is clear increase in prices with increase in bathrooms in some zip codes
Quality Vs Condition Vs Price		Highest Prices are seen in properties with Highest quality and property Conditions and lowest prices are seen in properties with low quality irrespective of the condition

EDA using Clustering- Clustering was done to identify subgroups within the data that may have different characteristics. Understanding these subgroups can guide feature selection and model building in the supervised learning phase. Assigning cluster labels to the data points can create a new categorical feature. This can be used as an additional feature for training a supervised model, providing information about the inherent structure of the data. For Detailed Clustering Analysis refer to Appendix G

Cluster	price	living_measure	lot_measure	quality	room_bath	room_bed	ceil	condition	years old
0	419399.24	1676.66	7672.14	7.14	1.85	3.15	1.36	3.47	49.95
1	864873.85	3083.41	11482.38	9.04	2.77	3.92	1.85	3.25	25.48

Table 2 Clusters

- Cluster 0, with a mean price of $419,399.24, represents the lowest-priced properties. This cluster is characterized by an average quality of 7, smaller living and lot measures, and older properties with an average age of 50 years.
- Cluster 1 represents high-end properties with an average price of $864,873.85, featuring spacious living areas, larger lots, and superior quality, likely reflecting recent renovations and newer constructions with an average age of 25 years.

How EDA analysis is impacting the business

Here are some of the ways in which EDA analysis impacts the Business decisions

1.EDA helped in understanding the structure, distribution, and quality of the dataset.

2. Insights gained during EDA influenced the creation of new features or the transformation of existing ones to improve the predictive power of the model.

3. EDA helped identify the most influential features that contribute to house prices and business significance of these features can guide strategic decisions.

4. Outliers identified during EDA which when handled can Increase the model's robustness and reliability.

5. Insights into relationships between different variables obtained through EDA. Strong correlations or patterns can inform business decisions related to property valuation, investment, or marketing.

6. Visualizations and charts generated during EDA can communicate complex insights to non-technical stakeholders. Visualizations facilitate better understanding and decision-making for business users.

7.Risks or challenges identified during EDA such as Incomplete or inaccurate data, unusual data points (outliers), High correlations between features may lead to biased or unreliable insights with multicollinearity issues, impacting the interpretability of the model and model's performance. We will be Implementing thorough data cleaning and preprocessing steps to treat missing values, anomalies and outliers

8.Insights from EDA, especially feature importance, can be leveraged for business strategy. Business can benefit from understanding which features significantly influence house prices.

Both visual and non-visual EDA

Visual Understanding:

We did Exploratory Data Analysis (EDA) using Visualizations (scatter plots, box plots, histograms) to grasp the distribution and patterns in the data. We created geographical plots to visualize the spatial distribution of house prices

Non-Visual Understanding:

Statistical Metrics: We calculate summary statistics (mean, median, standard deviation) for numerical variables and assessed skewness and kurtosis to understand the shape of distributions.

Descriptive Analysis: We described the key insights and trends by providing context and interpretation for statistical measures.

Data Cleaning and Pre-processing

Missing Value Treatment

Treating missing and junk values in a regression problem is crucial step in data preprocessing as they can distort statistical measures, violate model assumptions, and compromise model accuracy, leading to biased parameter estimates and reduced interpretability. Addressing missing and junk values ensures data quality, improves model robustness, and contributes to the reliability and validity of regression analyses.
Before proceeding with missing value treatment, using functions in pandas we identify missing and Junk values and visualize missing values using heatmaps.

Room Bed	Room Bath	Living measure	Lot measure	Ceil measure	Living measure15	Lot measure15	Total area	Basement	Yr Built	Furnished	Ceil	Coast	Sight	Condition	Quality
108	108	17	42	1	166	29	29	1	1	29	42	1	57	57	1

Table 3 Count of Missing Values

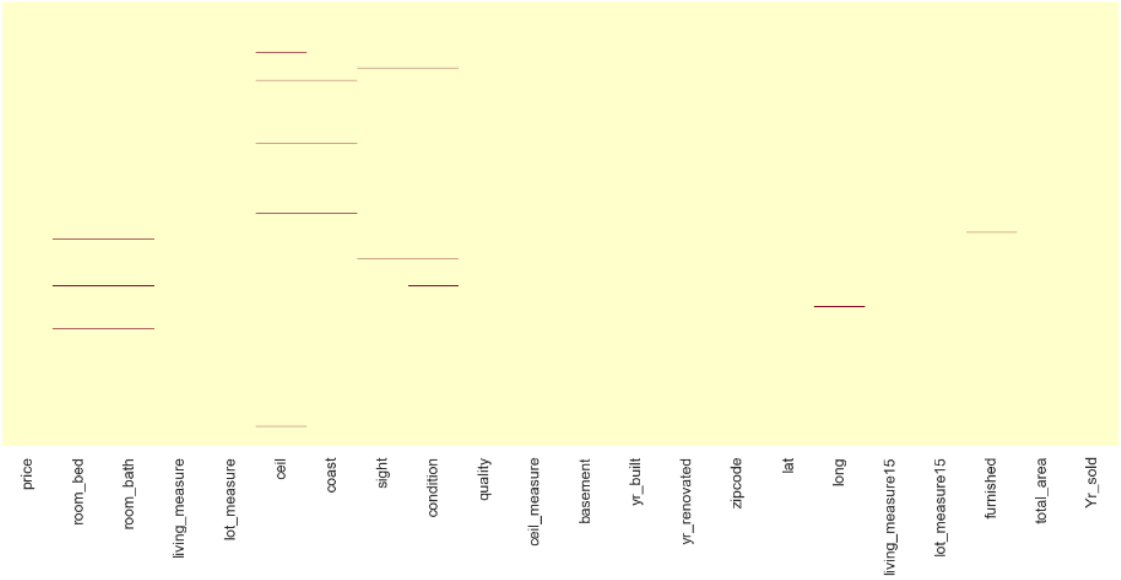


Figure 3 Visual representation of Missing values in dataset

As we can see the red lines in plot indicate the missing values. 0.14 % of the dataset has missing values.
Let's see the count of junk value "$" in variables in our dataset:

Ceil	Coast	Condition	Yr Built	Long	Total area
30	30	28	14	34	39

Table 4 Count of Junk Values

0.04 % of the dataset has Junk values. It's not a very high percentage.

Incomplete data can bias the results of the machine learning models and/or reduce the accuracy of the model and hence need to treated correctly

Let’s see the various imputation techniques used on our dataset to treat missing values and junk values

- The "Total_area" variable, representing the sum of "Living_measure" and "Lot_measure," contains null values and junk values (e.g., $). Initially, the junk values are converted to null, and subsequently, the null values are imputed with the sum of "Living_measure" and "Lot_measure."
- Addressing missing values in "Living_measure" involves imputing them with the difference between "Lot_measure" and "Total_area." Conversely, missing values in "Lot_measure" are imputed with the difference between "Living_measure" and "Total_area."
- Null values in "Living_measure15," indicating the living room area in 2015, are imputed with the values from "Living_measure," signifying no changes in the living measure in 2015.
- Similarly, null values in "Lot_measure15," representing lot size area in 2015, are imputed with values from "Lot_measure," indicating no changes in lot measure in 2015.
- For variables "Furnished," "Sight," "Condition," and "Quality," null values are imputed with the most frequently occurring values (mode) within each respective column.
- Null values in "Furnished" are specifically imputed with 0.
- Junk values represented by '$' in the "Condition" variable are converted to null values, and along with other null values, they are imputed with the mode, which is 3.
- Null values in "Quality" are imputed with 7, and null values in "Sight" are imputed with 0.
- The "Coast" variable has junk values represented by '$,' which are first converted to null and they are imputed with the mode, which is 0.
- The "Ceil" variable has junk values represented by '$,' which are first converted to null and they are imputed with the mode, which is 1.
- The " Room_bed " are imputed with the mode, which is 3.
- The " Room_bath " are imputed with the mode, which is 2.5.
- Variables "Ceil_measure,”,” Years_old” and "long" are imputed using the KNN Imputer.
- "Yr_built" has one null value and 14 junk values represented by '$,' which are converted to null values. "Yr_built" and "Yr_sold" will be used to create a new variable, "Years_old," to calculate the age of the property. 15 Missing values in "Years_old" will then be imputed using the KNN Imputer.

Outlier treatment

Linear regression is highly sensitive to outliers as it minimizes the sum of squared differences, and regularization in ridge and lasso regression only partially mitigates their impact. Support Vector Regression (SVR) is prone to outliers, influencing hyperplane creation, and while decision trees are less sensitive, random forests, as an ensemble method, can be affected by extreme values.
Outliers in a regression problem should be treated as they can distort statistical measures, violate model assumptions, and negatively impact model performance, leading to biased parameter estimates and reduced predictive accuracy. Addressing outliers enhances model robustness, improves interpretability, and ensures the integrity of regression analyses by mitigating the influence of extreme values on decision boundaries and data patterns.
To detect the outliers, we can check the boxplot of the variables . Boxplots graphically represent the distribution of the data and provide a lot of statistical information, including — medians, ranges, and outliers. When reviewing a box plot, an outlier is defined as a data point that is located outside the whiskers of the box plot. For example, outside 1.5 times the interquartile range above the upper quartile and below the lower quartile (Q1 – 1.5 * IQR or Q3 + 1.5 * IQR). Here is the Boxplot of Numerical Variables

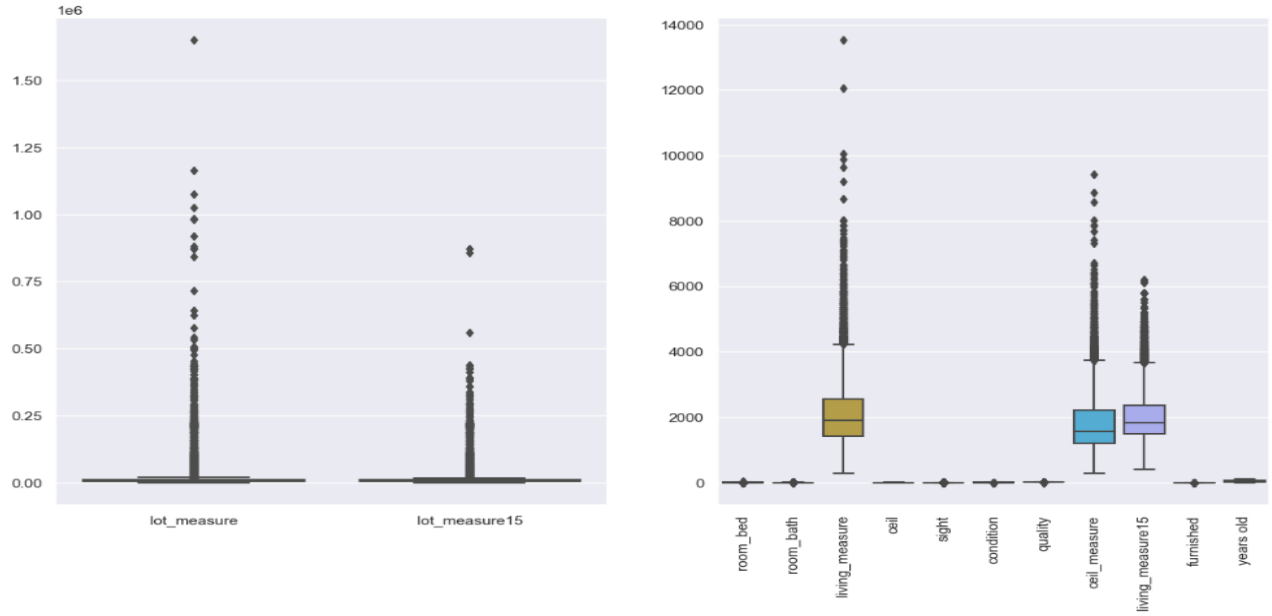


Figure 4 Outliers in Numerical variables

There are outliers in all variables except years old. Extreme and heavy outliers can be seen in variables like Lot Measure, Lot measure15,Living Measure, Ceil Measure and Living Measure15.

We will use the Quantile Based Flooring and Capping method to remove the outliers in the dataset .Values above Q3+1.5*IQR will be capped to Q3+1.5*IQR and values less than Q1-1.5*IQR will be floored to Q1-1.5*IQR

Let's see the boxplot after removing the outliers in the variables

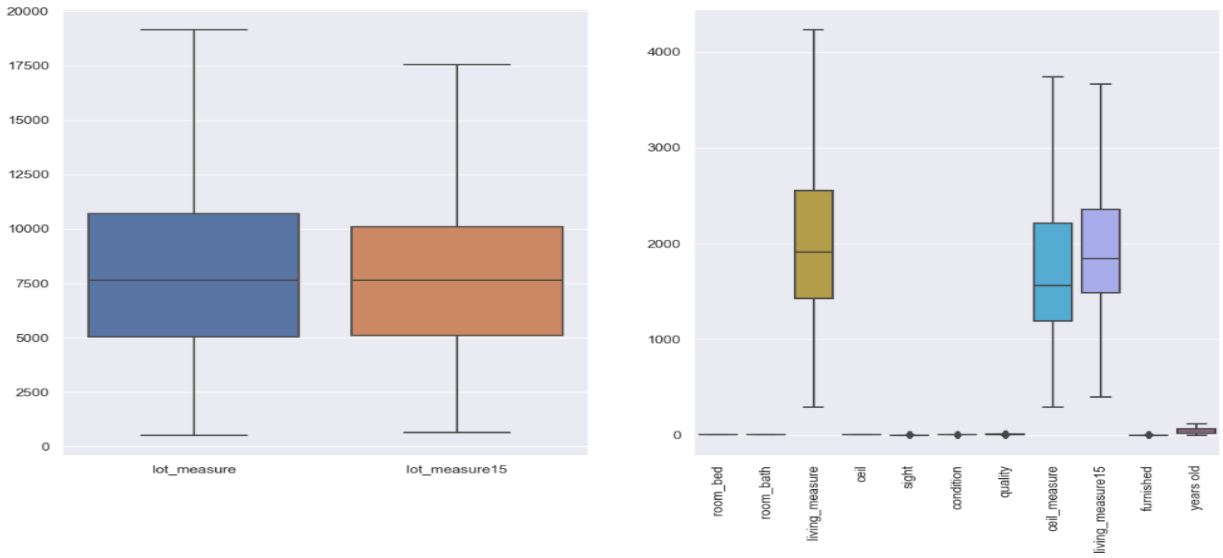


Figure 5 After outlier treatment

After the outlier treatment the variables are now treated and as we can see there are no outliers in the boxplot

Variable transformation

Variable Transformation helps in transforming raw data into a format that enhances a model's ability to make accurate predictions. We have done different variable transformations by creating new features out of existing features or modifying existing ones to improve the performance of a model. Here are some techniques we have used in variable transformation:

- **Variable Transformation for data Balancing:**
  We apply log transformation on the target variable 'Price', because the price variable is not normally distributed. This Imbalance can lead to problems in model building because if the variable is skewed the model better predicts the price values around the mean, and not very good at predicting the prices values from the tails. For details on Log Transformation of Price Variable refer to Appendix F
- **Imputation:** Handling missing values by imputing them with appropriate values, such as mean, median, or using more advanced imputation techniques like KNN. As we saw in previous section, we imputed the missing values for some of the variables
- **Creating Interaction Terms:** We created a new feature Years_old using two variables Yr_built and Yr_sold.
- **Encoding Categorical Variables:** We transformed Variable 'Zipcode' using one hot encoding and for each Zipcode' a new variable has been added 'zipcode_98002' up to 'zipcode_98199 are created.
- **Creating Time-Based Features:** Variable "dayhours" was transformed to Year_sold by extracting the year portion from this variable
- **Creating Domain-Specific Features:** We created new variables "Basement Present" and" Is renovated" to establish presence or absence of a certain feature of the property
- **Scaling:** Scaling numerical features to a similar range to ensure that no particular feature dominates the learning process. We will scale the train and test dataset before model building

Removal of unwanted variables:

- CID is acting solely as a property ID without providing meaningful insights. Given its limited utility, this variable can be safely removed.
- The variable "Dayhours" indicating the property's sale date, proves to be of limited use in data analysis, aside from determining the property's age at the time of sale. Consequently, it is recommended for removal. Instead, a new variable containing only the extracted year component will be created.
- New Variable "Price Range" was created to visualize the price variable. Now that we have analyzed we can drop it
- 96% of records show "Yr_renovated" as 0, making it less useful for model building. A more meaningful approach involves creating a new column "Is_renovated" to ascertain whether the property underwent renovation, providing a more suitable feature for model building. Given the redundancy of Yr_renovated, we can omit it from the dataset
- 60% of records show "Basement " as 0, making it less useful for model building. A better approach involves creating a new column "Basement_present" to ascertain whether the property has a basement feature, providing a more suitable feature for model building. Given the redundancy of 'Basement', we can drop it from the dataset
- The variable "Total_area" redundantly combines two variables, "Living_measure" and "Lot_measure." Given this redundancy, we can omit "Total_area" from the dataset.
- The variable Zip code is redundant after we have encoded it. So, we can drop this variable

Addition of New Variables:

Several new variables have been introduced into our machine learning dataset, namely "is_renovated”,"Basement Present”, "years_old," "year_sold" and all the encoded zipcodes

- **Creation of "Year_sold":**
  The variable "year_sold" is a derivative of the "dayhours" variable, where the year is extracted to represent the year in which the property was sold.
- **Introduction of Binary Variable "Is_renovated":**
  A binary variable, "is_renovated," has been incorporated based on the "yr_renovated" variable. It is assigned a value of 0 if "yr_renovated" is 0 and 1 if any other non-zero value is present.
- **Introduction of Binary Variable "Basement_present ":**
  A binary variable, " Basement_present" has been incorporated based on the "Basement " variable. It is assigned a value of 0 if " Basement " is 0 and 1 if any other non-zero value is present.
- **Generation of "Years_old":**
  The new variable "years_old" has been constructed by calculating the difference between "year_sold" and "yr_built," providing the age of the property at the time of sale.
- **Generation of Label Encoded Zipcodes:**
- Converting categorical variables into numerical representations using encoding. We transformed Variable ‘Zipcode’ using one hot encoding and for each Zipcode’ a new variable has been added 'zipcode_98002', 'zipcode_98003', 'zipcode_98004','zipcode_98005', 'zipcode_98006', 'zipcode_98007', 'zipcode_98008','zipcode_98010', 'zipcode_98011', 'zipcode_98014', 'zipcode_98019','zipcode_98022', 'zipcode_98023', 'zipcode_98024', 'zipcode_98027', 'zipcode_98028', 'zipcode_98029', 'zipcode_98030', 'zipcode_98031', 'zipcode_98033', 'zipcode_98034', 'zipcode_98038', 'zipcode_98039', 'zipcode_98040', 'zipcode_98042', 'zipcode_98045', 'zipcode_98052', 'zipcode_98053', 'zipcode_98055', 'zipcode_98056', 'zipcode_98058', 'zipcode_98059', 'zipcode_98065', 'zipcode_98070', 'zipcode_98072', 'zipcode_98074', 'zipcode_98075', 'zipcode_98077', 'zipcode_98092','zipcode_98102', 'zipcode_98103', 'zipcode_98105', 'zipcode_98106', 'zipcode_98107', 'zipcode_98108', 'zipcode_98109', 'zipcode_98112','zipcode_98115', 'zipcode_98116', 'zipcode_98117', 'zipcode_98118', 'zipcode_98119', 'zipcode_98122', 'zipcode_98125', 'zipcode_98126', 'zipcode_98133', 'zipcode_98136', 'zipcode_98144', 'zipcode_98146','zipcode_98148', 'zipcode_98155', 'zipcode_98166', 'zipcode_98168','zipcode_98177', 'zipcode_98178', 'zipcode_98188', 'zipcode_98198','zipcode_98199 are created

Model building

After data preprocessing next step is model building . We have the dataset with 88 columns which is now ready for building a regression model

Train Test Split
For our analysis we split the dataset into train and test with the ratio of 70:30.
After splitting the dataset with 21613 rows, the shape of train dataset is 15129 rows and 88 columns, the shape of test dataset is 6484 rows and 88 columns. Price column is the target variable.

Note: **The Target Variable Price is log transformed**

Before Modelling we have to scale the train and test dataset. There are features in dataset that vary in scale and distance-based ML algorithms like KNN fail to give a reasonable recognition to the smaller feature .We should apply feature scaling after the split, so that we prevent information leakage on the test set
We will use Standard Scalar method on our data.

After splitting the data and scaling, we can proceed with model building.
Here we are predicting continuous variable price and we have to show the relationship of house price with other features in the dataset, so we f i r s t choose to build the linear regression model for the prediction.

We will build other models like KNN, Support Vectors, Decision Tree, Random Forest, Ridge and Lasso and see which ones perform well on both train and test data.
In the context of Regression, (R-squared), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), Mean absolute Percentage Error (MAPE) are commonly used metrics to assess the performance of a regression model. For details of each of these Metrics refer to Appendix G

Linear Regression Models

We will build both statistical models (using the Ordinary Least Squares (OLS) method) and machine learning linear regression models to establish a linear relationship between dependent variable Price and independent variables.

Linear regression model (using OLS)

Check Appendix C to see the OLS model building process and for Residual Analysis

The performance metrics of the Final OLS regression model on Train and Test data are as below
With OLS model we get an R-squared value of 0.86 on train data meaning the model is able to explain 86 % variability in the data and RMSE for training data is 0.1975 .The OLS model demonstrates a good fit with an R-squared value of 0.855 on test data , signifying its ability to explain 85% of the variability in the price, and a testing RMSE of 0.199. Both the R-squared values for testing and training sets (0.86 and 0.85) indicate a robust model capturing a significant portion of the variance, while a slightly higher RMSE on the testing set (0.199) suggests consistent but marginally increased prediction errors on unseen data.

Here are top 10 rows of the test data with actual values, predicted values and their residuals. Note that Price has been log transformed

	Actual Values	Fitted Values	Residuals
0	13.17	12.93	0.25
1	12.68	13.11	-0.44
2	12.69	12.44	0.25
3	13.15	13.16	-0.01
4	12.32	12.42	-0.09
5	14.39	14.03	0.37
6	14.22	13.88	0.34
7	12.91	13.04	-0.13
8	12.49	12.37	0.12
9	12.82	12.70	0.12

Table 5 Actual VS Predicted Values

We can build a linear regression equation using the coefficient and predictors

```
price= 13.048161214389463 + 0.03088232900719387 * ( room_bed ) +  0.07821892598107728 * ( room_bath ) +  0.046313186377569204 *
( lot_measure ) +  0.010949903496800535 * ( ceil ) +  0.040533201399220536 * ( coast ) +  0.04502612191308884 * ( sight ) +  0.0
328959420917967 * ( condition ) +  0.15295131795429207 * ( quality ) +  0.10049307879300473 * ( living_measure15 ) +  0.0120197
42155220461 * ( furnished ) +  0.031001224119803772 * ( years old ) +  0.015228249547542993 * ( is_renovated ) +  0.00778837231
4868848 * ( basement_present ) +  0.1303719209457837 * ( zipcode_98004 ) +  0.05656317241059215 * ( zipcode_98005 ) +  0.087477
94672174236 * ( zipcode_98006 ) +  0.049803239717505665 * ( zipcode_98007 ) +  0.07299230469493362 * ( zipcode_98008 ) +  0.017
859591761096616 * ( zipcode_98010 ) +  0.04105457787233378 * ( zipcode_98011 ) +  0.02606153022014889 * ( zipcode_98014 ) +  0.03
150434276833695 4 * ( zipcode_98019 ) +  0.005942732611520795 * ( zipcode_98022 ) +  -0.005425618352092501 * ( zipcode_98023 ) +
0.030895492606455292 * ( zipcode_98024 ) +  0.06515237228846592 * ( zipcode_98027 ) +  0.046838570653137734 * ( zipcode_98028 )
+  0.07027867977662716 * ( zipcode_98029 ) +  0.006997748275084245 * ( zipcode_98030 ) +  0.009314484399149656 * ( zipcode_9803
1 ) +  0.10936636570882782 * ( zipcode_98033 ) +  0.08407622057232128 * ( zipcode_98034 ) +  0.02907547929310913 * ( zipcode_98
038 ) +  0.06381160661225152 * ( zipcode_98039 ) +  0.09301405745137059 * ( zipcode_98040 ) +  0.011803771557085 5 * ( zipcode_
98042 ) +  0.035134139666291614 * ( zipcode_98045 ) +  0.09970719013926535 * ( zipcode_98052 ) +  0.08413153402961018 * ( zipco
de_98053 ) +  0.016155222511837 11 * ( zipcode_98055 ) +  0.045295958089288 84 * ( zipcode_98056 ) +  0.024776798221594648 * ( zip
code_98058 ) +  0.05055515781663199 85 * ( zipcode_98059 ) +  0.052478550714326 88 * ( zipcode_98065 ) +  0.025260981980269465 * (
zipcode_98070 ) +  0.05065645535978095 * ( zipcode_98072 ) +  0.07555832301972173 * ( zipcode_98074 ) +  0.06950988812450762 *
( zipcode_98075 ) +  0.04150745809003581 * ( zipcode_98077 ) +  0.003660397880622 8066 * ( zipcode_98092 ) +  0.0630414914025229
9 * ( zipcode_98102 ) +  0.12988766203351948 * ( zipcode_98103 ) +  0.09273793844441897 * ( zipcode_98105 ) +  0.04198074654755
714 * ( zipcode_98106 ) +  0.08923623295386961 * ( zipcode_98107 ) +  0.03303521513917003 * ( zipcode_98108 ) +  0.067314984348
08321 * ( zipcode_98109 ) +  0.11199571697601973 * ( zipcode_98112 ) +  0.13305760678291 66 * ( zipcode_98115 ) +  0.09296955174
613508 * ( zipcode_98116 ) +  0.12892419341564798 * ( zipcode_98117 ) +  0.07054950244462979 * ( zipcode_98118 ) +  0.087033428
9961152 * ( zipcode_98119 ) +  0.09111022094624361 * ( zipcode_98122 ) +  0.07732106003843316 * ( zipcode_98125 ) +  0.071802747
02679754 * ( zipcode_98126 ) +  0.07055592422082232 * ( zipcode_98133 ) +  0.07141260905950517 * ( zipcode_98136 ) +  0.0807950
7487869961 * ( zipcode_98144 ) +  0.031667716314745534 * ( zipcode_98146 ) +  0.008620266066822892 * ( zipcode_98148 ) +  0.059
89603320215681 * ( zipcode_98155 ) +  0.033345750274105516 * ( zipcode_98166 ) +  0.009529614119540179 * ( zipcode_98168 ) +
0.062206213453875434 * ( zipcode_98177 ) +  0.01559455324818701 * ( zipcode_98178 ) +  0.007625542000277513 * ( zipcode_98188 )
+  0.008015079626683325 * ( zipcode_98198 ) +  0.10221795611138064 * ( zipcode_98199 )
```

Figure 6 Linear Regression Equation OLS Method

The provided linear regression equation describes the linear relationship between the house price and various predictor variables. Here's how we can interpret the coefficients for each variable:

Intercept: The intercept term is 13.05 This represents the estimated log of house price when all other predictors are zero

Coefficients for Predictor Variables:

For each continuous variable (e.g., room_bed, room_bath, lot_measure, ceil, condition, quality, basement, years old), the coefficient represents the estimated change in log of house price for a one-unit increase in that variable while holding all other variables constant. For categorical variables (e.g., zipcode_98002), the coefficients represent the estimated change in log of house price compared to the reference category in this case zipcode_98001.

Linear regression model (Machine Learning)

Performance Metrics using Linear Regression:

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Linear Regression	0.875	0.185	0.0342	0.1366	1.0504	0.8811	0.1822	0.0332	0.1343	1.0316

Table 6 Metrics of Linear Regression

The linear regression model demonstrates good predictive performance with R-squared values of 0.875 for the test set and 0.88 for the train set. So, there is no overfitting. However, the model exhibits substantial errors, as evidenced by high RMSE, MSE,MAE and MAPE values.

With Linear regression model we get the following regression coefficients:

Features	Coefficient Values	Features	Coefficient Values
zipcode_98004	0.1162		
quality	0.1068	condition	0.034
zipcode_98115	0.1015	zipcode_98177	0.0339
zipcode_98103	0.0995	room_bath	0.0337
living_measure	0.0982	zipcode_98072	0.0334
zipcode_98112	0.0971	zipcode_98133	0.0331
zipcode_98117	0.0929	zipcode_98024	0.0311
ceil_measure	0.0914	zipcode_98077	0.0281
zipcode_98033	0.0876	lot_measure	0.0281
zipcode_98040	0.0836	zipcode_98155	0.0251
zipcode_98006	0.0809	zipcode_98106	0.0249
zipcode_98052	0.0809	zipcode_98014	0.0243
zipcode_98122	0.0773	zipcode_98010	0.024
zipcode_98199	0.0772	zipcode_98022	0.0235
zipcode_98105	0.0765	zipcode_98058	0.0232
lat	0.0748	zipcode_98011	0.0226
zipcode_98119	0.0726	zipcode_98028	0.0225
zipcode_98116	0.0724	zipcode_98019	0.0222
zipcode_98029	0.0716	zipcode_98108	0.0222
zipcode_98107	0.0693	zipcode_98166	0.0213
zipcode_98074	0.0675	zipcode_98042	0.0176
zipcode_98053	0.0667	basement_present	0.017
zipcode_98027	0.0652	zipcode_98070	0.0159
zipcode_98075	0.0648	zipcode_98146	0.0156
zipcode_98144	0.0644	is_renovated	0.0143
zipcode_98008	0.0641	yearsold	0.0121
zipcode_98039	0.0566	zipcode_98055	0.0115
zipcode_98109	0.0566	zipcode_98092	0.0103
zipcode_98034	0.0561	zipcode_98031	0.0093
zipcode_98136	0.0558	zipcode_98030	0.0078
zipcode_98065	0.0551	zipcode_98178	0.0066
zipcode_98102	0.0547	zipcode_98148	0.0046
zipcode_98118	0.0529	lot_measure15	0.0031
zipcode_98126	0.0509	zipcode_98002	0.0018
zipcode_98005	0.05	zipcode_98198	0.0017
living_measure15	0.0494	zipcode_98188	0.0015
zipcode_98045	0.0479	zipcode_98003	0.0003
zipcode_98125	0.0472	zipcode_98168	-0.003
sight	0.0462	zipcode_98032	-0.0036
zipcode_98007	0.0437	furnished	-0.0056
zipcode_98059	0.0431	room_bed	-0.006
zipcode_98038	0.0403	zipcode_98023	-0.0096
coast	0.0395	ceil	-0.0148
zipcode_98056	0.0346	long	-0.0594

Table 7 Linear Regression Coefficients

Plot below shows the Features and their Coefficient values

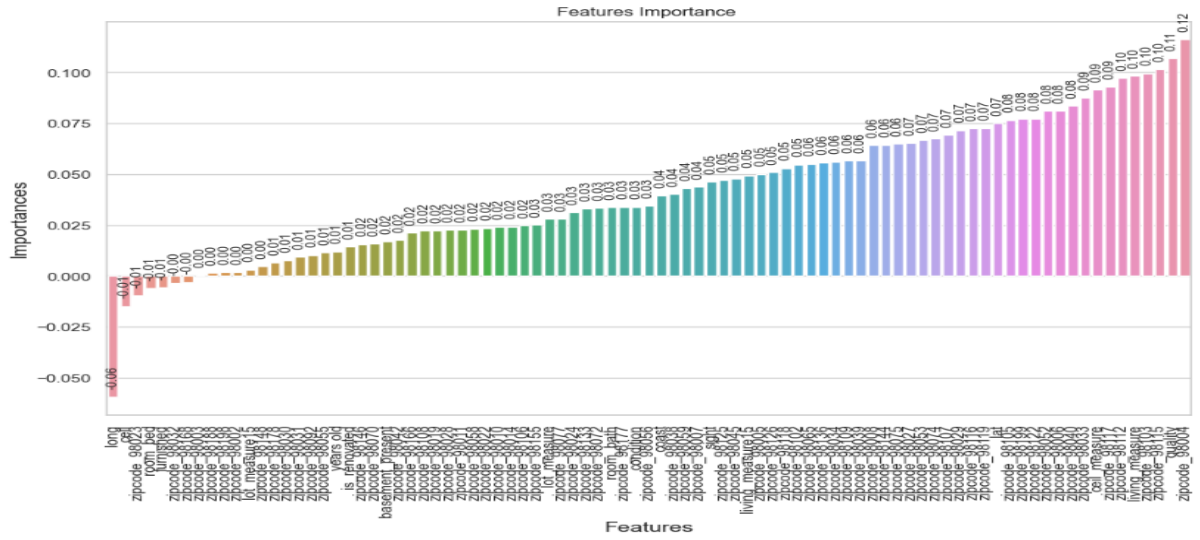


Figure 7 Feature Importance from Linear Regression

As we can see from the plot above features quality, zipcode_98004, zipcode_98115, zipcode_98103, living measure, zipcode_98112, zipcode_98117, ceil measure, lat, living measure 15 are some important features and they have positive impact on the price whereas long, ceil, zipcode 98023, furnished has a negative impact.

Linear Regression equation using the above coefficients

price=13.04+ -0.00495384302 * (room_bed) + 0.0342461865 * (room_bath) + 0.082792551 * (living_measure) + 0.0305467672 * (lot_measure) + -0.0180666161 * (ceil) + 0.0412306729 * (coast) + 0.0465134182 * (sight) + 0.0358913921 * (condition) + 0.114304219 * (quality) + 0.105669131 * (ceil_measure) + 0.0775573492 * (lat) + -0.0502222018 * (long) + 0.0479578888 * (living_measure15) + -0.00822545126 * (furnished) + 0.0124388161 * (years old) + 0.0138946536 * (is_renovated) + 0.0227601422 * (basement_present) + 0.000853317309 * (zipcode_98002) + -0.00146893367 * (zipcode_98003) + 0.112385283 * (zipcode_98004) + 0.0487386212 * (zipcode_98005) + 0.0790709474 * (zipcode_98006) + 0.0420028493 * (zipcode_98007) + 0.0585309124 * (zipcode_98008) + 0.0229030833 * (zipcode_98010) + 0.0202596261 * (zipcode_98011) + 0.0198354706 * (zipcode_98014) + 0.0172930005 * (zipcode_98019) + 0.0215656824 * (zipcode_98022) + -0.0119909788 * (zipcode_98023) + 0.0296971824 * (zipcode_98024) + 0.0640157013 * (zipcode_98027) + 0.0184846078 * (zipcode_98028) + 0.0681108825 * (zipcode_98029) + 0.00522985791 * (zipcode_98030) + 0.00530736948 * (zipcode_98031) + -0.00611031252 * (zipcode_98032) + 0.0813757439 * (zipcode_98033) + 0.050571679 * (zipcode_98034) + 0.0372032926 * (zipcode_98038) + 0.0507450238 * (zipcode_98039) + 0.0795376883 * (zipcode_98040) + 0.0126187587 * (zipcode_98042) + 0.0410766799 * (zipcode_98045) + 0.0743455254 * (zipcode_98052) + 0.0622343555 * (zipcode_98053) + 0.00995777112 * (zipcode_98055) + 0.0315806758 * (zipcode_98056) + 0.0190913483 * (zipcode_98058) + 0.0381993785 * (zipcode_98059) + 0.0502121674 * (zipcode_98065) + 0.0145698078 * (zipcode_98070) + 0.0297869544 * (zipcode_98072) + 0.0615520867 * (zipcode_98074) + 0.0561435508 * (zipcode_98075) + 0.0233937498 * (zipcode_98077) + 0.00785028163 * (zipcode_98092) + 0.0508869886 * (zipcode_98102) + 0.0979645131 * (zipcode_98103) + 0.0754626487 * (zipcode_98105) + 0.0234497174 * (zipcode_98106) + 0.0683573559 * (zipcode_98107) + 0.0205458825 * (zipcode_98108) + 0.0537764219 * (zipcode_98109) + 0.093194963 * (zipcode_98112) + 0.0940004617 * (zipcode_98115) + 0.0667874279 * (zipcode_98116) + 0.0908145901 * (zipcode_98117) + 0.0514248676 * (zipcode_98118) + 0.0701214969 * (zipcode_98119) + 0.0717608613 * (zipcode_98122) + 0.0458661711 * (zipcode_98125) + 0.047824025 * (zipcode_98126) + 0.0290596004 * (zipcode_98133) + 0.0574156163 * (zipcode_98136) + 0.0621916727 * (zipcode_98144) + 0.0132896685 * (zipcode_98146) + 0.00133202944 * (zipcode_98148) + 0.0228541503 * (zipcode_98155) + 0.0175267394 * (zipcode_98166) + -0.00301818847 * (zipcode_98168) + 0.0317353193 * (zipcode_98177) + 0.00506879753 * (zipcode_98178) + 1.70332721e-05 * (zipcode_98188) + -0.00158028244 * (zipcode_98198) + 0.0754782143 * (zipcode_98199)

Interpretation of the coefficients in terms of feature importance:
The linear regression equation represents the estimated relationship between the features and the log of house price. interpretations assume a linear relationship between price and variables

Intercept: 13.04 is the estimated house log of price when all other predictor variables are zero..
Coefficients for Continuous Variables: The coefficients for continuous variables reveal specific associations with house prices: an extra bedroom corresponds to a $0.005 decrease, an additional bathroom links to a $0.03 increase, each unit increase in living space contributes to a $0.08 rise, an increase in lot size associates with a $0.03 increase, each floor relates to a $0.04 increase, and each additional year of age results in a $0.01 increase in the log of house price.
Coefficients for Categorical Variables: The coefficients for zipcode-related variables signify the estimated change in house price for each city compared to the reference zipcode (zipcode_98001), with, for instance, the coefficient for zipcode_98002 ($0.00085) indicating a $0.00085 higher estimated log of house price than the reference. Additionally, coefficients for features like is_renovated ($0.014) and Quality ($0.1143) provide insights into their impact on house prices, bearing in mind that these

Lasso and Ridge Regression:

Lasso Regression and Ridge Regression are both regularization techniques used in linear regression to address the issue of multicollinearity and to prevent overfitting. They add a penalty term to the ordinary least squares (OLS) objective function to constrain the coefficients of the regression model. Lasso Regression tends to shrink some coefficients exactly to zero, effectively performing feature selection
Performance Metrics of Ridge Model on Train Data:
Ridge Model with α =20

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Ridge	0.875	0.185	0.0342	0.1366	1.0505	0.881	0.1823	0.0332	0.1344	1.0322

Table 8 Metrics of Ridge Model

The Ridge regression model performs similarly to linear regression, with R-squared values of 0.875 for the test set and 0.88 for the train set. Despite introducing regularization, the model maintains high error metrics.

The coefficients of Ridge Model are:

```
Ridge model: [-0.00591739  0.03390936  0.09812044  0.02779525 -0.01405976  0.03971761
  0.04624676  0.03407114  0.10677524  0.09071553  0.11528704 -0.05075789
  0.04965583  0.00286554 -0.00524006  0.01284248  0.0141834   0.01713862
 -0.0006801  -0.00203521  0.10224076  0.04003744  0.06589514  0.03437061
  0.05048941  0.02089938  0.00809542  0.01243645  0.00676022  0.02250397
 -0.0119619   0.02396464  0.05202311  0.00509283  0.05822939  0.00314631
  0.00309239 -0.00667765  0.06869451  0.03302859  0.03144494  0.05059798
  0.07261822  0.01007675  0.0379684   0.05838743  0.04749727  0.00373386
  0.02290282  0.01336165  0.03094134  0.04154193  0.01318689  0.01606627
  0.04942481  0.04928798  0.01230085  0.00710689  0.04668574  0.0790888
  0.06357596  0.01427287  0.05563876  0.0141389   0.04855798  0.08419986
  0.08023415  0.06086381  0.07275119  0.03916789  0.0618889   0.06441964
  0.02794489  0.04017688  0.01176705  0.04683876  0.05183191  0.00737459
  0.00206739  0.00358276  0.01480914 -0.01113932  0.01826435 -0.00150228
 -0.00324238 -0.00311803  0.06347118]
```

Performance Metrics of Lasso Model on Train Data:

Lasso Model with α =0.1

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Lasso	0.6352	0.3161	0.0999	0.246	1.8839	0.636	0.3187	0.1016	0.249	1.9062

Table 9 Metrics of Lasso

The Lasso regression model performs poorly, with R- squared values of 0.63 for the test set and 0.64 for the train set.

The coefficients of Lasso Model are:

```
Lasso model: [ 0.          0.          0.14574459  0.          0.          0.
  0.0054247   0.          0.13804999  0.          0.11365913 -0.
  0.01052442  0.          0.          0.          0.          0.
 -0.         -0.          0.          0.          0.          0.
  0.          0.         -0.         -0.         -0.         -0.
 -0.          0.          0.         -0.          0.         -0.
 -0.         -0.          0.         -0.         -0.          0.
  0.         -0.         -0.          0.          0.         -0.
 -0.         -0.         -0.         -0.          0.          0.
  0.          0.         -0.         -0.          0.          0.
  0.         -0.          0.         -0.          0.          0.
  0.          0.          0.         -0.          0.          0.
 -0.          0.         -0.          0.          0.         -0.
 -0.         -0.         -0.         -0.          0.         -0.
 -0.         -0.          0.        ]
```

Notably, some predictors have coefficients reduced to zero, indicating feature selection. With very features selected for model building the r squared is very less .The low R-squared value suggests that the Lasso model does not fit the training data well, and its predictive power might be limited. The low R-squared value might be attributed to coefficient value of 0 for some important predictors .With very few variables the prediction accuracy tends to take a hit

Decision Tree Regressor

A Decision Tree Regressor predicts a continuous target variable based on input features.

Decision Tree models with default parameters and with some user input parameters

Performance Metrics on Train Data using Decision Tree Regressor with default parameters:

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Decision Tree with Default parameters	0.7608	0.256	0.0655	0.1824	1.4005	0.9996	0.0104	0.0001	0.0004	0.0035

Table 10 Metrics of Decision Tree with default parameters

Decision Tree Model with default parameters shows good predictive performance, with an R-squared value of 0.76 for the test set and 1.0 for the train set, indicating potential overfitting. The model has relatively high error metrics compared to Linear and Ridge Models.

Performance Metrics on Train Data using Decision Tree Regressor with following parameters:

random_state=100, max_depth=10, min_samples_leaf=5

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Decision Tree with user parameters	0.8231	0.2201	0.0484	0.1599	1.2267	0.892	0.1736	0.0301	0.1277	0.9813

Table 11 Metrics of Decision Tree with hyper parameters

Decision Tree Model with user parameters performs well, with an R-squared value of 0.82 for the test set and 0. 89 for the train set, suggesting a better fit than previous Model . The model achieves lower error metrics, indicating improved predictive accuracy.

Feature Importance from Decision Tree Regressor

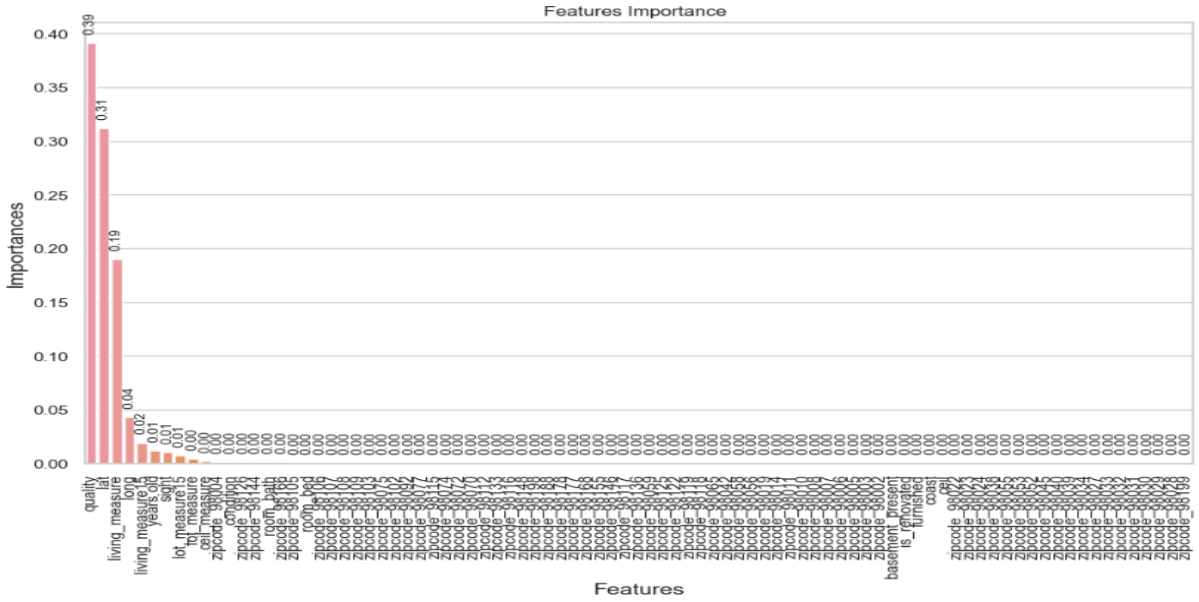


Figure 8 Feature Importance from Decision Tree Regressor

As we can see important features as per decision Tree regressor are Quality, Lat, living_measure, Long, living_measure 15, years_old, sight are most important features

Support Vector Regressor

Support Vector Regressor (SVR) works by finding a hyperplane that best fits the data while allowing a certain margin of error, making it suitable for predicting continuous outcomes in a robust and flexible manner.

Performance Metrics on Train Data using Support Vector Regressor:

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Support Vector	0.8805	0.1809	0.0327	0.1319	1.0147	0.9233	0.1463	0.0214	0.1157	0.8905

Table 12 Metrics of Support Vector

The Support Vector Regression (SVR) model performs well good R squared values 0.92 on train and 0.88 on test) for both test and train sets, indicating a good fit. Lower RMSE, MSE, and MAPE values compared to Linear models highlight good predictive capability.

KNN Regressor

K-Nearest Neighbors (KNN) Regressor predicts the target variable of a data point by averaging the values of its k nearest neighbors, where "k" is a user-defined parameter. The algorithm is non-parametric and relies on the similarity of feature values to make predictions

Performance Metrics on Train Data using KNN Regressor with neighbors=4, weights='distance'

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
KNN	0.8462	0.2053	0.0421	0.1465	1.1244	0.9996	0.0104	0.0001	0.0004	0.0035

Table 13 Metrics of KNN

The KNN regression model demonstrates good predictive accuracy with an R-squared value of 0.85 for the test set and 1.0 for the train set, suggesting potential overfitting. The model achieves lower error metrics compared to Ridge and Lasso models.

Here is a summary of the performance metrics of models built so far

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Linear	0.875	0.185	0.0342	0.1366	1.0504	0.8811	0.1822	0.0332	0.1343	1.0316
Ridge	0.875	0.185	0.0342	0.1366	1.0505	0.881	0.1823	0.0332	0.1344	1.0322
Lasso	0.6352	0.3161	0.0999	0.246	1.8839	0.636	0.3187	0.1016	0.249	1.9062
Supprt Vector	0.8805	0.1809	0.0327	0.1319	1.0147	0.9233	0.1463	0.0214	0.1157	0.8905
KNN	0.8462	0.2053	0.0421	0.1465	1.1244	0.9996	0.0104	0.0001	0.0004	0.0035
Decision Tree with Default parameters	0.7608	0.256	0.0655	0.1824	1.4005	0.9996	0.0104	0.0001	0.0004	0.0035
Decision Tree with user parameters	0.8231	0.2201	0.0484	0.1599	1.2267	0.892	0.1736	0.0301	0.1277	0.9813

Table 14 Model performances

Interpretation of the models

Linear Regression: With a test score (R-squared: 0.87) signifying a good fit and the lowest error metrics (RMSE, MSE, MAE,MAPE) among Linear Models, coupled with a training score (R-squared: 0.88) indicating a solid fit to the training data, this model serves as a suitable baseline, particularly when interpretability is a priority.
Ridge Regression: With a test score (R-squared: 0.87) and errors (RMSE, MSE, MAE,MAPE) comparable to Linear Regression, and a training score (R-squared: 0.88) indicating a similar fit, Ridge Regression can be considered, especially when addressing concerns about multicollinearity, as it maintains interpretability while mitigating the impact of correlated predictors through added regularization.
Lasso Regression: The model's test performance is poor (R-squared: 0.63), with considerably higher RMSE, MSE,MAPE and MAE compared to Linear and Ridge Regression, indicating a notable gap between predicted and actual values. While Lasso introduces feature selection and performs similarly to linear regression, its reduced interpretability and sparsity might lead to exclusion of predictors, making it less suitable in cases where interpretability is crucial.
Support Vector Regression (SVR): Test Score (R-squared): 0.88, indicating strong performance; RMSE, MSE, MAE and MAPE on Test comparable to Linear Models; Training Score (R-squared): 0.92, demonstrating consistent good performance; SVR models are effective in high-dimensional spaces, robust to outliers, and can handle non-linear relationships
k-Nearest Neighbors (KNN): Test Score (R-squared): 0.85, indicating a good fit to the test data; Lower RMSE, MSE, MAE,MAPE on Test compared to linear models, signifying better predictive accuracy; Training Score (R-squared): 1.00, hinting at potential overfitting; KNN models, especially with more neighbors, lack interpretability regarding individual predictor impacts.
Decision Trees (DT): DT with default parameters, showed moderate performance on the test set (R-squared: 0.76) but displayed potential overfitting (Training Score: 1.00). In contrast, DT, with user input parameters, exhibited improved test performance (R-squared: 0.82), surpassing DT and aligning with linear models, while maintaining interpretability and lower errors.

To improve the model performance, we use Ensemble modelling technique and model tuning methods like Hyper parameters tuning

Ensemble modelling

Ensemble modeling enhances predictive performance and robustness in machine learning by combining multiple models, a particularly effective approach for continuous variable prediction, such as in price prediction problems. Here are performance metrics from ensemble methods for price prediction:

	Test					Train				
Ensemble Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Random Forest	0.8682	0.19	0.0361	0.1369	1.0515	0.9124	0.1564	0.0245	0.1131	0.8688
Bagging with Decision Tree	0.8673	0.1906	0.0363	0.1373	1.0543	0.9117	0.157	0.0246	0.1136	0.8728
Bagging with Random Forest	0.8707	0.1882	0.0354	0.1353	1.0393	0.905	0.1628	0.0265	0.1167	0.8961
Gradient Boosting with Default Parameters	0.8727	0.1867	0.0349	0.1373	1.0558	0.8894	0.1757	0.0309	0.1295	0.9951
Gradient Boosting with user Parameters	**0.9013**	**0.1645**	**0.027**	**0.1169**	**0.9013**	**0.9628**	**0.102**	**0.0104**	**0.0734**	**0.5667**
ADA Boosting	0.8791	0.1819	0.0331	0.1357	1.0429	0.9312	0.1386	0.0192	0.1135	0.8719
XG Boosting with default Parameters	0.8914	0.1724	0.0297	0.1227	0.9439	0.9557	0.1112	0.0124	0.0822	0.6333
XG Boosting with Dart	**0.9002**	**0.1653**	**0.0273**	**0.1166**	**0.8983**	**0.9711**	**0.0898**	**0.0081**	**0.0653**	**0.5043**
XG Boosting with Gbtree	0.8948	0.1698	0.0288	0.1221	0.94	0.9256	0.1441	0.0208	0.1055	0.8124

Figure 9 Performance Metrics of Ensemble Models

For Ensemble Model Building Steps refer to Appendix D

Insights from Ensemble Modelling:

- Ensemble models such as Random Forest, Bagging with Decision Tree, and Bagging with Random Forest exhibit balanced performance, with good R-squared and relatively low errors
- XG Boosting with Dart and Gradient Boosting with tuned parameters, stands out with the highest performance, robustness, and efficiency in capturing underlying patterns among the ensemble models.
- Gradient Boosting, both with default and user parameters, demonstrates effectiveness comparable to Random Forest, showcasing the power of ensemble methods in capturing complex data relationships, though it may require tuning for optimal performance.

Hyperparameter Tuning with cross validation:

Further optimization and hyperparameter tuning using GridSearchCV and RandomizedSearchCV may improve the performance of certain models, especially Gradient Boosting and XG Boosting. Hyperparameter tuning and cross-validation are crucial for optimizing the performance of regression and ensemble models.
For More information on Hyper Parameter Tuning of Model Building refer to Appendix E

Here are performance metrics from Hyper parameter Tuning methods for price prediction:

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Ridge with GridsearchCV	0.8750	0.1850	0.0342	0.1366	1.0502	0.8810	0.1822	0.0332	0.1343	1.0318
Lasso with GridsearchCV	0.6352	0.3161	0.0999	0.2460	1.8839	0.6360	0.3187	0.1016	0.2490	1.9062
KNN with GridsearchCV	0.8613	0.1949	0.0380	0.1380	1.0595	0.9996	0.0104	0.0001	0.0004	0.0035
RF with GridsearchCV	0.8812	0.1804	0.0325	0.1286	0.9886	0.9688	0.0934	0.0087	0.0689	0.5305
XGB with RandomSearchCV	0.8967	0.1682	0.0283	0.1192	0.9177	0.9513	0.1166	0.0136	0.0852	0.6568
XGB with GridsearchCV	0.9000	0.1655	0.0274	0.1185	0.9131	0.9253	0.1444	0.0208	0.1051	0.8093
Gradient Boosting with GridsearchCV	**0.9011**	**0.1646**	**0.0271**	**0.1172**	**0.9029**	**0.9514**	**0.1165**	**0.0136**	**0.0837**	**0.6453**

Figure 10 Performance Metrics of Model after Hyperparameter Tuning

Insights from Model Tuning

- Ensemble models (Random Forest, XG Booster, Gradient Booster) with tuning generally outperform simpler models like Ridge, Lasso, and KNN.
- The Random Forest model, after tuning, exhibits strong performance with a high R-squared on the training set and good performance on the test set, as evidenced by a reasonable R-squared and lower RMSE.
- Similarly, the XG Booster model, both with GridSearchCV and RandomsearchCV performs well with high R-squared on the training set and lower RMSE on the test set compared to simpler models.
- The Gradient Booster model, optimized with GridSearchCV, outperforms others with the highest R-squared on the test and second highest on training sets, accompanied by a lower RMSE ,MSE ,MAE and MAPE on the test set.

Model validation

Choosing Best Model and Validation of Model using R squared RMSE, MSE, MAE, MAPE

Ensemble models (Random Forest, XG Booster, Gradient Booster) with tuning generally outperform simpler models like Ridge, Lasso, and KNN.
The complexity of the models seems to be positively correlated with their predictive performance.
It's crucial to consider the balance between model complexity and interpretability based on the specific needs of the problem. Ensemble models, while more complex, provide strong predictive capabilities.

We achieved 90% accuracy with a gradient boosting model with 87 predictors .We can apply PCA(Principal component analysis) to 87 predictors to reduce dimensions, but even with 50 principal components, only 70% of the data variability is explained , rendering it less beneficial. So, we will not apply PCA on dataset.

In the context of Regression, (R-squared), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), Mean absolute Percentage Error (MAPE) are commonly used metrics to assess the performance of a regression model. For details of each of these Metrics refer to Appendix G

Here is a summary of performance metrics of all the models we have built so far

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Linear Regression	0.875	0.185	0.0342	0.1366	1.0504	0.8811	0.1822	0.0332	0.1343	1.0316
Ridge	0.875	0.185	0.0342	0.1366	1.0505	0.881	0.1823	0.0332	0.1344	1.0322
Lasso	0.6352	0.3161	0.0999	0.246	1.8839	0.636	0.3187	0.1016	0.249	1.9062
Support Vector	0.8805	0.1809	0.0327	0.1319	1.0147	0.9233	0.1463	0.0214	0.1157	0.8905
KNN	0.8462	0.2053	0.0421	0.1465	1.1244	0.9996	0.0104	0.0001	0.0004	0.0035
Decision Tree with Default parameters	0.7608	0.256	0.0655	0.1824	1.4005	0.9996	0.0104	0.0001	0.0004	0.0035
Decision Tree with user parameters	0.8231	0.2201	0.0484	0.1599	1.2267	0.892	0.1736	0.0301	0.1277	0.9813
Ensemble Models										
Random Forest	0.8682	0.19	0.0361	0.1369	1.0515	0.9124	0.1564	0.0245	0.1131	0.8688
Bagging with Decision Tree	0.8673	0.1906	0.0363	0.1373	1.0543	0.9117	0.157	0.0246	0.1136	0.8728
Bagging with Random Forest	0.8707	0.1882	0.0354	0.1353	1.0393	0.905	0.1628	0.0265	0.1167	0.8961
Gradient Boosting with Default Parameters	0.8727	0.1867	0.0349	0.1373	1.0558	0.8894	0.1757	0.0309	0.1295	0.9951
Gradient Boosting with user Parameters	**0.9013**	**0.1645**	**0.027**	**0.1169**	**0.9013**	**0.9628**	**0.102**	**0.0104**	**0.0734**	**0.5667**
ADA Boosting	0.8791	0.1819	0.0331	0.1357	1.0429	0.9312	0.1386	0.0192	0.1135	0.8719
XG Boosting with default Parameters	0.8914	0.1724	0.0297	0.1227	0.9439	0.9557	0.1112	0.0124	0.0822	0.6333
XG Boosting with Dart	0.8914	0.1724	0.0297	0.1227	0.9439	0.9557	0.1112	0.0124	0.0822	0.6333
XG Boosting with Gbtree	0.8948	0.1698	0.0288	0.1221	0.94	0.9256	0.1441	0.0208	0.1055	0.8124
Hyper Parameter Tuned Models										
Ridge with GridsearchCV	0.875	0.185	0.0342	0.1366	1.0502	0.881	0.1822	0.0332	0.1343	1.0318
Lasso with GridsearchCV	0.6352	0.3161	0.0999	0.246	1.8839	0.636	0.3187	0.1016	0.249	1.9062
RF with GridsearchCV	0.8812	0.1804	0.0325	0.1286	0.9886	0.9688	0.0934	0.0087	0.0689	0.5305
XGB with RandomSeracgCV	0.8967	0.1682	0.0283	0.1192	0.9177	0.9513	0.1166	0.0136	0.0852	0.6568
XGB with GridsearchCV	0.9000	0.1655	0.0274	0.1185	0.9131	0.9253	0.1444	0.0208	0.1051	0.8093
KNN with GridsearchCV	0.8613	0.1949	0.038	0.138	1.0595	0.9996	0.0104	0.0001	0.0004	0.0035
Gradient Boosting with GridsearchCV	0.9011	0.1646	0.0271	0.1172	0.9029	0.9514	0.1165	0.0136	0.0837	0.6453

Table 15 Performance Metrics on All models

Why Gradient Boost

- Best Model selection hinges on balancing interpretability and accuracy, with linear models offering simplicity and interpretability, while ensemble models like Random Forest and boosting methods excel in accuracy but can be more complex. Regularization techniques like Ridge and Lasso help prevent overfitting in linear models by penalizing large coefficients.
- Gradient Boosting is often considered superior to linear models and other ensemble models with similar accuracy metrics like RMSE or MAE due to its ability to capture complex nonlinear relationships in the data. Gradient Boosting builds a strong predictive model by combining weak learners (typically decision trees) sequentially, each correcting the errors of the previous one. This adaptability allows it to outperform linear models and other ensembles in capturing intricate patterns and relationships within the data, leading to improved predictive performance in various scenarios.

Gradient Boosting as highlighted in the figure above seems to be the best optimum model. Here are reasons supporting this choice:

- **High Validation Score (R-squared):**
  The Gradient Boosting Model has a test score of 0.9013 and train score of 0.96, indicating that it explains a substantial proportion of the variance in the dependent variable without overfitting .
- **Lowest RMSE, MSE, MAE, and MAPE:**
  The model has the lowest test RMSE (0.1645), MSE (0.0270), MAE (0.1169), and MAPE (0.9013) among the models considered. Lower values for these metrics indicate better predictive performance.
- **Consistent Performance Across Metrics:**
  The Gradient Boosting with Parameters consistently performs well across all metrics, demonstrating its robustness and effectiveness in capturing the underlying patterns in the data.
- **Outperforms Other Models:**
  In comparison to other models, including Linear Regression, Support Vector Regressor, Random Forest, and others, Gradient Boosting consistently shows superior performance, making it a strong candidate for the best model.
- **Consideration of Both Complexity and Performance:**
  Gradient Boosting achieves a high level of performance without excessive complexity. This balance is essential to avoid overfitting and ensures that the model is likely to generalize well to new, unseen data.

Interpretation of Metrics for Gradient Boost Model:

When we have log-transformed the target variable (price) and are working with a model like Gradient Boost, it's important to keep in mind that the interpretation of the evaluation metrics is based on the log-transformed scale.
Interpretation of the metrics:

- An R-squared value of 0.9013 indicates that 90% of the variance in log-transformed prices is explained by the model.
- With an RMSE of 0.1645, the average difference between predicted and true log-transformed prices is approximately 0.1645, while an MSE of 0.0270 signifies the squared average difference. An MAE of 0.1169 implies an average absolute difference between predicted and true log-transformed prices, and a MAPE of 0.9013% signifies the average absolute percentage difference between predicted and actual log-transformed prices.
- If we need to interpret the results in terms of the original prices, we can exponentiate the predicted values and compare them with the original prices for a more meaningful interpretation.

Here is a table that shows the Actual Vs Predicted prices on the sample of test set

	Test Data	Predicted Price	Difference
0	390,000.0000	417,034.5405	27,034.5405
1	290,000.0000	390,345.5098	100,345.5098
2	455,000.0000	506,095.7470	51,095.7470
3	524,000.0000	560,901.3478	36,901.3478
4	295,000.0000	306,089.1833	11,089.1833
5	594,000.0000	556,058.8775	37,941.1225
6	660,000.0000	615,783.3674	44,216.6326
7	272,500.0000	291,759.4159	19,259.4159
8	95,000.0000	180,239.5044	85,239.5044
9	411,500.0000	344,484.2249	67,015.7751

Table 16 Predicted Vs Actual Price Difference

In Gradient Boost, feature importance is another metric that helps us understand the contribution of each feature (variable) in making predictions, higher values indicate greater importance. For complete List of Feature importance refer to Appendix I. Here are some of top features influencing house Prices.

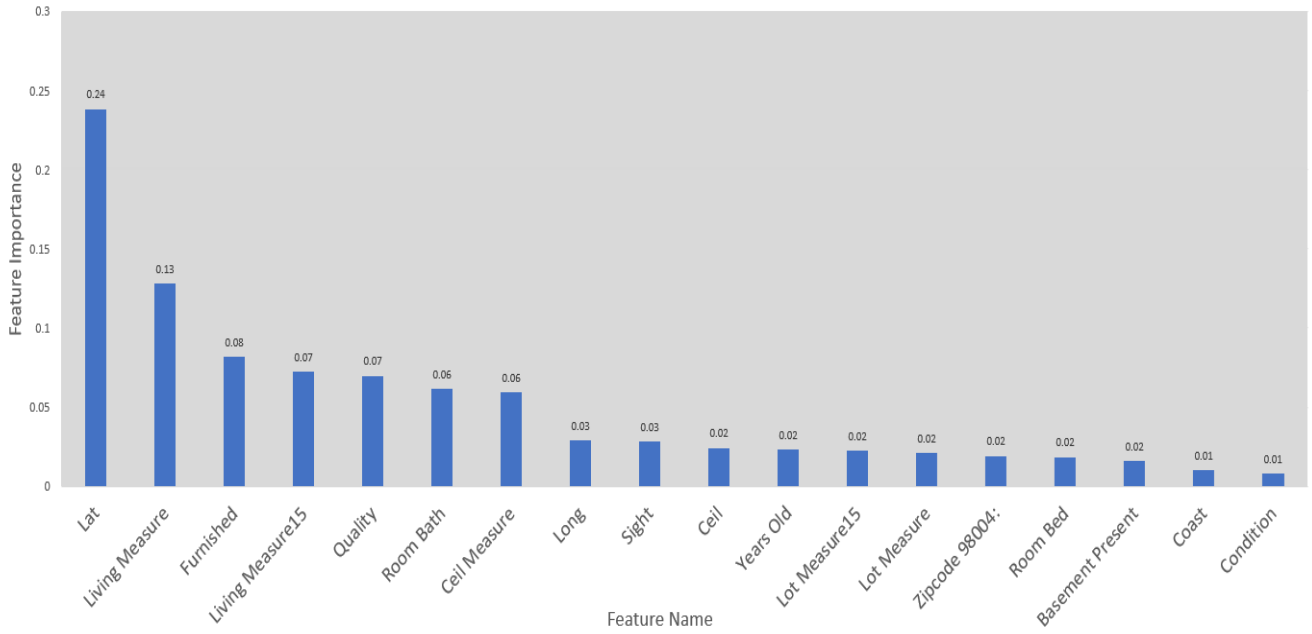


Figure 11 Top Feature Importance from Gradient Boost

Insights from Feature Importance

- Latitude (lat):Geographical location, as indicated by latitude, strongly influences house prices, with certain latitudinal positions associated with higher property values.
- Living Measure: The size of the current as well as past living area significantly impacts house prices, suggesting that larger living spaces command higher values in the real estate market.
- Furnished Status: The presence of furnishings positively contributes to property values, indicating that furnished homes may be perceived as more valuable.
- Quality: The quality of construction or finishes is a key determinant of house prices, with higher-quality homes being associated with elevated property values.
- Room Bath: The number of bathrooms plays a significant role in predicting house prices, suggesting that homes with more bathroom's command higher market values.
- Ceiling Measure: The size of the ceiling contributes to property values, indicating that features related to the ceiling impact the overall desirability of a home.
- Longitude (long): While less influential than latitude, longitude still has a notable impact on house prices, highlighting the importance of the property's overall geographical position.
- Number of Views: Properties with attractive views positively influence house prices, suggesting that scenic surroundings contribute to the perceived value of a home.
- Number of Floors: The number of floors in a property is a relevant factor, indicating that multi-story homes may have higher market values.
- Age of Property (Years Old): While still contributing to the model, the age of the property has a moderate influence on house prices, suggesting that newer properties may not be the sole driver of increased values.
- Lot Measure 15: The overall size of the lot has a moderate effect on house prices, highlighting that larger lots may contribute to higher property values.
- Zip Codes (e.g., zipcode_98004, zipcode_98040): Different zip codes exhibit varying degrees of importance, emphasizing the diverse impact of location-specific factors on house prices. Other Zip Codes (e.g., zipcode_98056, zipcode_98030) have minimal influence on house prices, suggesting that properties in these areas may be less affected by location-specific factors.
- Basement Present: The presence of a basement has a moderate impact on house prices, indicating that homes with basements may be perceived as more valuable.
- Waterfront View: While still contributing, having a waterfront view has a smaller influence on house prices, suggesting that this feature is not as critical in determining property values.
- Condition: The overall condition of the property has a relatively low impact on house prices

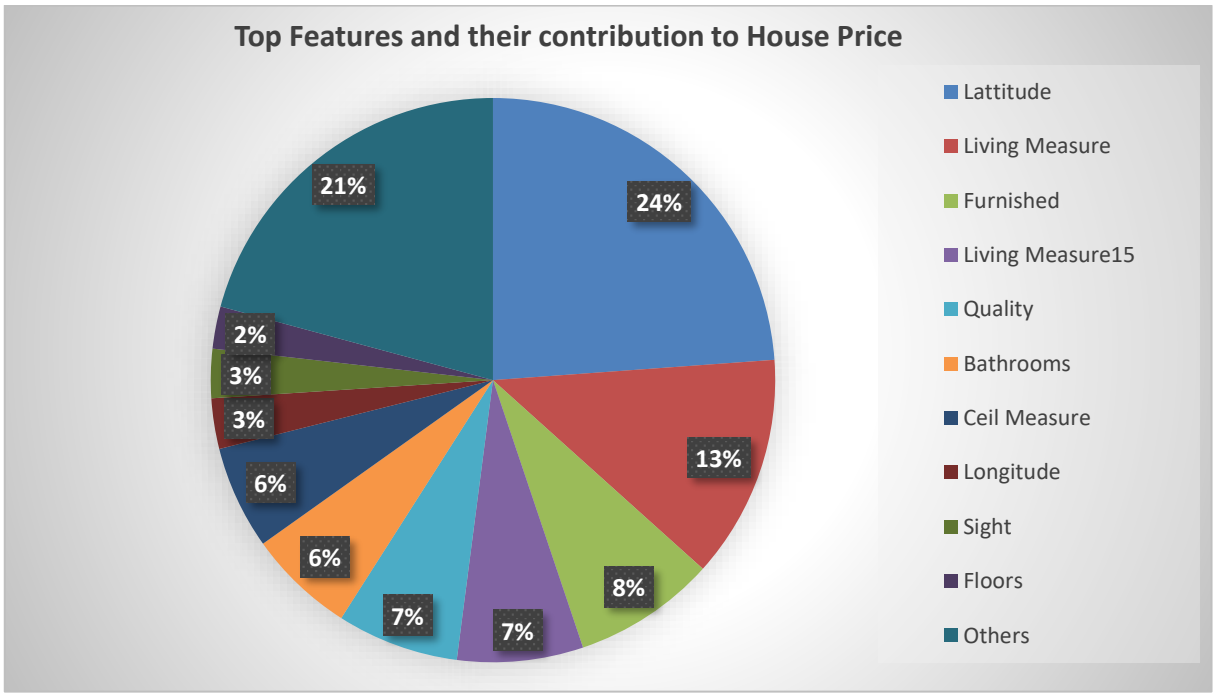
Final interpretation / recommendation

Interpretations from Model Building

- Linear models are suitable for straightforward interpretation; Ridge and Lasso handle multicollinearity and feature selection . But it has limitations in capturing complex relationships.
- Decision Tree provides interpretable decision rules but may overfit the training data. Its Suitable for decision-making but may lack generalization.
- KNN is prone to overfitting and SVR is a good model but sensitive to outliers .
- Advanced ensemble methods XGBoost and Gradient Boost combining weak learners, Offers high accuracy, handles non-linear relationships, and adapts to complex patterns.
- Gradient Boosting performs the best among the ensemble methods. It's preferred for its superior predictive performance, making it a robust choice for accurate house price predictions. Careful hyperparameter tuning may further enhance its effectiveness.

Interpretations from EDA and Feature Importance Analysis

- **Geographic Location is Paramount**: Geographical location of the property play a pivotal role in determining house prices
- **Size Matters** : Property size, both in terms of current as well as past living area and Floor(Ceil) measures , is a critical factor in predicting house prices.
- **Furnished Homes Hold Value**: Furnished properties have a notable impact on house prices, suggesting that well-appointed homes may command higher values.
- **Quality and Amenities Influence Prices**: Quality of construction, the number of bathrooms, and the presence of premium features impact property values.
- **Viewed Property** : Properties that are viewed help discern the house prices better
- **Stories Matter** : Multi-storied homes command higher market values.
- **Location-Specific Considerations**: Different zip codes have varying importance, indicating diverse location-specific factors influencing house prices.
- **Moderate Impact of Property Age and Condition**: Property age and condition have a moderate influence on house prices.
- **Waterfront Views Add Value**: The presence of waterfront views contributes to property values to a moderate extent



Business Recommendations for Optimizing Housing Market Price

- Emphasize superior quality in construction materials, design, and finishes to justify higher prices. Highlight premium features, high-quality finishes, and superior construction standards in property listings to attract buyers seeking a luxurious living experience.
- Offer furnished options or well-staged properties to provide a hassle-free transition for buyers. Highlight the convenience of move-in-ready homes and emphasize the presence of furnishings in property marketing to showcase added value and convenience for potential buyers.
- Highlight area uniqueness, emphasizing amenities and accessibility. Tailor marketing to showcase specific neighborhood appeal, emphasizing prime location attractiveness local amenities, community features, and lifestyle offerings for potential buyers.
- Highlight spacious living areas in marketing, considering renovations or expansions for added value. Emphasize generous living spaces, outdoor areas, and larger lots to attract buyers seeking ample living space.
- Actively market waterfront properties, justifying premium pricing, and highlight their picturesque views in listings to appeal to potential buyers.
- Utilize online platforms and marketing strategies to increase property visibility. Professional photography, virtual tours, and engaging property descriptions can enhance online appeal.
- Ensure well-maintained properties for premium market appeal.
- Highlight recent renovations for a modern feel. Emphasize property age positively. Leverage historical features and current conditions for enhanced perceived value.
- Tailor marketing strategies to the unique characteristics of each zip code, addressing preferences and trends, and customize messages based on specific features to enhance the property's overall appeal and value proposition.
- Leverage positive features like quality, specific zip codes, and waterfront views, while addressing negatives by mitigating the impact of less favorable features through targeted marketing or renovations.
- Regularly monitor housing market trends and update models as needed to ensure relevance. Establish a process for continuous evaluation of feature importance and model performance, adapting strategies based on shifts in market dynamics and buyer preferences.
- Develop marketing materials and campaigns that educate potential buyers on the significance of positive impact features. Clearly communicate the value proposition to potential buyers, enhancing understanding and appreciation for key property features.

Appendix A

Column Description

Column	Description
cid	A notation for a house
dayhours	Date house was sold
price	Price is prediction target
room_bed	Number of Bedrooms/House
room_bath	Number of bathrooms/bedrooms
living_measure	Square footage of the home
lot_measure	Square footage of the lot
ceil	Total floors (levels) in house
coast	Does the house have a view to a waterfront
sight	How many times property has been viewed
condition	Overall condition of the house. Rating given between 1 to 5, where 1 is the poor and 5 is the best
quality	Grade given to the housing unit, based on grading system ranging from 1 to 13. 1 being least and 13 being highest
ceil_measure	Square footage of house apart from basement
basement_measure	Square footage of the basement
yr_built	Year when house was built
yr_renovated	Year when house was renovated
zipcode	ZipCode of the property
lat	Latitude coordinate
long	Longitude coordinate
living_measure15	Living room area in 2015(implies-- some renovations) This might or might not
lot_measure15	lotSize area in 2015(implies-- some renovations)
furnished	Is the property furnished? 0 if Not furnished and 1 if Furnished
total_area	Square Footage of both living and lot

Table 17 Column Description

Top 5 records of the dataset

	cid	dayhours	price	room_bed	room_bath	living_measure	lot_measure	ceil	coast	sight	condition	quality	ceil_measure	basement
0	3876100940	20150427T000000	600000	4.00	1.75	3050.00	9440.00	1	0	0.00	3	8.00	1800.00	1250.00
1	3145600250	20150317T000000	190000	2.00	1.00	670.00	3101.00	1	0	0.00	4	6.00	670.00	0.00
2	7129303070	20140820T000000	735000	4.00	2.75	3040.00	2415.00	2	1	4.00	3	8.00	3040.00	0.00
3	7338220280	20141010T000000	257000	3.00	2.50	1740.00	3721.00	2	0	0.00	3	8.00	1740.00	0.00
4	7950300670	20150218T000000	450000	2.00	1.00	1120.00	4590.00	1	0	0.00	3	7.00	1120.00	0.00

yr_built	yr_renovated	zipcode	lat	long	living_measure15	lot_measure15	furnished	total_area
1966	0	98034	47.72	-122.18	2020.00	8660.00	0.00	12490
1948	0	98118	47.55	-122.27	1660.00	4100.00	0.00	3771
1966	0	98118	47.52	-122.26	2620.00	2433.00	0.00	5455
2009	0	98002	47.34	-122.21	2030.00	3794.00	0.00	5461
1924	0	98118	47.57	-122.28	1120.00	5100.00	0.00	5710

Table 19 Top 5 records

	cid	dayhours	price	room_bed	room_bath	living_measure	lot_measure	ceil	coast	sight	condition	quality	ceil_measure	basement	yr_built	yr_renovated	zipcode	lat	long	living_measure15	lot_measure15	furnished	total_area
21608	203600600	20150310T000000	685530	4.00	2.50	3130.00	60467.00	2	0	0.00	3	9.00	3130.00	0.00	1996	0	98014	47.66	-121.96	2780.00	44224.00	1.00	63597
21609	625049281	20140521T000000	535000	2.00	1.00	1030.00	4841.00	1	0	0.00	3	7.00	920.00	110.00	1939	0	98103	47.69	-122.34	1530.00	4944.00	0.00	5871
21610	424069018	20140905T000000	998000	3.00	3.75	3710.00	34412.00	2	0	0.00	3	10.00	2910.00	800.00	1978	0	98075	47.59	-122.04	2390.00	34412.00	1.00	38122
21611	7258200055	20150206T000000	262000	4.00	2.50	1560.00	7800.00	2	0	0.00	3	7.00	1560.00	0.00	1997	0	98168	47.51	-122.32	1160.00	7800.00	0.00	9360
21612	8805900430	20141229T000000	1150000	4.00	2.50	1940.00	4875.00	2	0	0.00	4	9.00	1940.00	0.00	1925	0	98112	47.64	-122.30	1790.00	4875.00	1.00	6815

Table 18 Bottom 5 records

Bottom 5 records of the dataset

- CID (Property ID):"CID" is a numerical variable representing property IDs. Duplicate IDs exist in the dataset, each associated with different property attributes.
- Price (Target Variable): "Price" is the target variable with a range between $75,000 and $7,700,000. The mean price is $540,182.16.
- Room_bed (Number of Bedrooms):"Room_bed" is a numerical variable indicating the number of

bedrooms, varying from 0 to 33.

- Room_bath (Number of Bathrooms):"Room_bath" is a numerical variable indicating the number of bathrooms, varying from 0 to 8. It has decimal values like 1.5 . 1.5 bath would mean one full bathroom and one-half bathroom. A half bath is just a toilet and sink. There isn't a shower or tub in it, so a house with 1.5 baths has one full bathroom and one-half bath
- Living_Measure (Living Space Square Footage): "Living_Measure" is a numerical variable indicating the square footage of living space, ranging from 290 sqft to 13,540 sqft.
- Lot_measure (Lot Size Square Footage):"Lot_measure" is a numerical variable indicating the square footage of the lot, ranging from 520 sqft to 1,651,359 sqft.
- Ceil (Number of Floors):"Ceil" is a numerical variable indicating the number of floors, ranging from 1 to 3.5. A 3.5 story house is defined as a building with three full stories and a partial third story that is usually smaller and may only be used for storage or as a sleeping area.
- Ceil_measure (Living Space Square Footage without Basement):"Ceil_measure" is a numerical variable indicating the square footage of the house, excluding the basement, ranging from 290 sqft to 9,410 sqft.
- Basement (Basement Square Footage):"Basement" is a numerical variable indicating the square footage of the basement, ranging from 0 to 4,820 sqft.
- Yr_Built (Year Built):"Yr_Built" is a numerical variable indicating the year the house was built, ranging from 1900 to 2015, indicating some houses are over 100 years old.
- Yr_Renovated (Year Renovated):"Yr_Renovated" is a numerical variable indicating the year the property was renovated. Many houses have not been renovated, resulting in a value of 0.
- Lat (Latitude):"Lat" is a numerical variable indicating the latitude of the property's location, ranging from 47.16 degrees to 47.78 degrees.
- Long (Longitude): "Long" is a numerical variable indicating the longitude of the property's location, ranging from -121.31 degrees to -122.52 degrees.
- Living_measure15 (Living Space Square Footage in 2015):
- "Living_measure15" is a numerical variable indicating the square footage of living space in 2015, ranging from 399 sqft to 6,210 sqft.
- Lot_measure15 (Lot Size Square Footage in 2015):"Lot_measure15" is a numerical variable indicating the square footage of the lot in 2015, ranging from 651 sqft to 871,200 sqft.
- Total_Area (Total Square Footage of Lot plus Living Area): "Total_Area" is a numerical variable indicating the total square footage of the lot plus living area, ranging from 1,423 sqft to 1,652,659 sqft.
- Coast:The "Coast" variable is categorical, denoting whether a property has a waterfront view. It has two unique values, 0 and 1. A value of 1 indicates a waterfront view, while 0 signifies no waterfront view.
- Sight:"Sight" is a categorical variable indicating the number of times a property has been viewed, with 5 unique values (0, 1, 2, 3, 4). The most frequently occurring category is 0, suggesting that a substantial portion of properties has not been viewed.
- Condition:The "Condition" variable is categorical, representing the condition of the property with 5 unique values (1, 2, 3, 4, 5). The frequently occurring category is 3, implying that properties with this condition are prevalent in the dataset.
- Quality:"Quality" is a categorical variable indicating the quality of the property, encompassing 12 unique values from 1 to 13. The frequently occurring category is 7, indicating that properties of this quality are prevalent in the dataset.
- Zipcode:"Zipcode" is a categorical variable with 70 unique values. Among these, 98103 is the frequently occurring zipcode, suggesting a concentration of properties in this specific area.
- Furnished:"Furnished" is a categorical variable denoting whether a property is furnished, with 2 unique values (0 and 1). The frequently occurring category is 0, indicating that unfurnished properties dominate the dataset.

Appendix B

Uni-variate Analysis:

The CID variable is not relevant to our analysis. We need it to check for duplicate CID entries, of which there are 176. These indicate that the same property has had several sales in the past. We can remove the CID variable from your analysis because it is merely a property ID and has no utility in model building.

Analysis of Dayhours variable

Figure 12 Dayhours variable

'dayhours' has only 2 values 2014 and 2015 that have distributions, meaning that properties were sold in year 2014 and 2015, with majority sold in year 2014.

Analysis of Price variable -Price is the target variable. Below is the distribution plot this variable and analyze the variable

```
Distribution of price
Skewness  of price : 4.02
Kurtosis  of price : 34.514180829647714
```

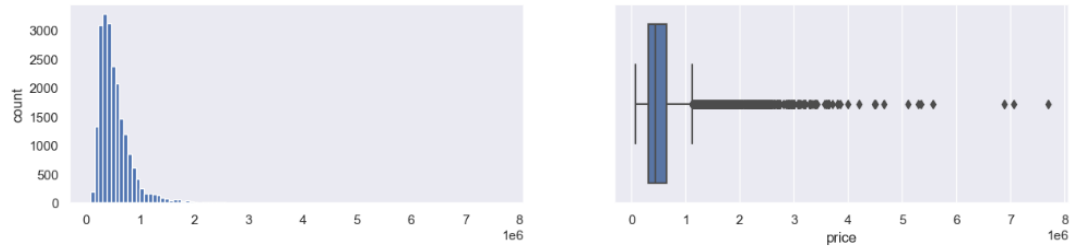


Figure 13 Distribution of Price Variable

Price Variable has a right skewed distribution with skewness of 4.02 and Kurtosis of 34.5 and from boxplot we see that there are many outliers in this variable. It has a mean of $540188.16 and standard deviation of 367362.23. Minimum Price is $75000 and Maximum value is $7700000. Median Price is $450000. Majority of the properties are in the range $30000-$50000 followed by $50000-$80000. Least number of properties are between $50000-$100000 and above $ 500000.

Visual Analysis of Continuous Numerical variables

Analysis of living_measure

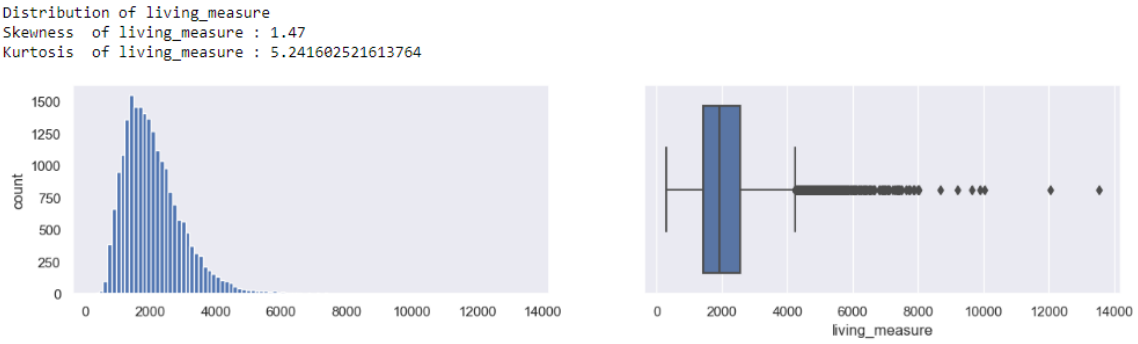


Figure 14 Density plot and Boxplot of Living Measure

From the Density plot and boxplot, we can see that it is right skewed with skewness of 1.47 and Kurtosis of 5.24 and from boxplot we see that there are many outliers in this variable. Let's see the data summary of the **living_measure** variable. It has a mean of 2079.90 sq ft and standard deviation of 918.44 sq ft. Minimum **living_measure** is 290 sq ft and Maximum **living_measure** value is 13540 sq ft. Median **living_measure** is 1910 sq ft. As mean is greater than median value, it's a right skewed distribution as indicated by the density plot and boxplot

mean	2079.90
std	918.44
min	290.00
25%	1427.00
50%	1910.00
75%	2550.00
max	13540.00

Table 20 Data summary of Living Measure

Analysis of lot measure

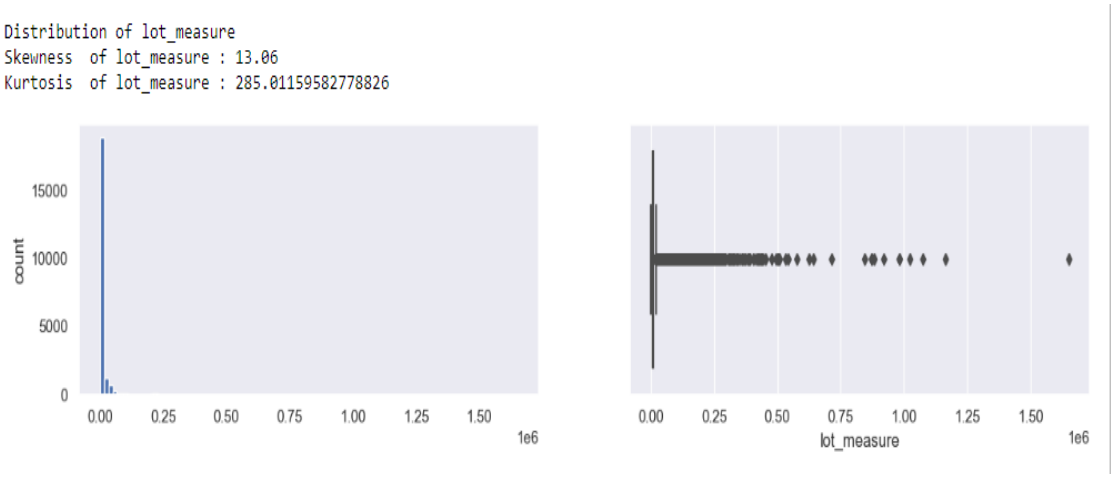


Figure 15 Density plot and Boxplot of lot Measure

From the Density plot and boxplot, we can see that it is heavily right skewed with skewness of 13.07 and from boxplot we see that there are many outliers in this variable.

Data summary of the **lot_measure** variable.

mean	15106.97
std	41420.51
min	520.00
25%	5040.00
50%	7618.00
75%	10688.00
max	1651359.00

Table 21 Data Summary of Lot Measure

It has a mean of 15106.97 sq ft and standard deviation of 41420.51 sq ft. Minimum **lot _measure** is 520 sq ft and Maximum **lot _measure** value is 1651359 sq ft. Median **lot _measure** is 7618 sq ft. As mean is much

higher than median value, it's a right skewed distribution as indicated by the density plot and boxplot

Analysis of ceil measure

```
Distribution of ceil_measure
Skewness  of ceil_measure : 1.45
Kurtosis  of ceil_measure : 3.401665753958614
```

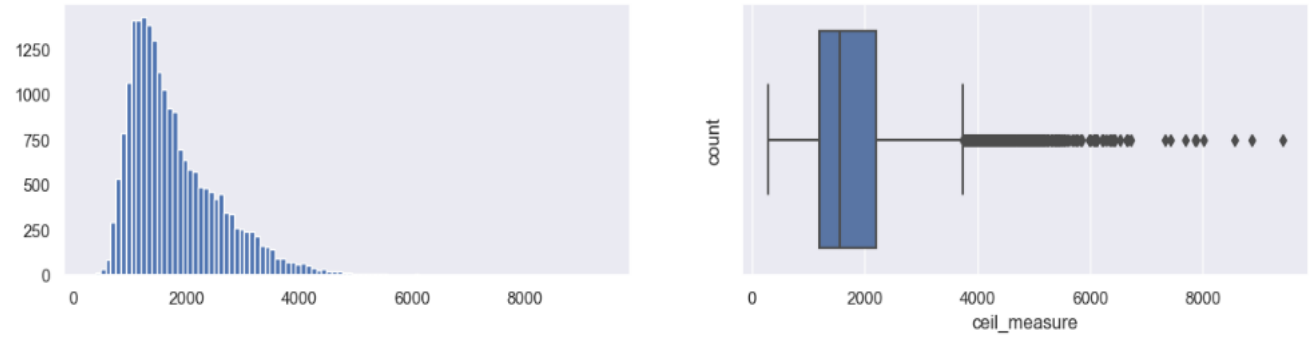


Figure 16 Density plot and Boxplot of Ceil Measure

From the Density plot and boxplot, we can see that it is right skewed with skewness of 1.45 and Kurtosis of 3.4 and from boxplot we see that there are outliers in this variable.
Data summary of the **ceil_measure** variable

mean	1788.37
std	828.10
min	290.00
25%	1190.00
50%	1560.00

Table 22 Data summary of ceil Measure

It has a mean of 1788.37 sq ft and standard deviation of 828.1 sq ft. Minimum **ceil _measure** is 290 sq ft and Maximum **ceil _measure** value is 9410 sq ft. Median **ceil _measure** is 1560 sq ft. As mean is higher than median value, it's a right skewed distribution as indicated by the density plot and boxplot

Analysis of basement measure

```
Distribution of basement
Skewness  of basement : 1.58
```

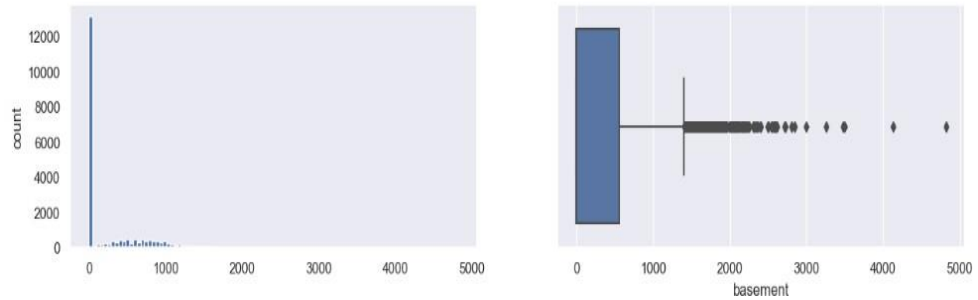


Figure 17 Density plot and Boxplot of Basement Measure

From the Density plot and boxplot, we can see that it is right skewed with skewness of 1.58 and from boxplot we see that there are outliers in this variable. Kurtosis is not available because there are 0 values in dataset
It has a mean of 291.52 sq ft and standard deviation of 442.58 sq ft. Minimum **basement _measure** is 0 sq ft and Maximum **basement _measure** value is 4820 sq ft. Median **basement _measure** is 0 sq ft meaning 50% of the properties have basement measure 0. As mean is higher than median value, it's a right skewed distribution as indicated by the density plot and boxplot
Data summary of the **basement _measure** variable.

mean	291.52
std	442.58
min	0.00
25%	0.00
50%	0.00
75%	560.00
max	4820.00

Table 23 Data summary of basement Measure

Analysis of Living measure15

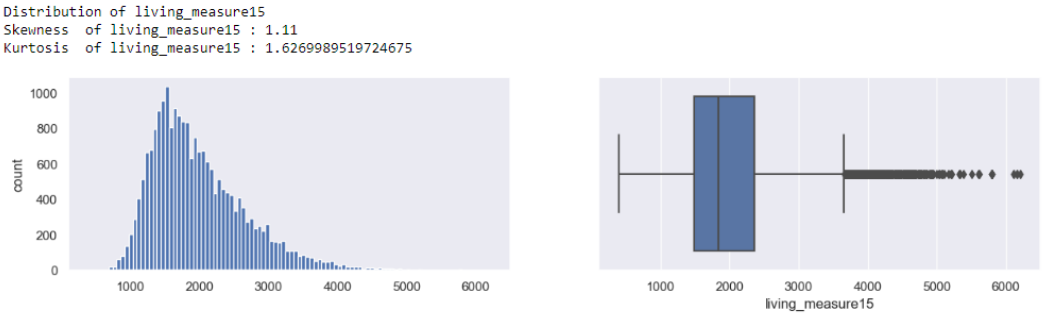


Figure 18 Density plot and Boxplot of Living Measure15

From the Density plot and boxplot, we can see that it is right skewed with skewness of 1.11 and Kurtosis of 1.62 and from boxplot we see that there are outliers in this variable.
It has a mean of 1987.07 sq ft and standard deviation of 685.52 sq ft. Minimum Living _measure15 is 399 sq ft and Maximum Living _measure15value is 6210 sq ft. Median Living _measure15 is 1840 sq ft meaning 50% of the properties have Living _measure15 value below 1840sqft. As mean is slightly higher than median value, it's a slightly right skewed distribution as indicated by the density plot and boxplot

mean	1987.07
std	685.52
min	399.00
25%	1490.00
50%	1840.00
75%	2360.00
max	6210.00

Table 24 Data summary of living Measure15

Analysis of Lot measure15

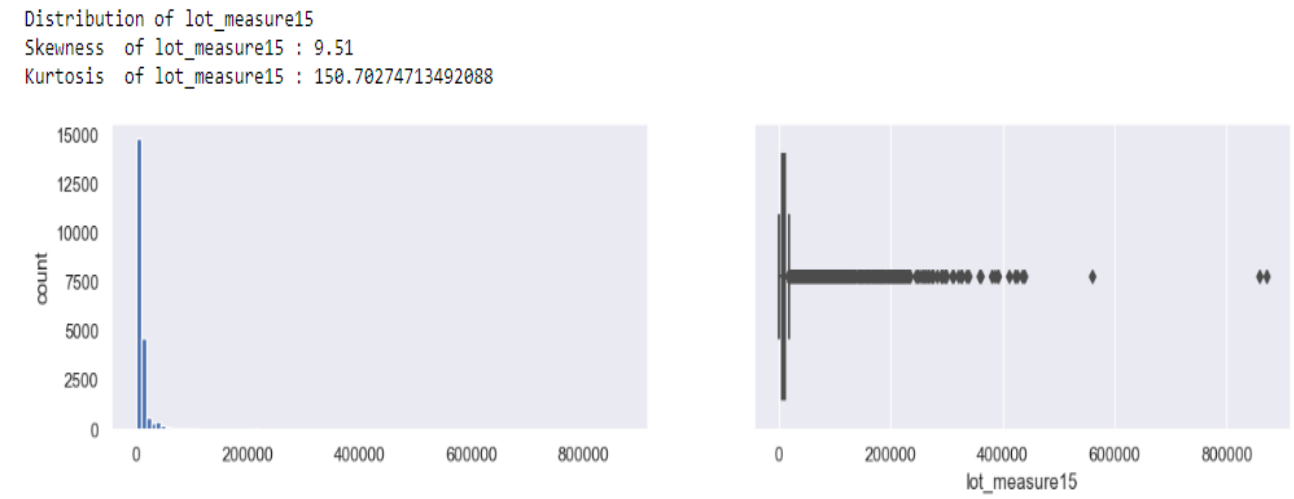


Figure 19 Density plot and Boxplot of lot Measure15

From the Density plot and boxplot, we can see that it is right skewed with skewness of 9.52 and Kurtosis of 150 and from boxplot we see that there are outliers in this variable.

It has a mean of 12766.54 sq ft and standard deviation of 27286.99 sq ft. Minimum lot_measure15 is 651 sq ft

and Maximum lot_measure15 value is 871200 sq ft. Median lot_measure15 is 7620 sq ft meaning 50% of the properties have lot_measure15 value below 7620 sqft. As mean is much higher than median value, it's a heavily right skewed distribution as indicated by the density plot and boxplot

mean	12766.54
std	27286.99
min	651.00
25%	5100.00
50%	7620.00
75%	10087.00
max	871200.00

Table 25 Data Summary of Lot Measure15

Analysis of Total Area

```
Distribution of total_area
Skewness  of total_area : 12.96
Kurtosis  of total_area : 281.1510003301337
```

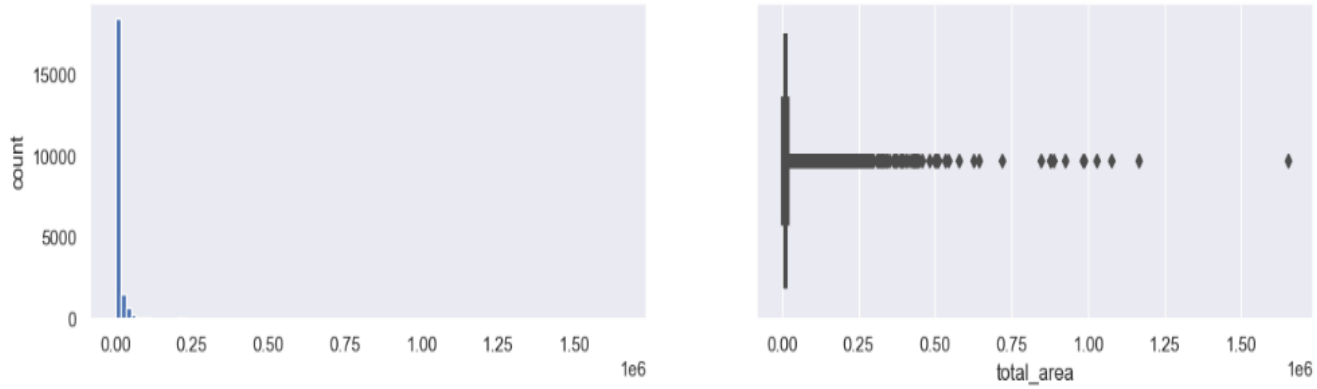


Figure 20 Density plot and Boxplot of Total area

From the Density plot and boxplot, we can see that it is highly right skewed with skewness of 12.96 and Kurtosis of 281.15 and from boxplot we see that there are outliers in this variable.
. It has a mean of 17186.87 sq ft and standard deviation of 41589.08 sq ft.
Minimum Total_area is 1423 sq ft and Maximum Total_area value is 1652659 sq ft. Median Total_area is 9575 sq ft meaning 50% of the properties have Total_area value below 9575 sqft. As mean is much higher than median value, it's a heavily right skewed distribution as indicated by the density plot and boxplot

mean	17186.87
std	41589.08
min	1423.00
25%	7035.00
50%	9575.00
75%	13000.00
max	1652659.00

Table 26 Data summary of Total Area

Figure 21 Distributions all Continuous Numerical Variable

Visual Analysis of Temporal variables

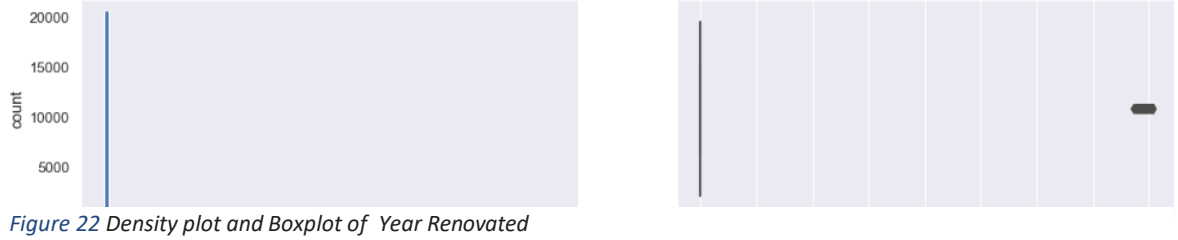


Figure 22 Density plot and Boxplot of Year Renovated

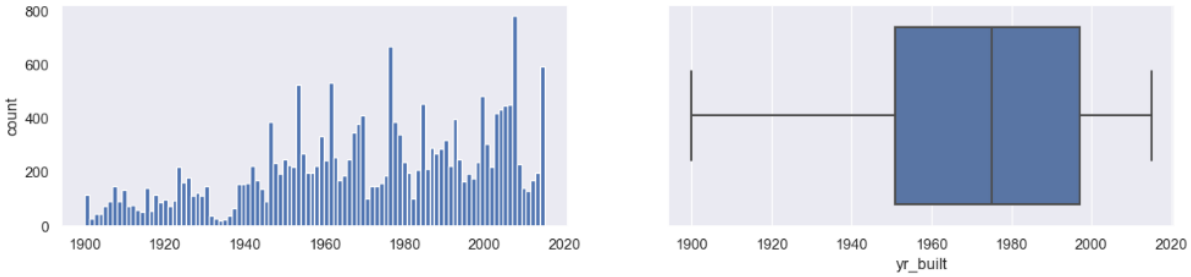


Figure 23 Density plot and Boxplot of Year Built

Properties have been built between 1900 and 2015. Majority of properties have Year Renovated as 0, meaning there has been no renovation. Renovations that have happened are between 1934 and 2015.

Visual Analysis of categorical variables

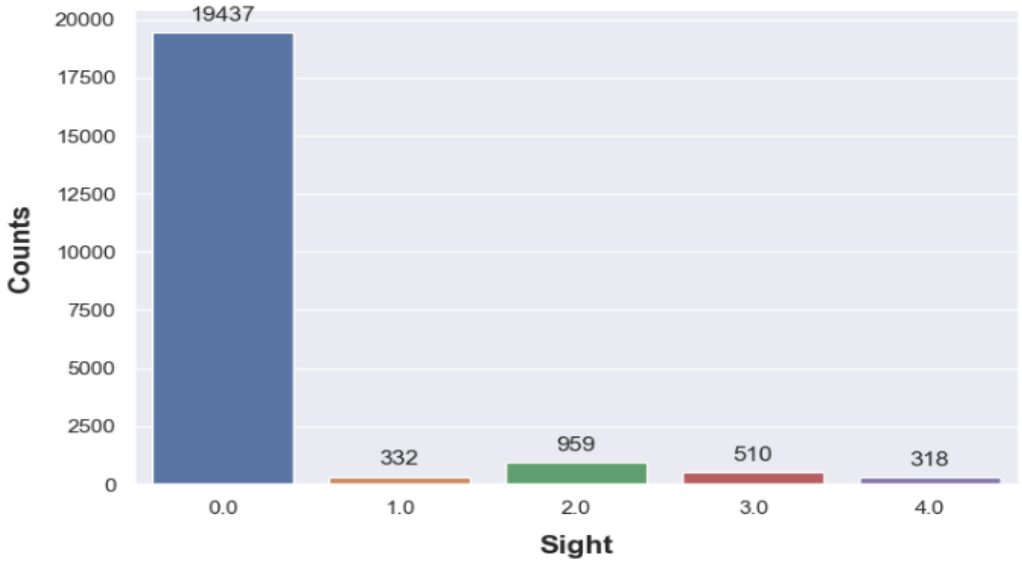


Figure 24 Count plot of Sight variable

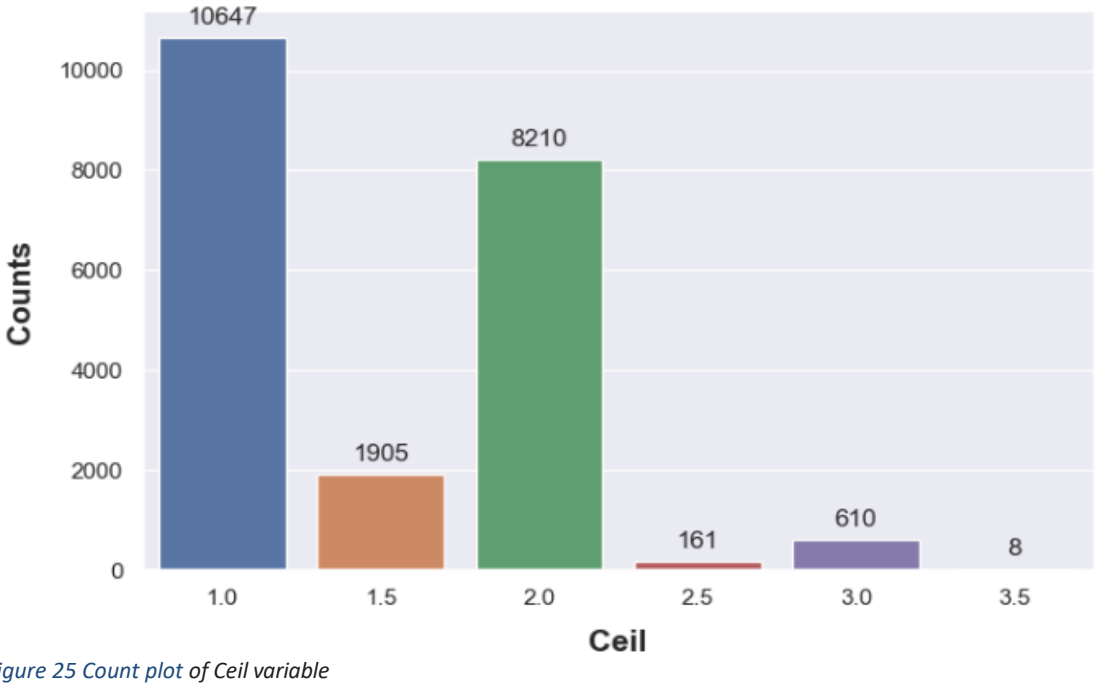


Figure 25 Count plot of Ceil variable

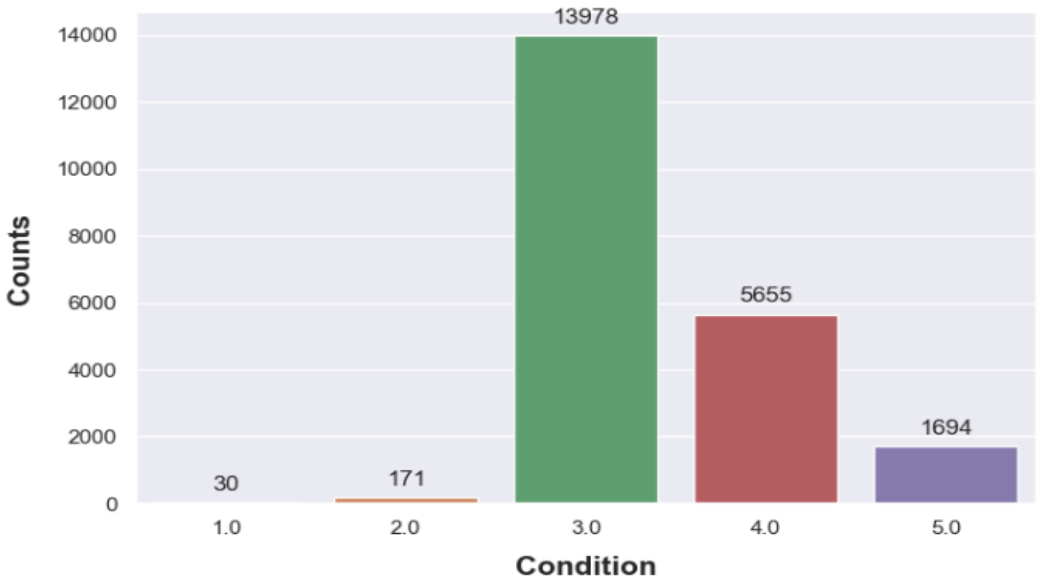


Figure 26 Count plot of condition variable

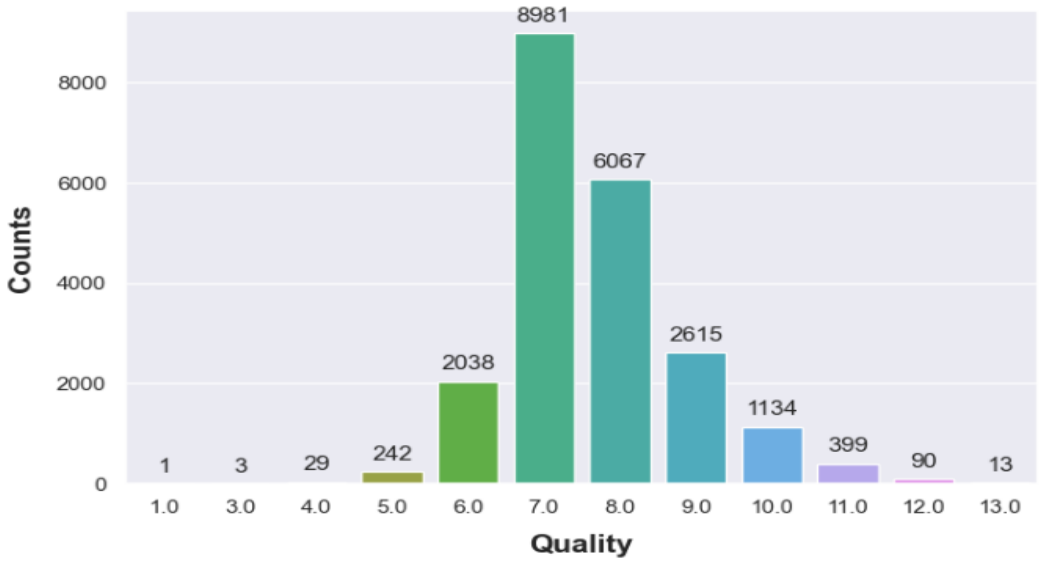


Figure 27 Count plot of quality variable

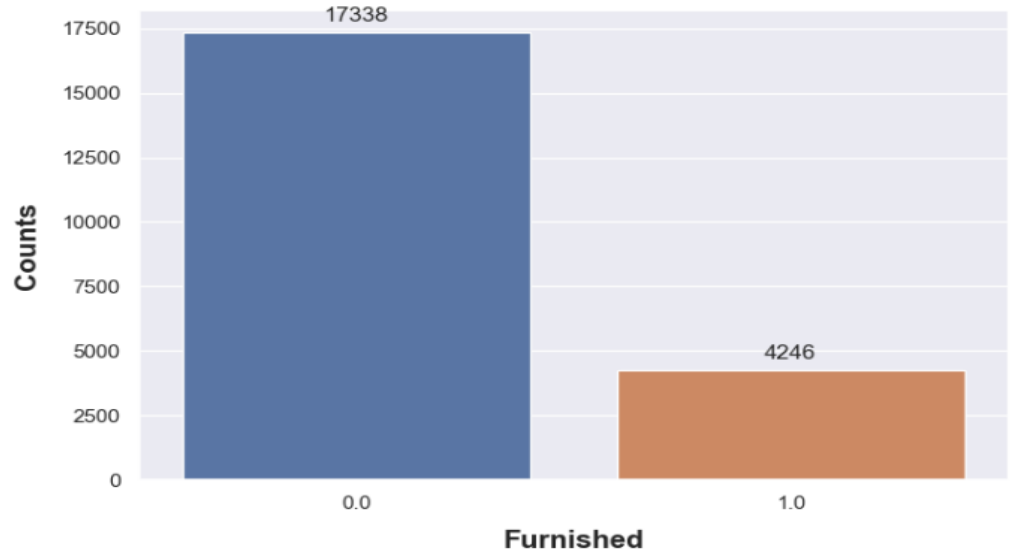


Figure 28 Count plot of Furnished

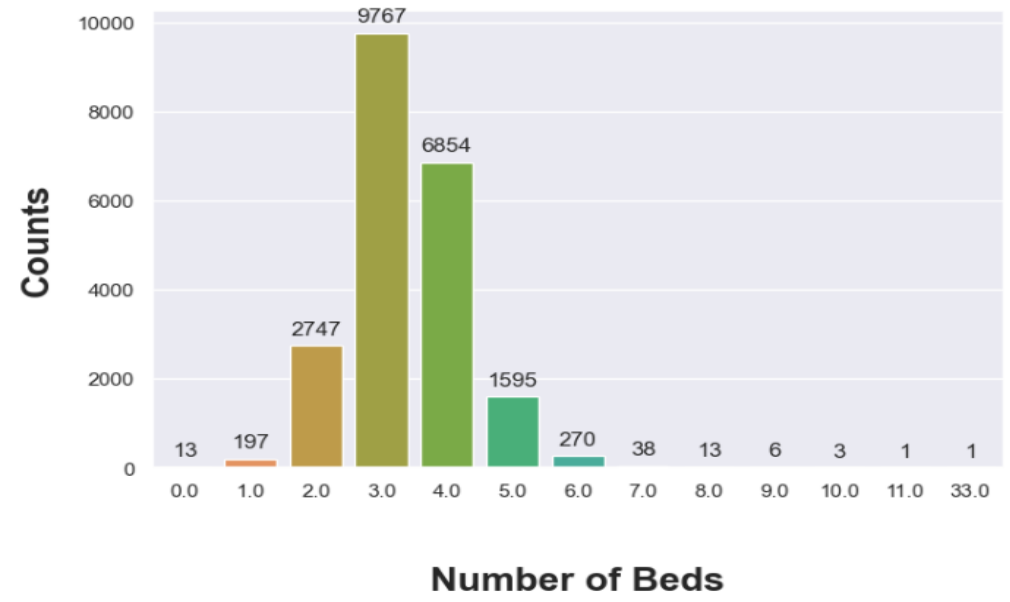


Figure 29 Count plot of Number of Bedrooms

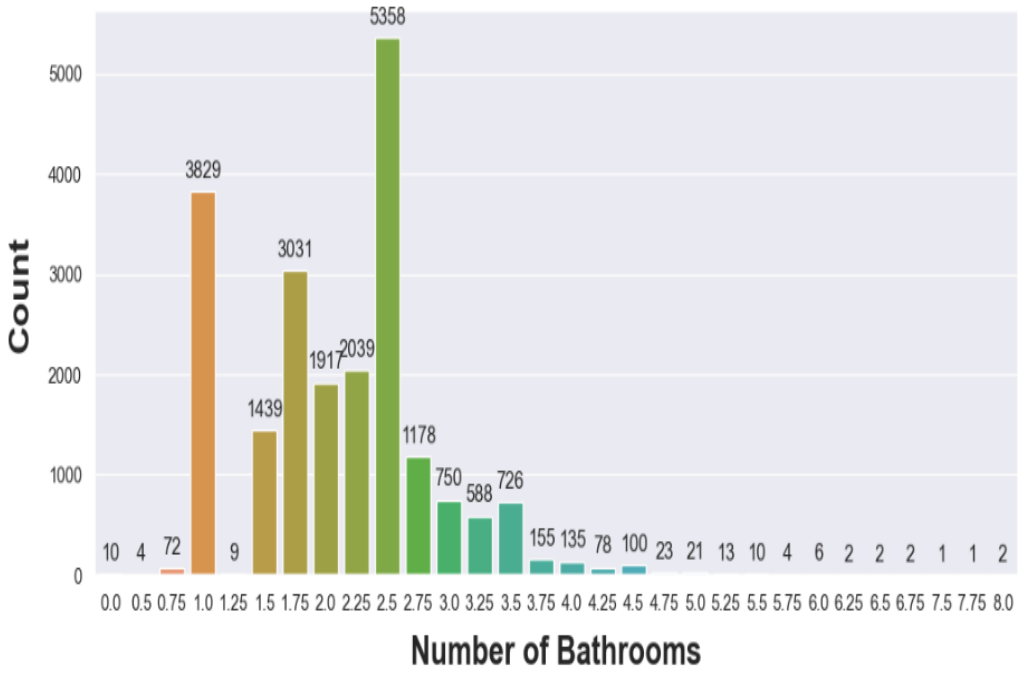


Figure 30 Count plot of Number of Bathrooms

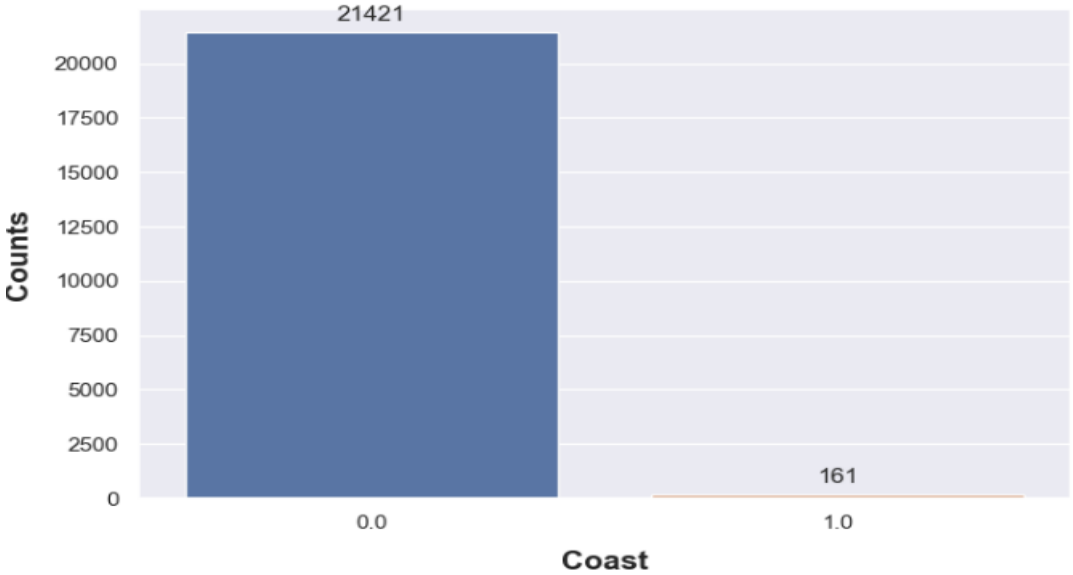
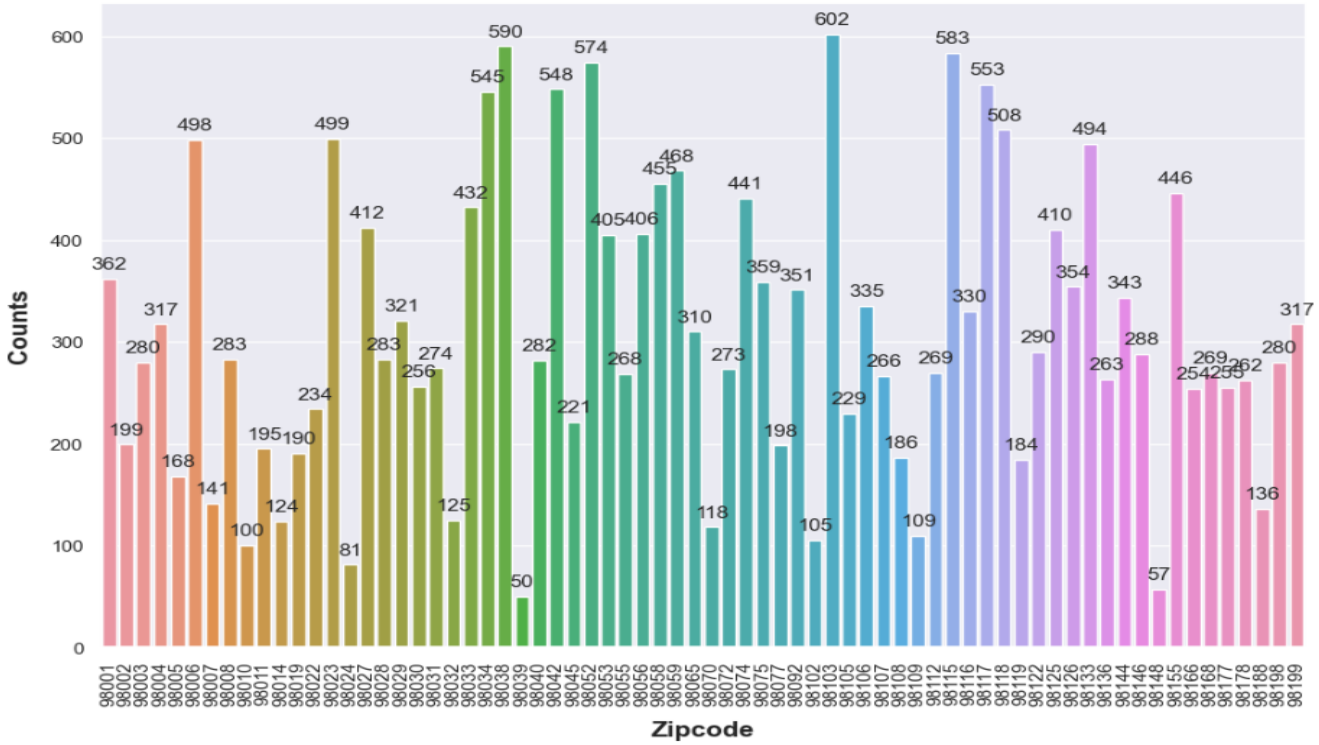


Figure 31-Count plot of Coast

- 80% of the properties are not furnished.
- Quality Ranges from 1 to 13 .42 % of the properties have quality 7 and 28 % with quality 8. Only 1 property is of least quality 1 and there are 13 properties which are of best quality rating 13.
- Condition ranges from 1 to 5.65 % of the properties have condition 3 and 26 % with condition 4. Only 7.9% of the properties are in best condition i.e. 5
- 90.2 % of the properties have no views and 99.3 % of the properties have no waterfront view.
- Bedrooms ranges from 0 to 33 .77% of the properties have 3 or 4 bedrooms .Bathrooms range from 10 to 8. 25 % of the properties have 2.5 bathrooms .
- Ceil ranges from 1 to 3.5 .87% of the properties have 1 or 2 Floors .

Visual Analysis of Geographical variables



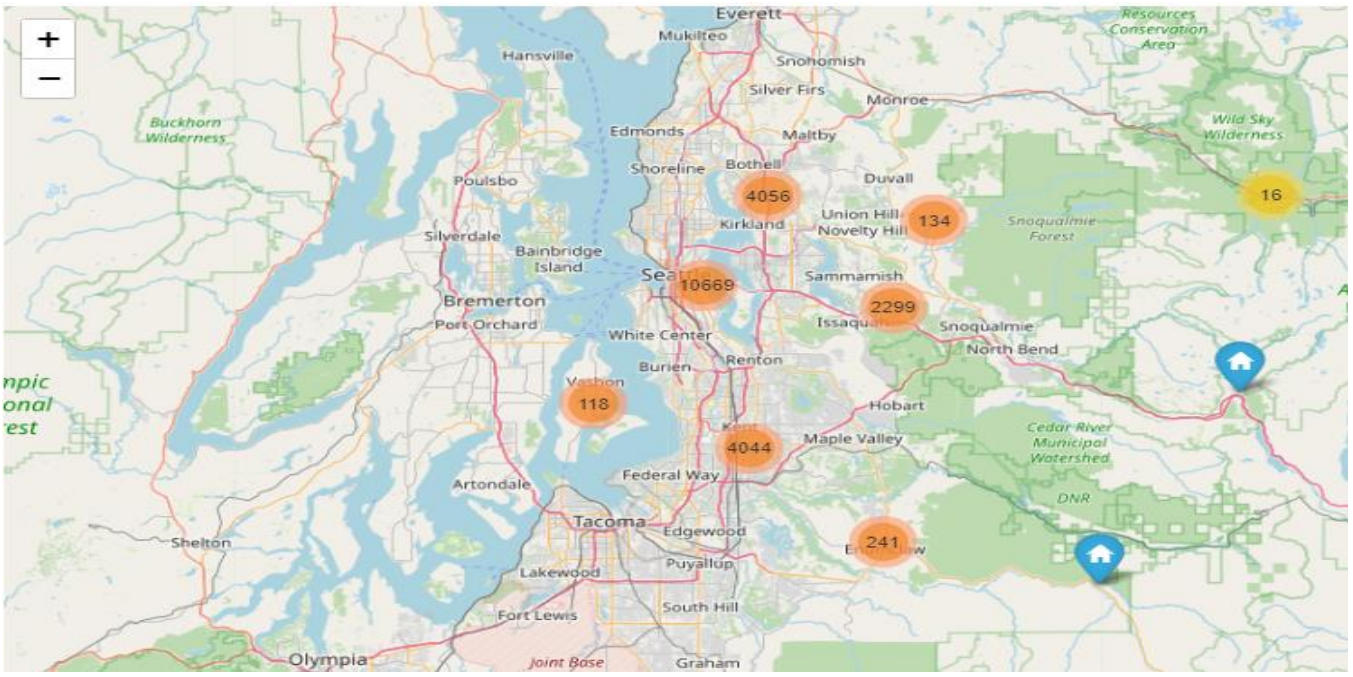


Figure 32 Latitude Vs longitude

All properties are in Kings County of US state of Washington. On plotting the zip code variables, we

see that maximum number of the properties listed are from zip code 98103 which amount to 41.5 % properties and least number of properties are listed from zipcode 98039

Bi-variate Analysis (relationship between different variables, correlations)

Relationship between Price and all the categorical variables

Analysis of Price Vs number of bedrooms

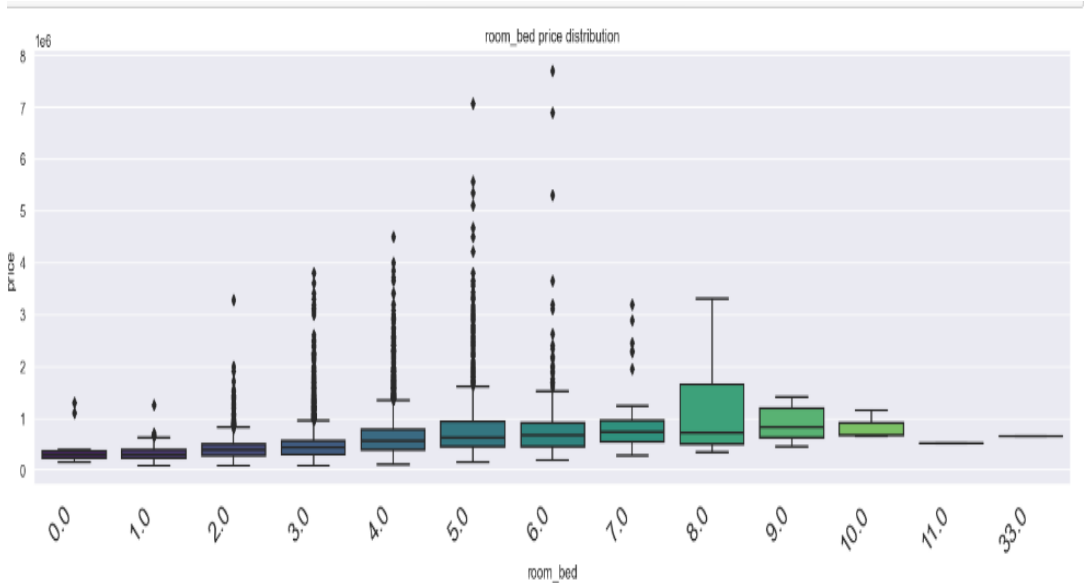


Figure 33 Boxplot of Price Vs room_bed

Analysis of Price Vs number of bathrooms

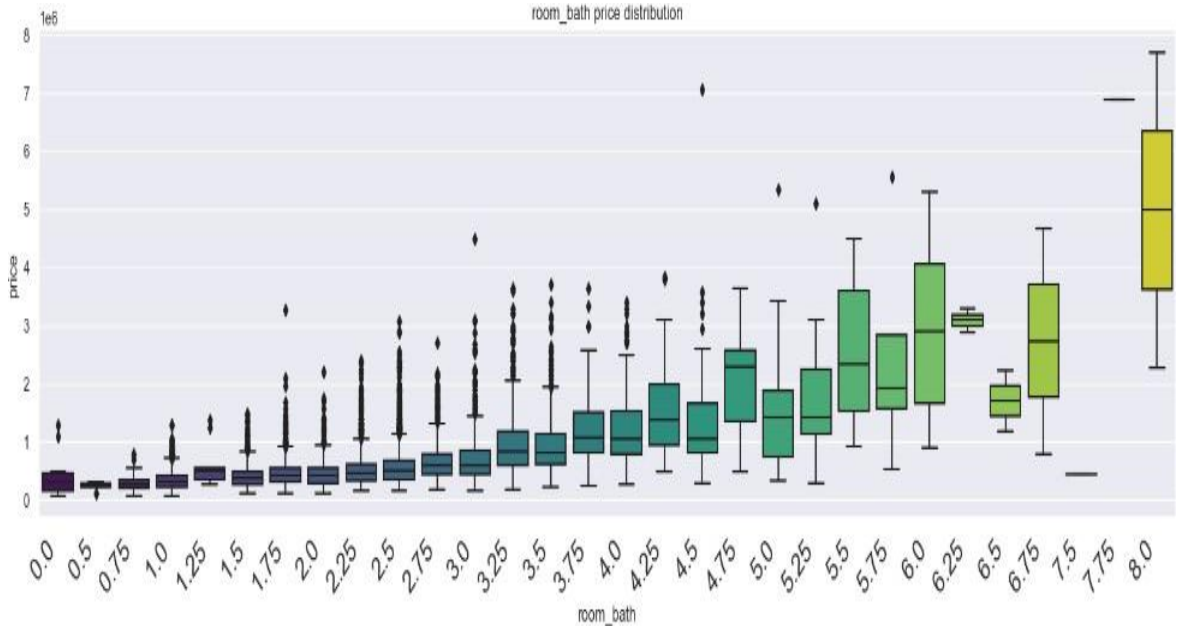


Figure 34 Boxplot of Price Vs room_bath

Analysis of Price Vs number of floors

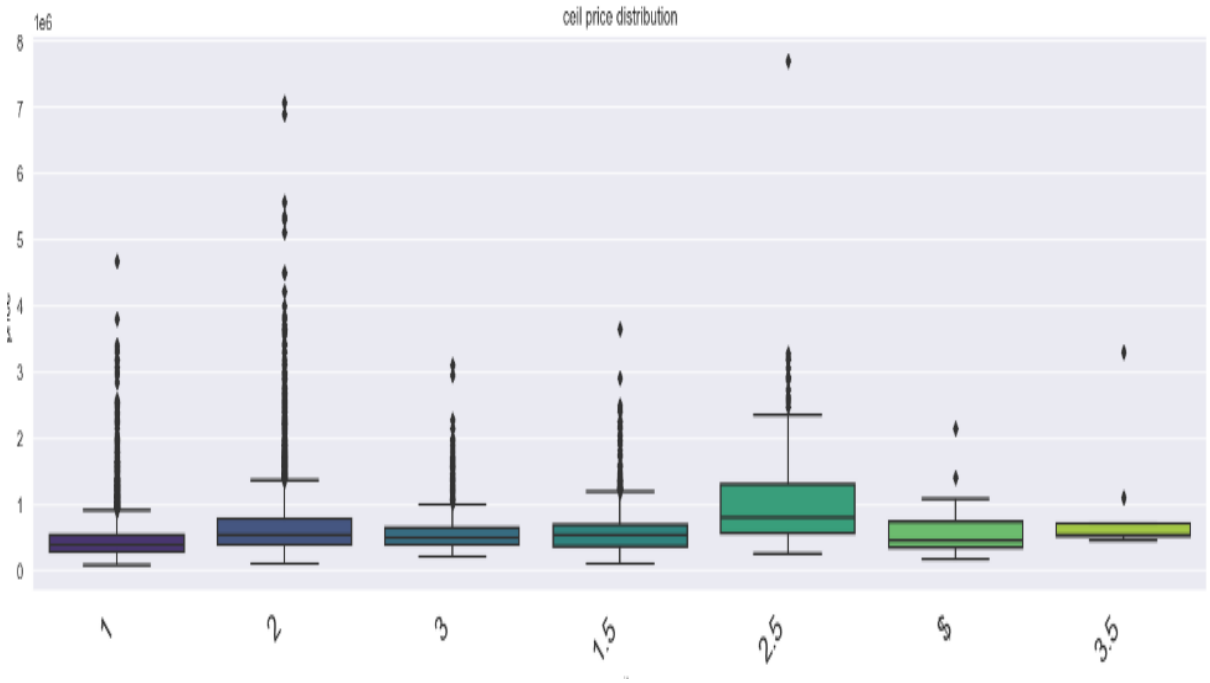


Figure 35 *Boxplot of Price Vs Ceil*

Analysis of Price Vs Waterfront View

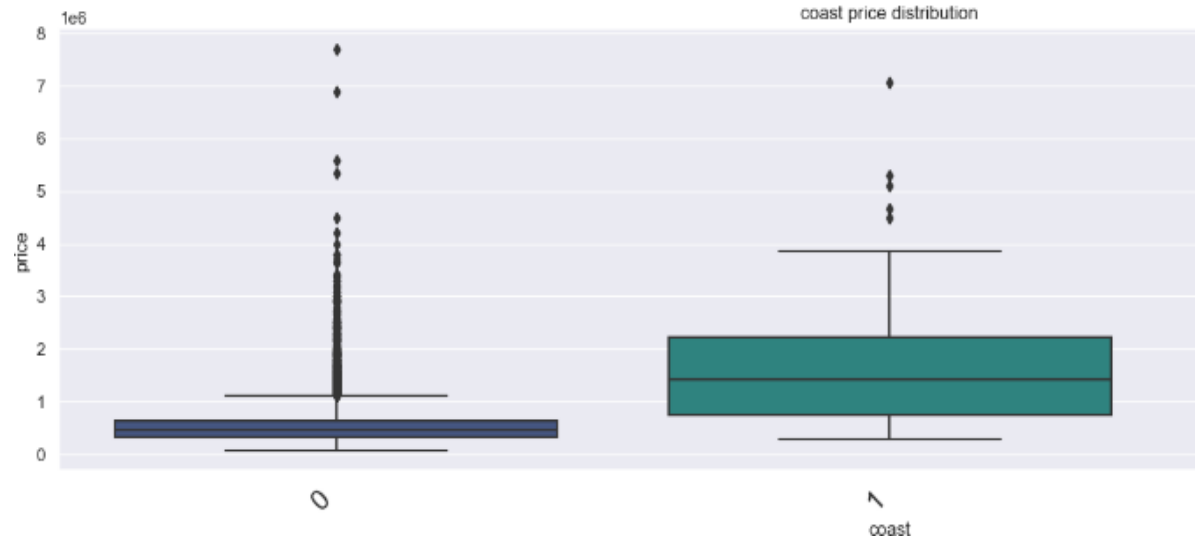


Figure 36 *Boxplot of Price Vs Coast*

Analysis of Price Vs Number of views

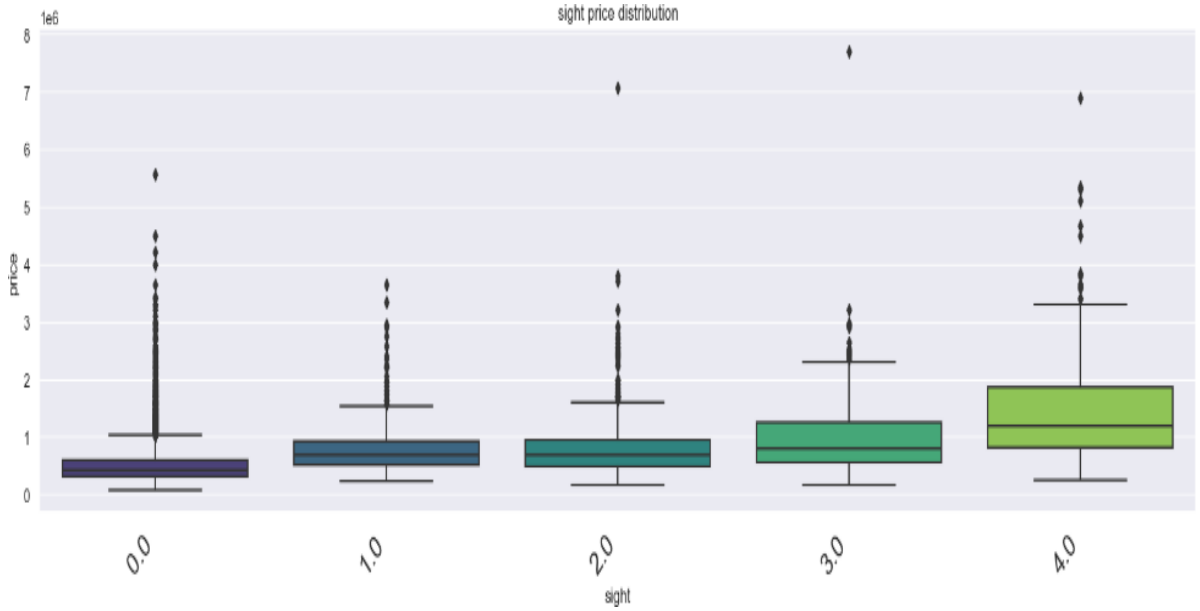


Figure 37 Boxplot of Price Vs Sight

Analysis of Price Vs Condition

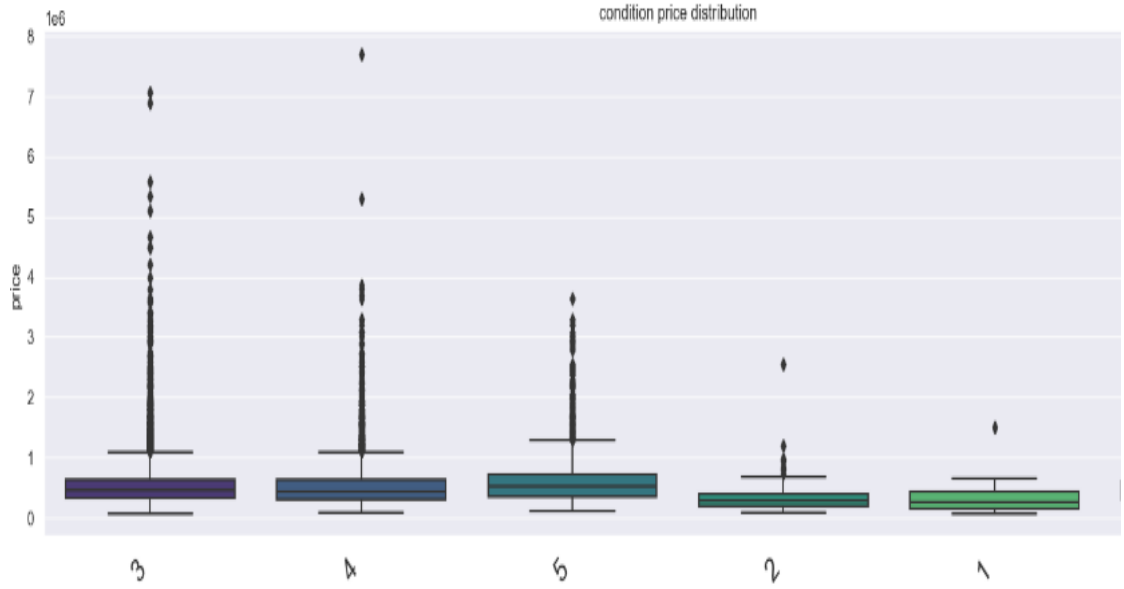


Figure 38 Boxplot of Price Vs Condition

Analysis of Price Vs Quality

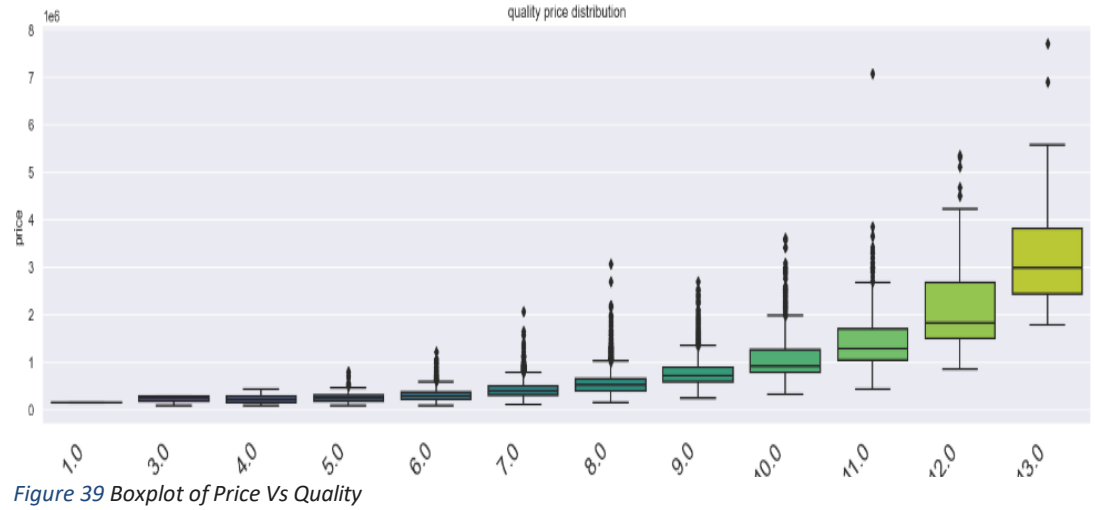


Figure 39 Boxplot of Price Vs Quality

Analysis of Price Vs Furnished

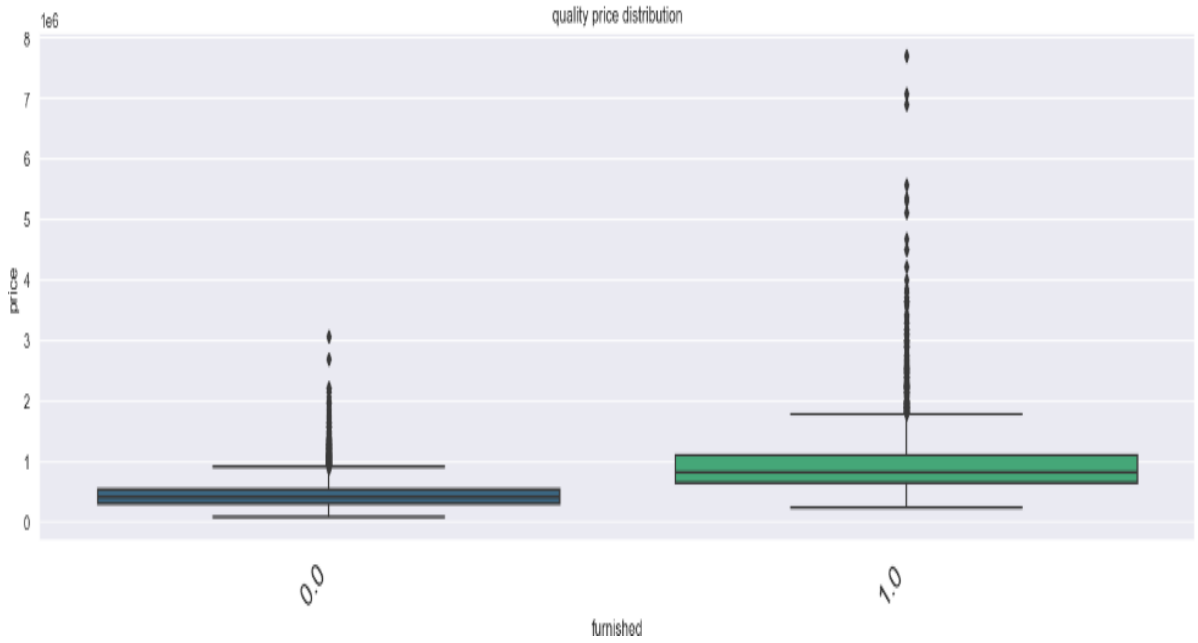


Figure 40 Boxplot of Price Vs Furnished

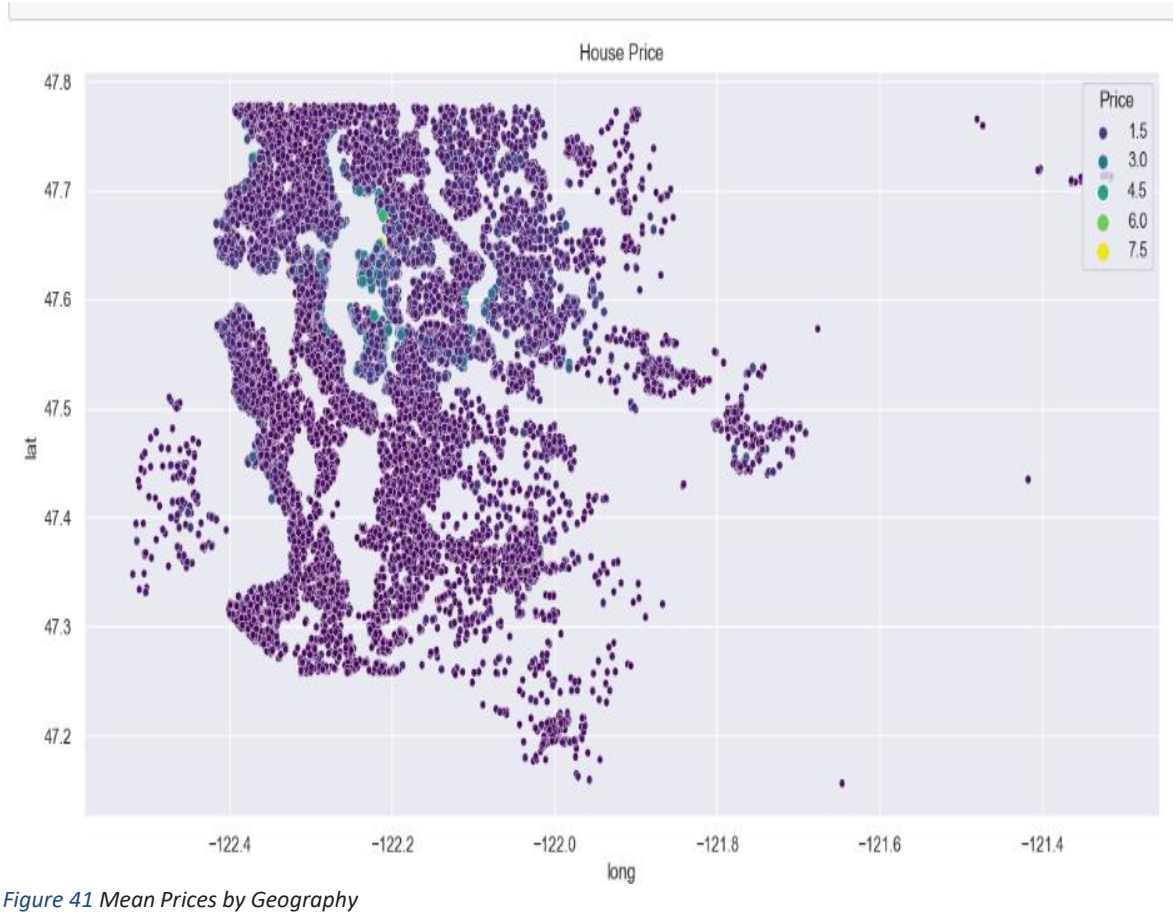


Figure 41 Mean Prices by Geography

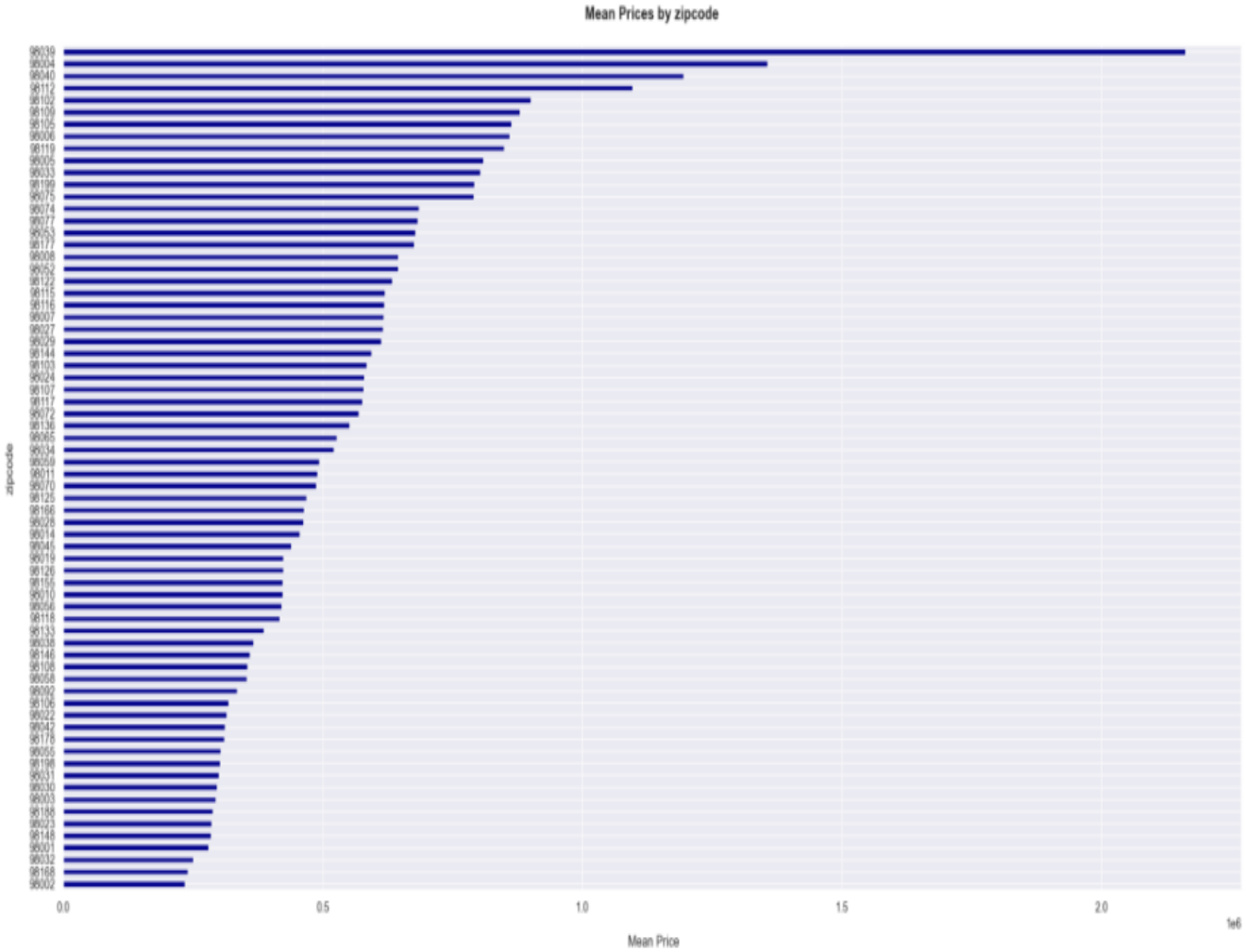


Figure 42 *Mean Prices by Zipcode*

Relationship between the Price and Numerical variables

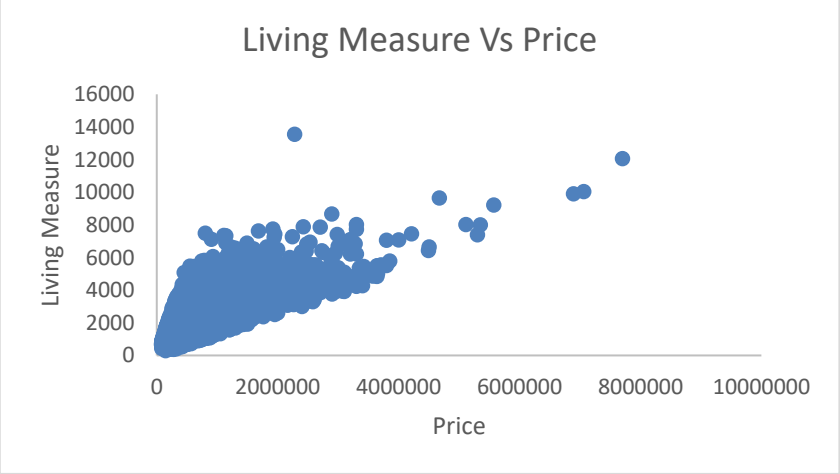


Figure 43 *Scatterplot of Price Vs Living Measure*

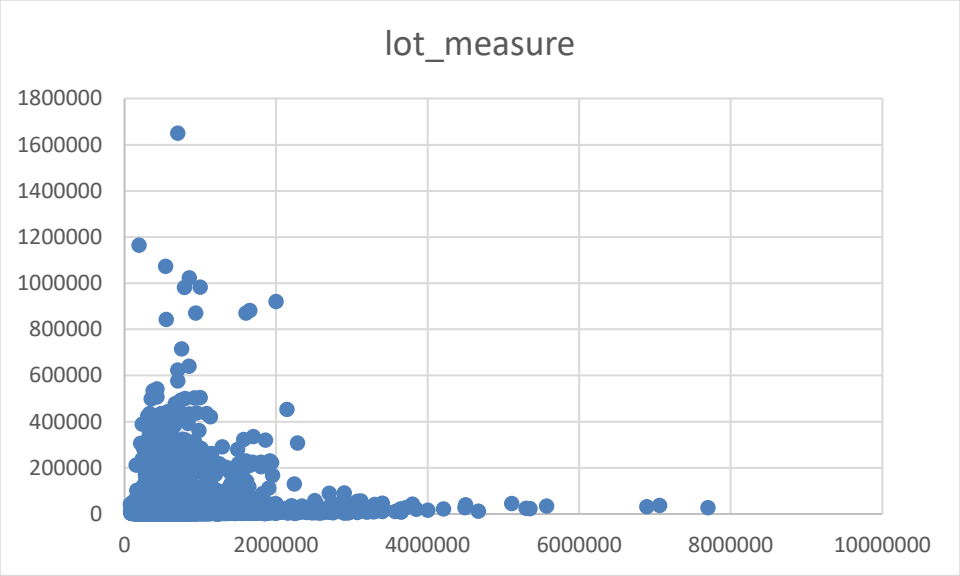


Figure 44 Scatterplot of Price Vs Lot Measure

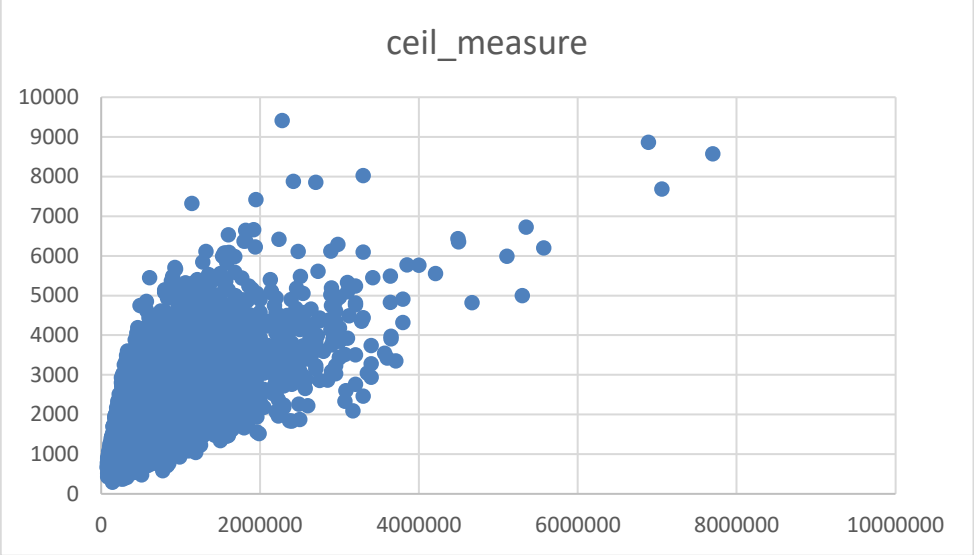


Figure 45 Scatterplot of Price Vs Ceil Measure

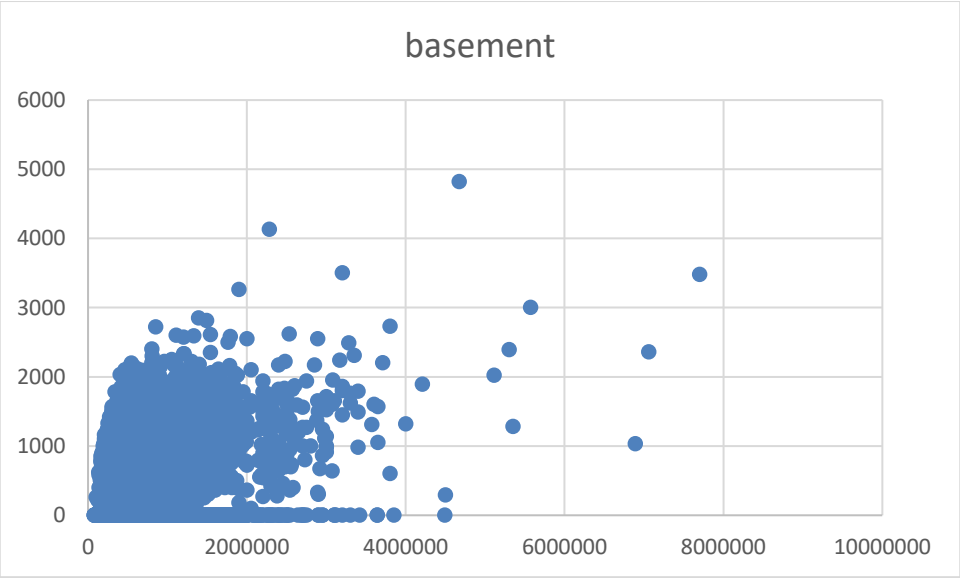


Figure 46 Scatterplot of Price Vs Basement Measure

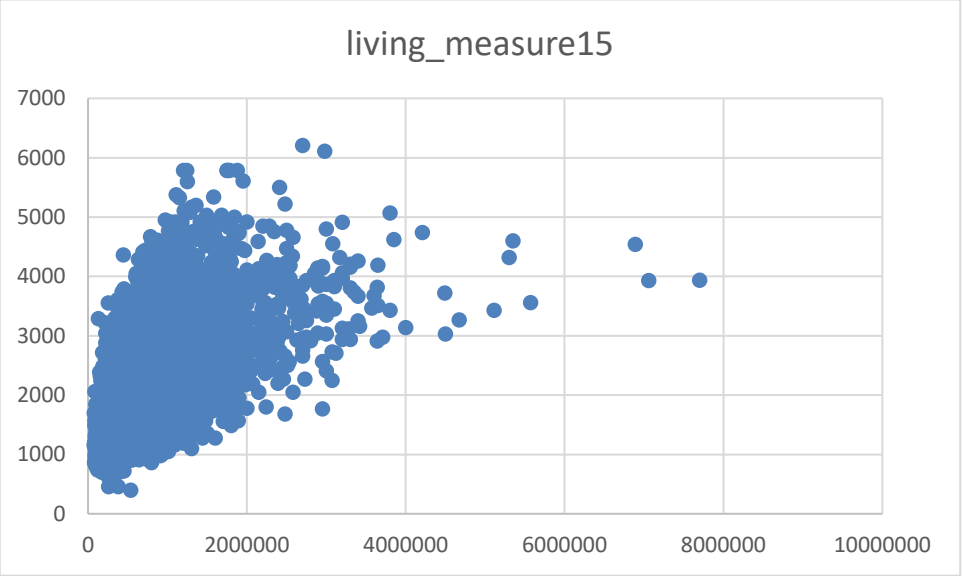


Figure 47 Scatterplot of Price Vs Living Measure 15

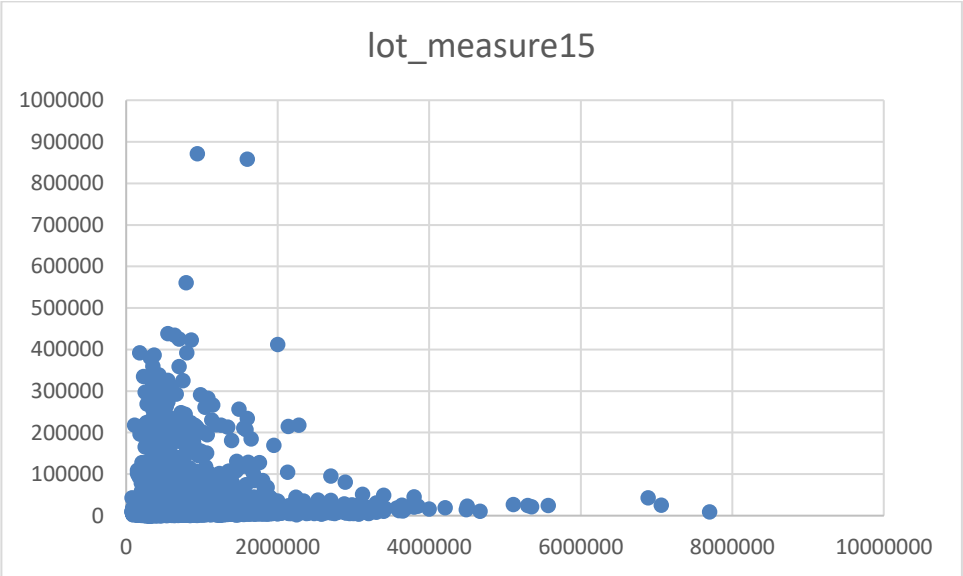


Figure 48 Scatterplot of Price Vs Lot Measure15

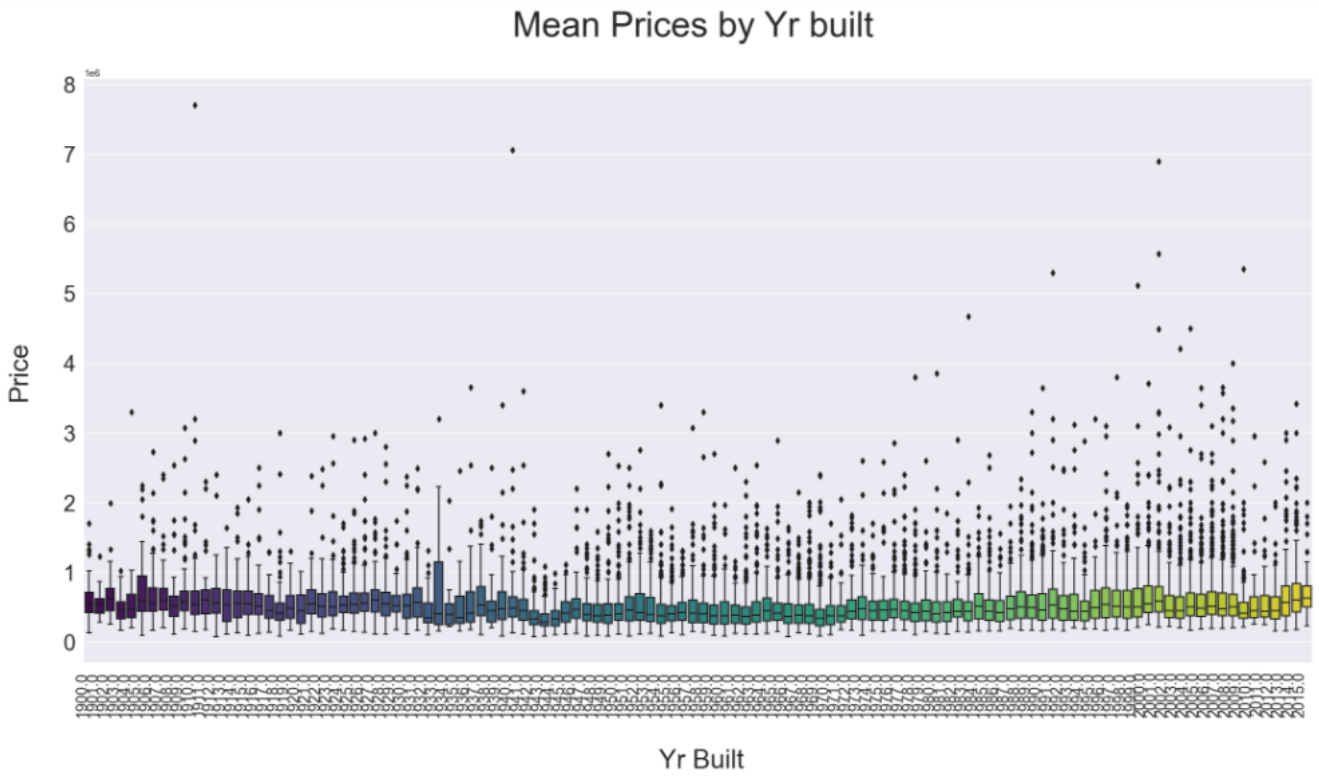


Figure 49 Year Built and Prices

Relationship between the categorical variables

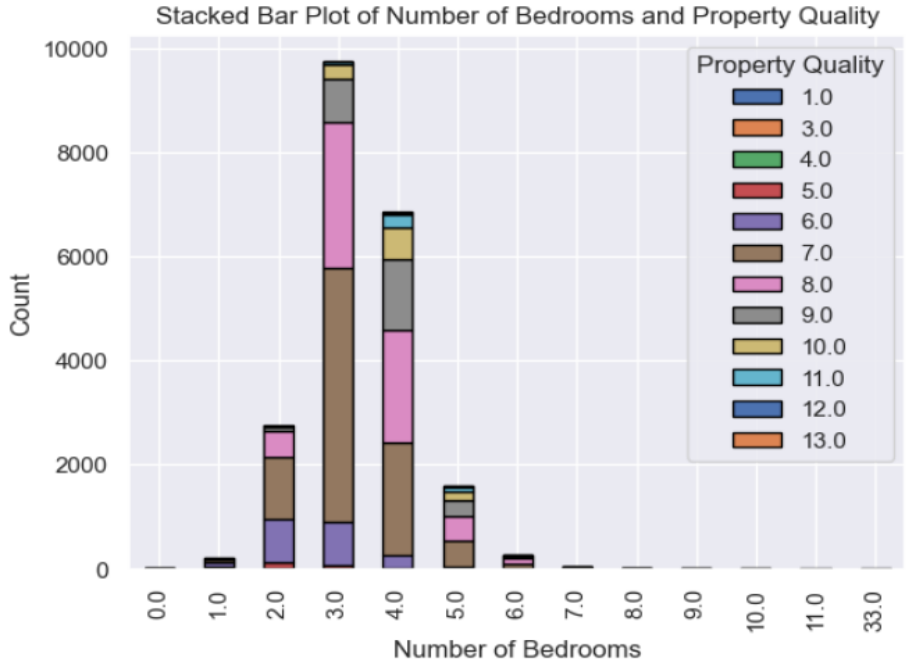


Figure 50 L-Stacked Bar plot of bedrooms Vs Quality

Figure 51 R- Stacked Bar plot of Condition Vs Quality

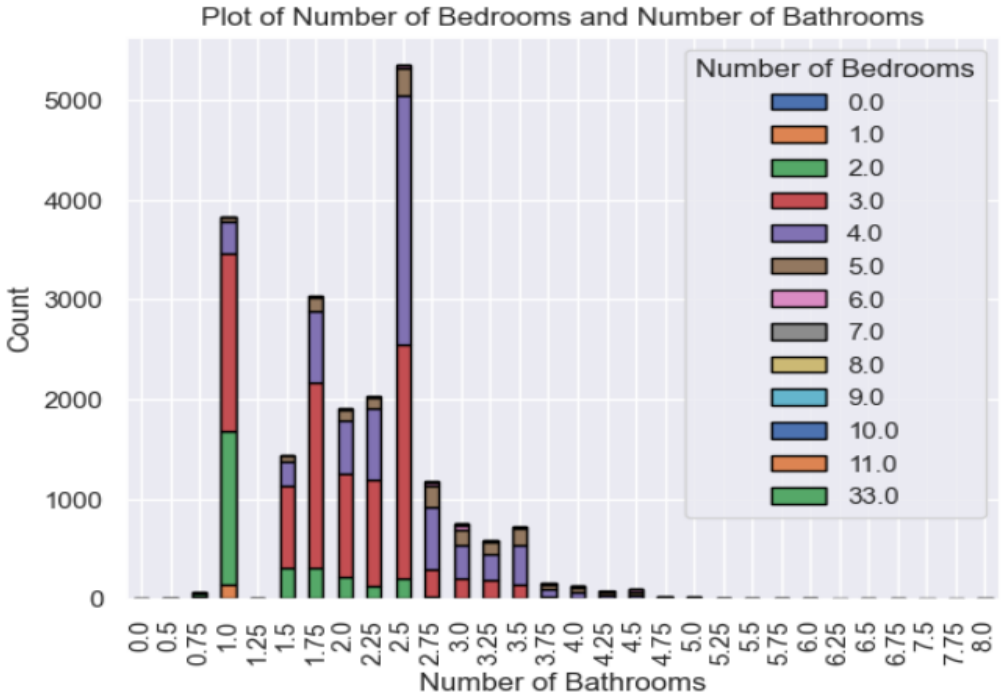


Figure 52 Stacked Bar plot of Number of Bedrooms Vs Bathrooms

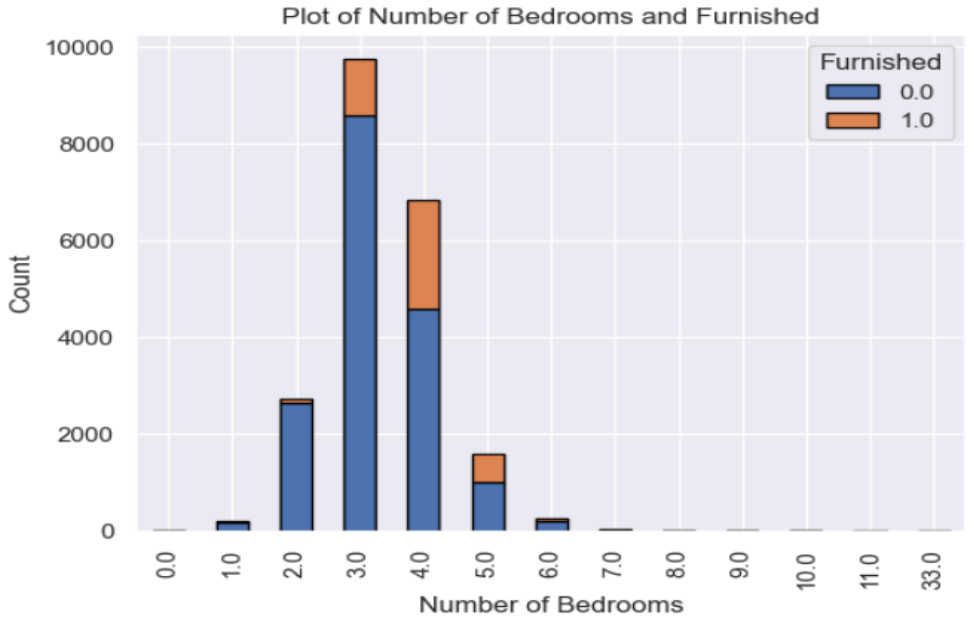


Figure 53 Stacked Bar plot of Number of Bedrooms Vs Furnished

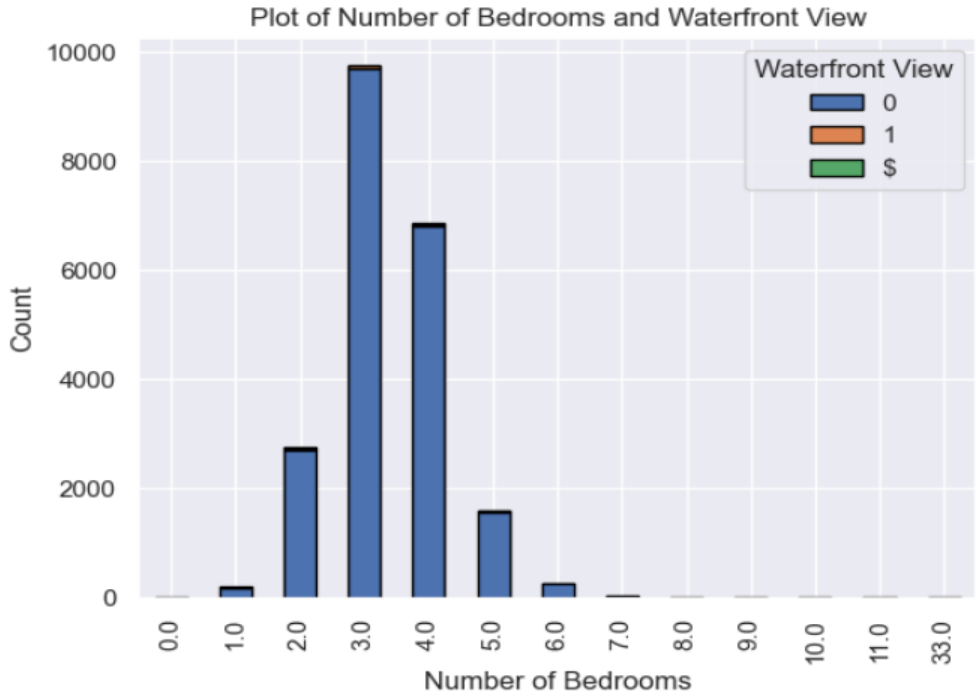


Figure 54 Stacked Bar plot of Number of Bedrooms Vs waterfront views

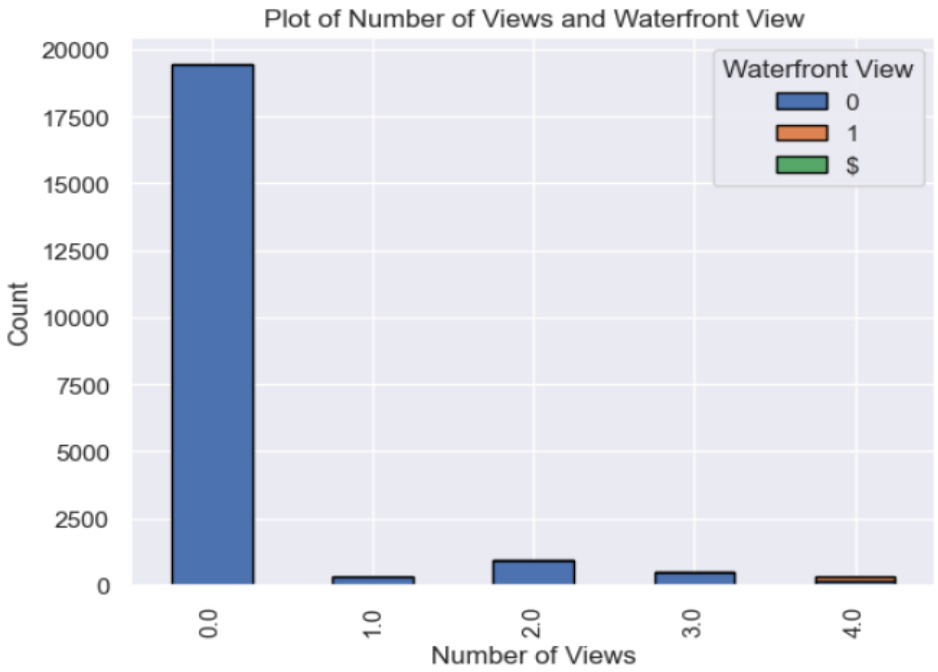


Figure 55 Stacked Bar plot of Sight Vs waterfront views

Multi-variate Analysis:

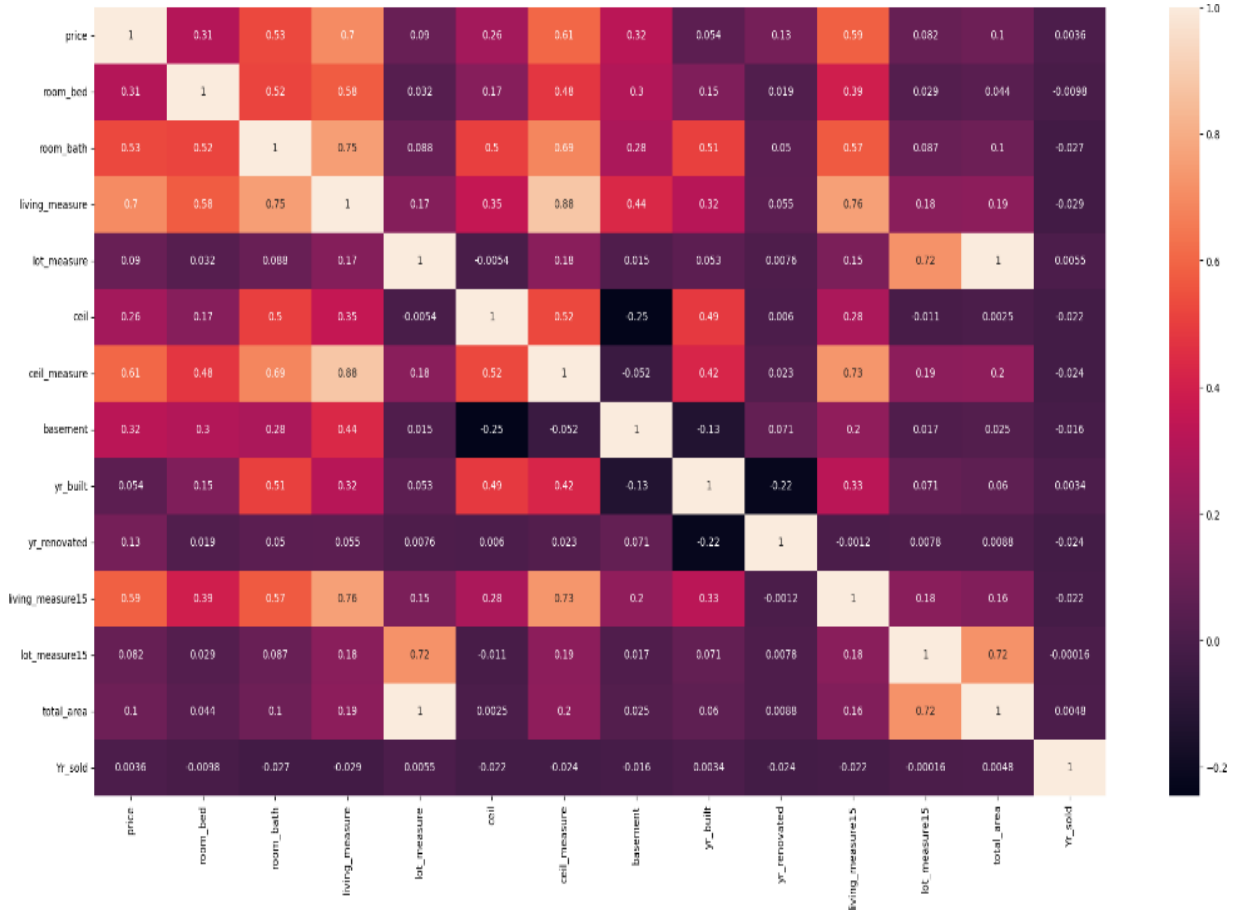


Figure 56 Correlation Plot between numerical variables

Let's see the combined effect of Zip code and Number of Bedrooms and Bathrooms on Price and of Quality and

Condition on Mean Price

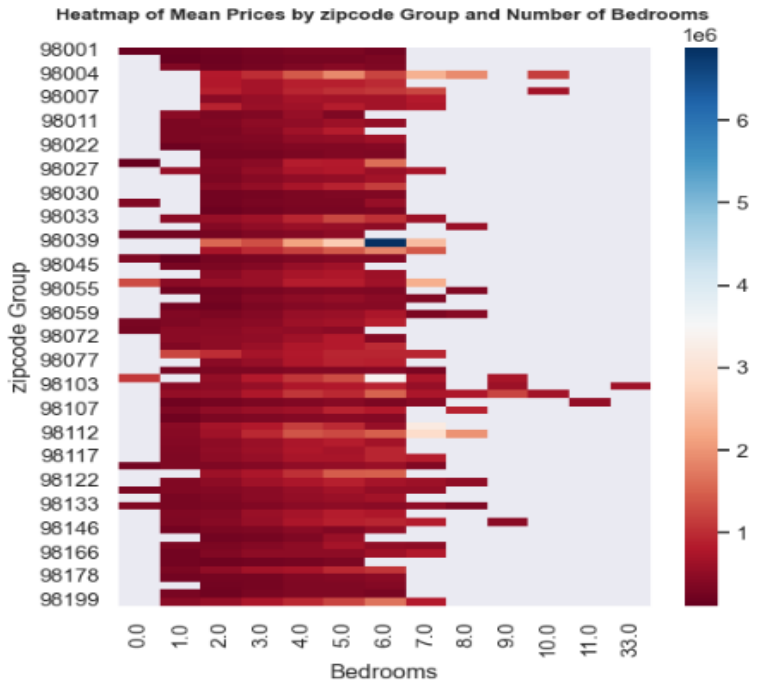


Figure 57 Zip code VS No of Bedrooms with Price

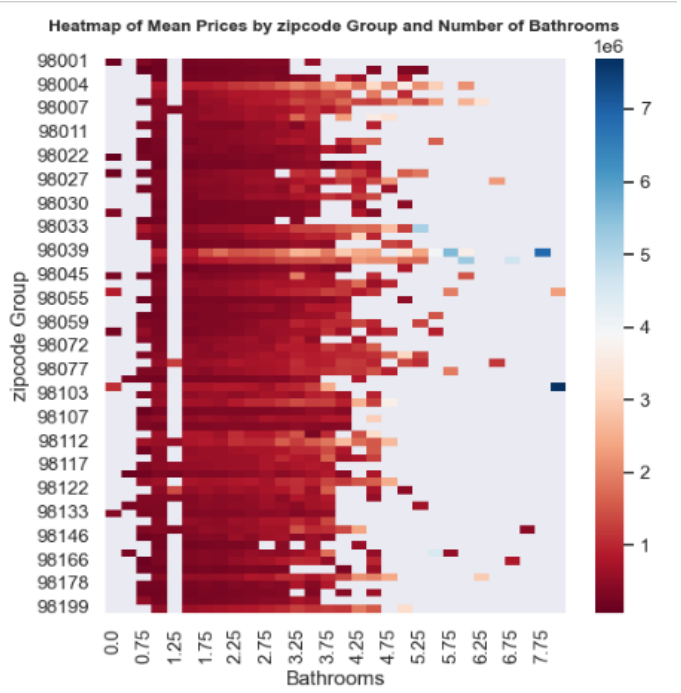


Figure 58 Zip code VS No of Bathrooms with mean prices

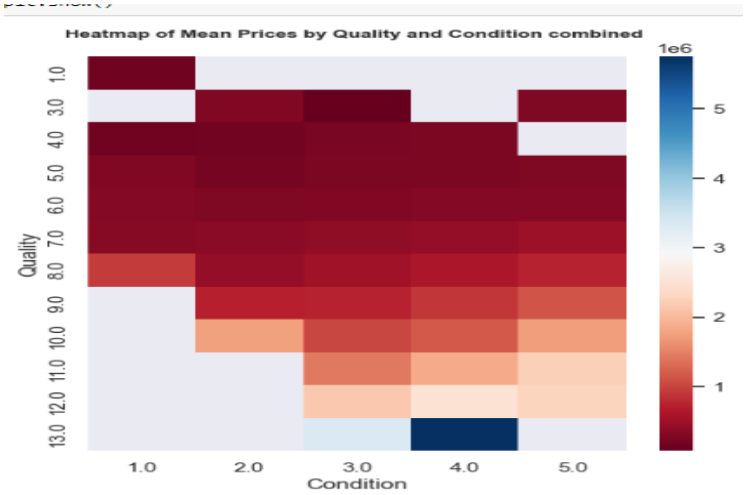


Figure 59 Quality VS Condition with price

Appendix C

OLS Model Building Steps

Steps in building OLS model

- Define the features and target variable(price in this case).
- Split the dataset into training and testing sets.
- We add a constant term to the predictor variable.
- We create an OLS model using sm.OLS.
- We fit the model on Train data
- We print a summary of the regression results using results. summary ().
- Make predictions on the test set and evaluate the model using metrics like Root Mean Squared Error R-squared and (RMSE).

The summary includes statistics such as coefficients, standard errors, t-statistics, and p-values, which are useful for interpreting the regression model.

Let's now build the model with all features in the train set. Here is summary of the regression results.

OLS Regression Results

Dep. Variable:	price	R-squared:	0.881
Model:	OLS	Adj. R-squared:	0.880
Method:	Least Squares	F-statistic:	1281.
Date:	Fri, 26 Jan 2024	Prob (F-statistic):	0.00
Time:	20:50:49	Log-Likelihood:	4294.1
No. Observations:	15129	AIC:	-8412.
Df Residuals:	15041	BIC:	-7741.
Df Model:	87		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	13.0482	0.001	8784.015	0.000	13.045	13.051
room_bed	-0.0060	0.002	-2.978	0.003	-0.010	-0.002
room_bath	0.0337	0.003	12.213	0.000	0.028	0.039
living_measure	0.0982	0.006	16.019	0.000	0.086	0.110
lot_measure	0.0281	0.004	7.573	0.000	0.021	0.035
ceil	-0.0148	0.002	-6.180	0.000	-0.020	-0.010
coast	0.0395	0.002	24.044	0.000	0.036	0.043
sight	0.0462	0.002	25.542	0.000	0.043	0.050
condition	0.0340	0.002	19.856	0.000	0.031	0.037
quality	0.1068	0.003	31.825	0.000	0.100	0.113
ceil_measure	0.0914	0.006	14.996	0.000	0.079	0.103
lat	0.0748	0.012	6.233	0.000	0.051	0.098
long	-0.0594	0.008	-7.049	0.000	-0.076	-0.043
living_measure15	0.0494	0.003	17.905	0.000	0.044	0.055
lot_measure15	0.0031	0.004	0.790	0.429	-0.005	0.011
furnished	-0.0056	0.003	-2.213	0.027	-0.011	-0.001
years old	0.0121	0.003	4.550	0.000	0.007	0.017
is_renovated	0.0143	0.002	8.848	0.000	0.011	0.017
basement_present	0.0170	0.003	5.951	0.000	0.011	0.023
zipcode_98002	0.0018	0.002	0.944	0.345	-0.002	0.005
zipcode_98003	0.0003	0.002	0.166	0.868	-0.004	0.004
zipcode_98004	0.1162	0.004	30.293	0.000	0.109	0.124
zipcode_98005	0.0500	0.003	16.985	0.000	0.044	0.056
zipcode_98006	0.0809	0.004	19.756	0.000	0.073	0.089
zipcode_98007	0.0437	0.003	15.619	0.000	0.038	0.049
zipcode_98008	0.0641	0.004	16.979	0.000	0.057	0.072
zipcode_98010	0.0240	0.002	11.866	0.000	0.020	0.028
zipcode_98011	0.0226	0.004	5.559	0.000	0.015	0.031
zipcode_98014	0.0243	0.004	6.632	0.000	0.017	0.031
zipcode_98019	0.0222	0.004	5.075	0.000	0.014	0.031
zipcode_98022	0.0235	0.003	8.817	0.000	0.018	0.029
zipcode_98023	-0.0096	0.002	-3.938	0.000	-0.014	-0.005

Table 27 OLS Mode Summary 1

We see that R-squared value is 0.88. which means 88 % of the variance is explained by this model. Let's look at the variance inflation factor to check which variables have very high multicollinearity. Multicollinearity exists when there is a correlation between multiple independent variables in a multiple regression model. This can adversely affect the regression results. Multicollinearity can be a problem in a regression model when using algorithms such as OLS (ordinary least squares) in stats models. This is because the estimated regression coefficients become unstable and difficult to interpret in the presence of multicollinearity. There are several ways to detect multicollinearity in a dataset, such as using the Variance Inflation Factor (VIF) or calculating the correlation matrix of the independent variables. VIF (Variance Inflation Factors) is a measure of the amount of multicollinearity in regression analysis. Let's see the VIF values of the predictors

```
VIF values:

lat              65.33
long             32.19
zipcode_98034    17.21
living_measure   17.02
ceil_measure     16.84
                  ...
condition         1.33
zipcode_98148     1.26
coast             1.23
is_renovated      1.18
const             1.00
Length: 88, dtype: float64
```

Table 28 VIF values

VIF starts at 1 and has no upper limit

- VIF = 1, no correlation between the independent variable and the other variables
- VIF exceeding 5 indicates high multicollinearity between this independent variable and the others.

As can be seen there are variables with VIF factor greater than 5. So, let's drop the variables one by one and see the impact on the R-squared.

After dropping 'Lat', 'long', 'living_measure', 'lot_measure15','ceil_measure' which had very high VIF, we have the following predictors with VIF values less than 5

```
VIF values:

quality           4.73
years old         3.12
room_bath         3.04
living_measure15  2.95
furnished         2.85
                  ...
coast             1.22
zipcode_98039     1.18
is_renovated      1.18
zipcode_98148     1.13
const             1.00
Length: 83, dtype: float64
```

Table 29 VIF values after dropping

R-squared values after dropping variables has reduced to 0.86 from 0.88

Now since all variables have VIF less than 5 we do not have multicollinearity in our data, the p-values of the coefficients have become reliable and we can remove the non-significant predictor variables. Non-significant predictors are having p-value >0.05.

Examining significance of multiple regression model

It is not enough to merely fit a multiple regression model to the data; it is necessary to check whether all regression coefficients are significant or not. Significance here means whether the population regression parameters are significantly different from zero. For the j-th slope parameter, the null hypothesis of interest is: H0: $\beta j = 0$ and alternate hypothesizes H1: $\beta j \neq 0$. This test is done separately for each regression coefficient, including the parameters corresponding to the dummy variables.

OLS Regression Results

Dep. Variable:	price	R-squared:	0.860
Model:	OLS	Adj. R-squared:	0.859
Method:	Least Squares	F-statistic:	1129.
Date:	Fri, 26 Jan 2024	Prob (F-statistic):	0.00
Time:	20:52:03	Log-Likelihood:	3070.4
No. Observations:	15129	AIC:	-5975.
Df Residuals:	15046	BIC:	-5342.
Df Model:	82		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	13.0482	0.002	8102.832	0.000	13.045	13.051
room_bed	0.0309	0.002	15.294	0.000	0.027	0.035
room_bath	0.0781	0.003	27.790	0.000	0.073	0.084
lot_measure	0.0464	0.002	20.163	0.000	0.042	0.051
ceil	0.0110	0.002	4.484	0.000	0.006	0.016
coast	0.0405	0.002	22.786	0.000	0.037	0.044
sight	0.0450	0.002	23.104	0.000	0.041	0.049
condition	0.0328	0.002	17.811	0.000	0.029	0.036
quality	0.1531	0.004	43.731	0.000	0.146	0.160
living_measure15	0.1005	0.003	36.324	0.000	0.095	0.106
furnished	0.0119	0.003	4.392	0.000	0.007	0.017
years old	0.0310	0.003	10.887	0.000	0.025	0.037
is_renovated	0.0152	0.002	8.716	0.000	0.012	0.019
basement_present	0.0079	0.002	3.930	0.000	0.004	0.012
zipcode_98002	0.0020	0.002	0.981	0.326	-0.002	0.006
zipcode_98003	0.0013	0.002	0.595	0.552	-0.003	0.005
zipcode_98004	0.1310	0.002	58.692	0.000	0.127	0.135
zipcode_98005	0.0570	0.002	29.221	0.000	0.053	0.061
zipcode_98006	0.0882	0.002	35.581	0.000	0.083	0.093
zipcode_98007	0.0502	0.002	26.498	0.000	0.047	0.054
zipcode_98008	0.0736	0.002	34.209	0.000	0.069	0.078
zipcode_98010	0.0182	0.002	9.971	0.000	0.015	0.022
zipcode_98011	0.0415	0.002	20.883	0.000	0.038	0.045
zipcode_98014	0.0265	0.002	14.021	0.000	0.023	0.030
zipcode_98019	0.0320	0.002	16.071	0.000	0.028	0.036
zipcode_98022	0.0065	0.002	3.139	0.002	0.002	0.011
zipcode_98023	-0.0046	0.002	-1.857	0.063	-0.010	0.000
zipcode_98024	0.0312	0.002	17.457	0.000	0.028	0.035

Table 30 OLS Mode Summary 2

From the above it may be noted that the regression coefficients corresponding to some variables are not statistically significant at level α=0.05. In other words, the regression coefficients corresponding to these variables are not significantly different from 0 in the population. Hence, they may be eliminated from the multiple regression model.

Let's eliminate the variables one by one and check if there is any significant change in R-squared values of 0.86

Let's start by eliminating variable 'zipcode_98003' which has a p value 0.552 and build the model

OLS Regression Results

Dep. Variable:	price	R-squared:	0.860
Model:	OLS	Adj. R-squared:	0.859
Method:	Least Squares	F-statistic:	1143.
Date:	Fri, 26 Jan 2024	Prob (F-statistic):	0.00
Time:	20:52:03	Log-Likelihood:	3070.2
No. Observations:	15129	AIC:	-5976.
Df Residuals:	15047	BIC:	-5351.
Df Model:	81		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	13.0482	0.002	8103.006	0.000	13.045	13.051
room_bed	0.0309	0.002	15.288	0.000	0.027	0.035
room_bath	0.0781	0.003	27.798	0.000	0.073	0.084
lot_measure	0.0464	0.002	20.155	0.000	0.042	0.051
ceil	0.0109	0.002	4.473	0.000	0.006	0.016
coast	0.0405	0.002	22.784	0.000	0.037	0.044
sight	0.0450	0.002	23.110	0.000	0.041	0.049
condition	0.0328	0.002	17.808	0.000	0.029	0.036
quality	0.1532	0.003	43.801	0.000	0.146	0.160
living_measure15	0.1005	0.003	36.328	0.000	0.095	0.106
furnished	0.0119	0.003	4.374	0.000	0.007	0.017
years old	0.0310	0.003	10.908	0.000	0.025	0.037
is_renovated	0.0152	0.002	8.711	0.000	0.012	0.019
basement_present	0.0079	0.002	3.937	0.000	0.004	0.012
zipcode_98002	0.0015	0.002	0.812	0.417	-0.002	0.005
zipcode_98004	0.1304	0.002	65.616	0.000	0.126	0.134
zipcode_98005	0.0566	0.002	31.215	0.000	0.053	0.060
zipcode_98006	0.0875	0.002	40.746	0.000	0.083	0.092
zipcode_98007	0.0498	0.002	28.092	0.000	0.046	0.053
zipcode_98008	0.0730	0.002	37.880	0.000	0.069	0.077
zipcode_98010	0.0179	0.002	10.287	0.000	0.014	0.021
zipcode_98011	0.0411	0.002	22.460	0.000	0.037	0.045
zipcode_98014	0.0261	0.002	14.678	0.000	0.023	0.030
zipcode_98019	0.0315	0.002	17.199	0.000	0.028	0.035
zipcode_98022	0.0060	0.002	3.180	0.001	0.002	0.010
zipcode_98023	-0.0054	0.002	-2.521	0.012	-0.010	-0.001
zipcode_98024	0.0309	0.002	18.024	0.000	0.028	0.034
zipcode_98027	0.0652	0.002	31.829	0.000	0.061	0.069
zipcode_98028	0.0469	0.002	24.350	0.000	0.043	0.051
zipcode_98029	0.0703	0.002	35.586	0.000	0.066	0.074

Table 31 OLS Model summary 3

After eliminating 'zipcode_98003', we can see that R-squared values hasn't changed and its still 0.86 and the regression coefficients corresponding to many are still not statistically significant at level α=0.0 5. Hence, they may be eliminated from the multiple regression model.

Let's eliminate variable 'zipcode_98002' which has a p value 0.417 and build the model.

OLS Regression Results

Dep. Variable:	price	R-squared:	0.860
Model:	OLS	Adj. R-squared:	0.859
Method:	Least Squares	F-statistic:	1157.
Date:	Fri, 26 Jan 2024	Prob (F-statistic):	0.00
Time:	20:52:03	Log-Likelihood:	3069.9
No. Observations:	15129	AIC:	-5978.
Df Residuals:	15048	BIC:	-5360.
Df Model:	80		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	13.0482	0.002	8103.098	0.000	13.045	13.051
room_bed	0.0309	0.002	15.299	0.000	0.027	0.035
room_bath	0.0781	0.003	27.819	0.000	0.073	0.084
lot_measure	0.0463	0.002	20.140	0.000	0.042	0.051
ceil	0.0110	0.002	4.479	0.000	0.006	0.016
coast	0.0405	0.002	22.788	0.000	0.037	0.044
sight	0.0450	0.002	23.115	0.000	0.041	0.049
condition	0.0329	0.002	17.858	0.000	0.029	0.037
quality	0.1531	0.003	43.802	0.000	0.146	0.160
living_measure15	0.1005	0.003	36.319	0.000	0.095	0.106
furnished	0.0120	0.003	4.407	0.000	0.007	0.017
years old	0.0311	0.003	10.929	0.000	0.025	0.037
is_renovated	0.0152	0.002	8.710	0.000	0.012	0.019
basement_present	0.0079	0.002	3.919	0.000	0.004	0.012
zipcode_98004	0.1300	0.002	67.827	0.000	0.126	0.134
zipcode_98005	0.0563	0.002	31.729	0.000	0.053	0.060
zipcode_98006	0.0870	0.002	42.456	0.000	0.083	0.091
zipcode_98007	0.0495	0.002	28.517	0.000	0.046	0.053
zipcode_98008	0.0726	0.002	39.072	0.000	0.069	0.076
zipcode_98010	0.0176	0.002	10.302	0.000	0.014	0.021
zipcode_98011	0.0407	0.002	22.877	0.000	0.037	0.044
zipcode_98014	0.0258	0.002	14.791	0.000	0.022	0.029
zipcode_98019	0.0312	0.002	17.466	0.000	0.028	0.035
zipcode_98022	0.0056	0.002	3.074	0.002	0.002	0.009
zipcode_98023	-0.0059	0.002	-2.934	0.003	-0.010	-0.002
zipcode_98024	0.0307	0.002	18.118	0.000	0.027	0.034
zipcode_98027	0.0647	0.002	32.985	0.000	0.061	0.069
zipcode_98028	0.0465	0.002	25.024	0.000	0.043	0.050
zipcode_98029	0.0699	0.002	36.763	0.000	0.066	0.074
zipcode_98030	0.0066	0.002	3.643	0.000	0.003	0.010
zipcode_98031	0.0089	0.002	4.820	0.000	0.005	0.013
zipcode_98032	-0.0019	0.002	-1.125	0.261	-0.005	0.001
zipcode_98033	0.1089	0.002	55.153	0.000	0.105	0.113
zipcode_98034	0.0835	0.002	40.674	0.000	0.080	0.088

Table 32 OLS model summary 4

After eliminating 'zipcode_98002', we can see that R-squared values hasn't changed and its still 0.86 and from above model that the regression coefficients corresponding to 'zipcode_98032'are still not statistically significant at level α=0.05. Hence, they may be eliminated from the multiple regression n model.
Let's eliminate variable 'zipcode_98032' which has a p value 0.261 and build the model.

OLS Regression Results

Dep. Variable:	price	R-squared:	0.860
Model:	OLS	Adj. R-squared:	0.859
Method:	Least Squares	F-statistic:	1172.
Date:	Fri, 26 Jan 2024	Prob (F-statistic):	0.00
Time:	20:52:04	Log-Likelihood:	3069.3
No. Observations:	15129	AIC:	-5979.
Df Residuals:	15049	BIC:	-5369.
Df Model:	79		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	13.0482	0.002	8103.026	0.000	13.045	13.051
room_bed	0.0309	0.002	15.286	0.000	0.027	0.035
room_bath	0.0782	0.003	27.866	0.000	0.073	0.084
lot_measure	0.0463	0.002	20.152	0.000	0.042	0.051
ceil	0.0109	0.002	4.477	0.000	0.006	0.016
coast	0.0405	0.002	22.785	0.000	0.037	0.044
sight	0.0450	0.002	23.126	0.000	0.041	0.049
condition	0.0329	0.002	17.856	0.000	0.029	0.037
quality	0.1530	0.003	43.789	0.000	0.146	0.160
living_measure15	0.1005	0.003	36.336	0.000	0.095	0.106
furnished	0.0120	0.003	4.427	0.000	0.007	0.017
years old	0.0310	0.003	10.911	0.000	0.025	0.037
is_renovated	0.0152	0.002	8.712	0.000	0.012	0.019
basement_present	0.0078	0.002	3.886	0.000	0.004	0.012
zipcode_98004	0.1304	0.002	69.289	0.000	0.127	0.134
zipcode_98005	0.0566	0.002	32.237	0.000	0.053	0.060
zipcode_98006	0.0875	0.002	43.724	0.000	0.084	0.091
zipcode_98007	0.0498	0.002	28.950	0.000	0.046	0.053
zipcode_98008	0.0730	0.002	39.966	0.000	0.069	0.077
zipcode_98010	0.0179	0.002	10.504	0.000	0.015	0.021
zipcode_98011	0.0411	0.002	23.334	0.000	0.038	0.045
zipcode_98014	0.0261	0.002	15.059	0.000	0.023	0.029
zipcode_98019	0.0315	0.002	17.844	0.000	0.028	0.035
zipcode_98022	0.0059	0.002	3.308	0.001	0.002	0.009
zipcode_98023	-0.0054	0.002	-2.751	0.006	-0.009	-0.002
zipcode_98024	0.0309	0.002	18.339	0.000	0.028	0.034
zipcode_98027	0.0652	0.002	33.924	0.000	0.061	0.069
zipcode_98028	0.0468	0.002	25.667	0.000	0.043	0.050

zipcode_98052	0.0997	0.002	48.945	0.000	0.096	0.104
zipcode_98053	0.0841	0.002	43.610	0.000	0.080	0.088
zipcode_98055	0.0162	0.002	8.894	0.000	0.013	0.020
zipcode_98056	0.0453	0.002	23.753	0.000	0.042	0.049
zipcode_98058	0.0248	0.002	12.838	0.000	0.021	0.029
zipcode_98059	0.0506	0.002	25.724	0.000	0.047	0.054
zipcode_98065	0.0525	0.002	27.930	0.000	0.049	0.056
zipcode_98070	0.0253	0.002	14.629	0.000	0.022	0.029
zipcode_98072	0.0507	0.002	27.727	0.000	0.047	0.054
zipcode_98074	0.0756	0.002	38.462	0.000	0.072	0.079
zipcode_98075	0.0695	0.002	35.675	0.000	0.066	0.073
zipcode_98077	0.0415	0.002	22.804	0.000	0.038	0.045
zipcode_98092	0.0037	0.002	1.957	0.050	-4.97e-06	0.007
zipcode_98102	0.0630	0.002	36.252	0.000	0.060	0.066
zipcode_98103	0.1299	0.002	60.393	0.000	0.126	0.134
zipcode_98105	0.0927	0.002	49.846	0.000	0.089	0.096
zipcode_98106	0.0420	0.002	22.098	0.000	0.038	0.046
zipcode_98107	0.0892	0.002	47.355	0.000	0.086	0.093
zipcode_98108	0.0330	0.002	18.746	0.000	0.030	0.036
zipcode_98109	0.0673	0.002	38.685	0.000	0.064	0.071
zipcode_98112	0.1120	0.002	57.988	0.000	0.108	0.116
zipcode_98115	0.1331	0.002	63.172	0.000	0.129	0.137
zipcode_98116	0.0930	0.002	48.201	0.000	0.089	0.097
zipcode_98117	0.1289	0.002	61.336	0.000	0.125	0.133
zipcode_98118	0.0705	0.002	34.838	0.000	0.067	0.075
zipcode_98119	0.0870	0.002	47.335	0.000	0.083	0.091
zipcode_98122	0.0911	0.002	47.150	0.000	0.087	0.095
zipcode_98125	0.0773	0.002	40.047	0.000	0.074	0.081
zipcode_98126	0.0718	0.002	37.381	0.000	0.068	0.076
zipcode_98133	0.0706	0.002	35.618	0.000	0.067	0.074
zipcode_98136	0.0714	0.002	38.753	0.000	0.068	0.075
zipcode_98144	0.0808	0.002	41.909	0.000	0.077	0.085
zipcode_98146	0.0317	0.002	17.287	0.000	0.028	0.035
zipcode_98148	0.0086	0.002	5.231	0.000	0.005	0.012
zipcode_98155	0.0599	0.002	30.870	0.000	0.056	0.064
zipcode_98166	0.0333	0.002	18.371	0.000	0.030	0.037
zipcode_98168	0.0095	0.002	5.200	0.000	0.006	0.013
zipcode_98177	0.0622	0.002	33.979	0.000	0.059	0.066
zipcode_98178	0.0156	0.002	8.679	0.000	0.012	0.019
zipcode_98188	0.0076	0.002	4.456	0.000	0.004	0.011
zipcode_98198	0.0080	0.002	4.396	0.000	0.004	0.012
zipcode_98199	0.1022	0.002	53.290	0.000	0.098	0.106

Omnibus:	883.773	Durbin-Watson:	2.002
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3412.984
Skew:	-0.160	Prob(JB):	0.00
Kurtosis:	5.305	Cond. No.	9.90

Table 33 OLS Model Summary 5

After eliminating 'zipcode_98032', we can see that R-squared values hasn't changed and its still 0.86 and there are no predictors with p-value>0.05. so, we can stop dropping the variables at this point. So, we have the Final Model which can be used to make predictions on Train and Test Data
Let's see if the assumptions related to residuals are satisfied.

Linearity and Independence of predictors

The residuals vs Fitted values plot is as shown below.

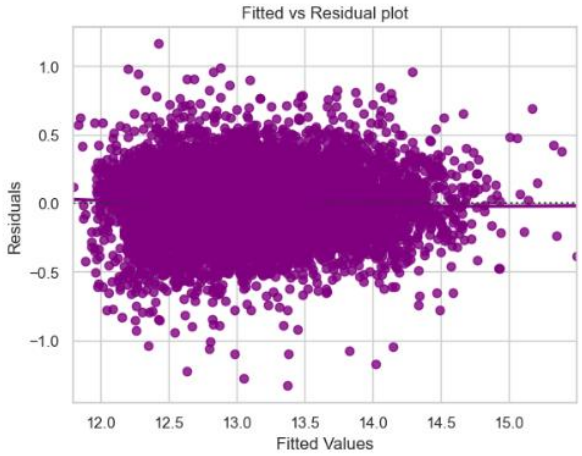


Figure 60 The residuals vs Fitted values plot

There is no pattern in the data thus the assumption of linearity and independence of predictors satisfied
Let's test for Normality of Residuals

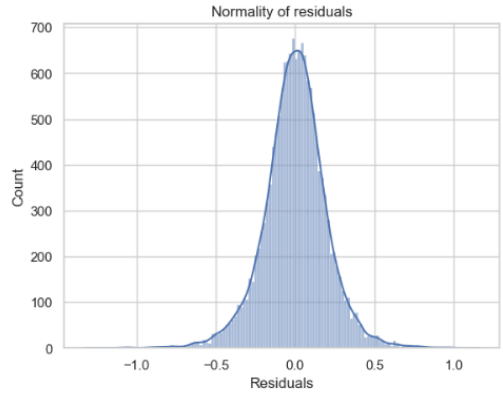


Figure 61 Plotting Density of Residuals

The Shapiro-Wilk test can be used for checking the normality of Residuals.

The null and alternate hypotheses of the test are as follows:

Null hypothesis - Data is normally distributed.
Alternate hypothesis - Data is not normally distributed.

```
ShapiroResult(statistic=0.9792096018791199, pvalue=6.974262456944615e-42)
```

Since p-value < 0.05, we reject the null hypothesis that the residuals are normal as per Shapiro test.
Also, the figure above proves that plot of residuals is not normal

Let's Test for Homoscedasticity using goldfeldquandt
The goldfeldquandt test can be used for checking the Homoscedasticity. The null and alternate hypotheses of the test are as follows:

Null hypothesis: Residuals are homoscedastic
Alternate hypothesis: Residuals have heteroscedasticity

The values are [('F statistic', 0.946740094439005), ('p-value', 0.9910390758884714)]
Since p-value > 0.05, we fail to reject the null hypothesis that the residuals are homoscedastic as per goldfeldquandt test.

Q-Q Plot of residuals with a normal distribution will produce a straight-line plot.

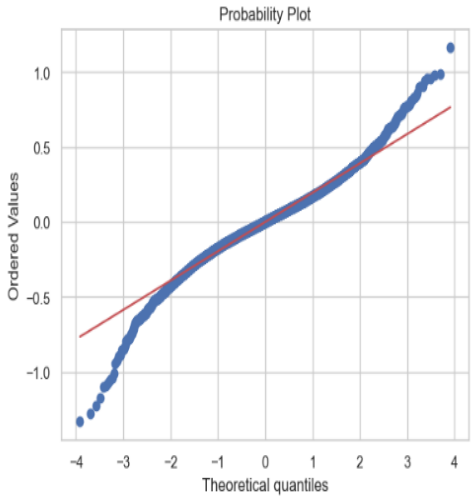


Figure 62 Q_Q plot of Residuals

As we can see there is deviation from the straight line indicating not a very normal distribution

So, some of the assumption for the residuals are satisfied. OLS estimates may be biased if the assumptions are not met, meaning that the estimated coefficients may not accurately reflect the true relationships in the population and can lead to inefficient estimates, resulting in wider confidence intervals and less precise parameter estimates.

Appendix D

To improve the model performance, we use Ensemble modelling technique and model tuning methods like Hyper parameters tuning

Ensemble modelling
Ensemble modelling involves combining multiple machine learning models to improve predictive performance and robustness. For a price prediction problem, where the goal is to predict a continuous variable, ensemble techniques can be particularly effective. Here are some popular ensemble methods for price prediction:

Random Forest:
Description: Random Forest is an ensemble of decision trees. It builds multiple decision trees and merges them together to get a more accurate and stable prediction.
Advantages: Handles non-linearity well. Robust to overfitting. Provides feature importance.
Considerations: The number of trees and tree depth are important hyperparameters to tune.

Gradient Boosting:
Description: Gradient Boosting builds an ensemble of weak learners (typically decision trees) sequentially. It corrects errors made by the previous models to improve prediction accuracy.
Advantages: Can capture complex relationships. Robust to outliers.
Considerations: Sensitive to hyperparameters, such as learning rate and tree depth.
Training time can be longer compared to some other models.

XGBoost (Extreme Gradient Boosting):
Description: XGBoost is an optimized and efficient implementation of gradient boosting. It includes regularization terms to control overfitting and parallel processing to speed up training.
Advantages: High predictive performance. Handles missing values well.
Considerations: Requires careful tuning of hyperparameters. Can be computationally intensive.

ADABoost (Adaptive Boosting):

Description: AdaBoost is an ensemble learning technique that combines weak learners (often decision trees) sequentially to minimize the mean squared error. Instances with larger prediction errors receive higher weights in each iteration.
Advantages: AdaBoost's adaptive learning assigns higher weights to misclassified instances, emphasizing difficult-to-predict cases, leading to improved accuracy in challenging scenarios.
Considerations: AdaBoost is sensitive to noisy data; preprocessing and hyperparameter tuning, especially for the number of estimators and learning rate, are crucial. It is generally less prone to overfitting than individual weak learners.

Bagging:

Description: Bagging combines model outputs using bootstrapping, creating subsets from the dataset with replacement. Utilizing techniques like Random and Stratified Sampling, bagging ensures unbiased data distribution, even with subset sizes smaller than the original dataset. Independent base models are trained on each subset, running in parallel to generate individual predictions, and the final prediction results from merging these independent model predictions.
Advantages: Bagging tends to be less prone to overfitting compared to boosting. This is because each model in the ensemble is trained independently on a different bootstrap sample. In Bagging Models are trained parallelly leading to fast training times
Consideration: The effectiveness of bagging relies on the diversity of the base learners. If the base learners are too similar, the ensemble may not capture a wide range of patterns in the data. It's beneficial to use different hyperparameters, subsets of features, or different types of models to introduce diversity.

Random Forest

Steps in building Random Forest Model

- After splitting dataset and selecting predictors and Target variable we Instantiate RandomForestRegressor regressor by Adjusting hyperparameters, with different hyperparameter values, such as n_estimators, max_depth, etc., to optimize the model's performance. and train the model using the training data. In this case we have max_depth = 10, min_samples_leaf=5, min_samples_split=5
- Use the trained model to make predictions on the test set.
- Assess the model's performance using metrics such as R-squared, RMSE, MSE ,MAPE and MAE on the test set.
- Feature Importance Analysis

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Random Forest	0.8682	0.19	0.0361	0.1369	1.0515	0.9124	0.1564	0.0245	0.1131	0.8688

Table 34 Metrics of Random Forest

- R-square score: - For this RF Model, the R-squared value is 0.87 for test set and 0.91 for train set.
- RMSE: - The RMSE is 0.19 on test dataset and 0.16 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.036 and on train Dataset 0.0245
- Mean Absolute Error for test data using RF Model is 0.1369 and on train dataset 0.1131
- Mean Absolute Percentage Error for test data using RF Model is 1.0515 and on train dataset 0.87

Bagging Using base estimator as Decision Tree

Building a Bagging Decision Tree involves using an ensemble technique called Bootstrap Aggregating (Bagging) with Decision Trees.

Steps in building Bagging Model

- After Train test split, decide on a base model for your ensemble. In the case of Bagging with Decision Trees, the base model a Decision Tree and n_estimators=50. This model is used as the building block for the ensemble.
- Train multiple Decision Trees, each on a different bootstrap sample. This creates an ensemble of diverse models.
- Evaluate the performance of the Bagging Decision Tree ensemble on the testing set. Using R-squared,RMSE, MSE MAE,MAPE etc.

Performance Metrics:

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Bagging with Decision Tree	0.8673	0.1906	0.0363	0.1373	1.0543	0.9117	0.157	0.0246	0.1136	0.8728

Table 35 Metrics of Bagging using base estimator as Decision Tree

- R-square score: - For this Bagging Model with DT, the R-squared value is 0.87 for test set a d 0.91 for train set.
- RMSE: - The RMSE is 0.1906 on test dataset and 0.16 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.037 and on train Dataset is 0.024
- Mean Absolute Error for test data using Bagging Model with DT is 0.1373 and on train dataset is 0.1136
- Mean Absolute Percentage Error for test data using Bagging Model with DT is 1.05 and on train dataset is 0.8728

Bagging Using base estimator as Random Forest

- After Train test split, decide on a base model for your ensemble. In the case of Bagging with Random Forest Trees, the base model is a random Forest, n_estimators=50. This model is used as the building block for the ensemble
- Train multiple Random Forest, each on a different bootstrap sample. This creates an ensemble of

diverse models.

- Evaluate the performance of the Bagging RF ensemble on the testing set using R-squared, RMSE, MSE,MAE,MAPE etc.

Performance Metrics:

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Bagging with Random Forest	0.8707	0.1882	0.0354	0.1353	1.0393	0.905	0.1628	0.0265	0.1167	0.8961

Table 36 Metrics of Bagging Using base estimator as RF

- R-square score: - For this Bagging Model with RF, the R-squared value is 0.87 for test set and 0.90 for train set.
- RMSE: - The RMSE is 0.188 on test dataset and 0.163 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.035 and on train Dataset is 0.026
- Mean Absolute Error for test data using Bagging Model with RF is 0.1353 and on train dataset is 0.1167
- Mean Absolute Error for test data using Bagging Model with RF is 1.039 and on train dataset is 0.896

Gradient Boosting Regressor with default Parameters

Steps in building Gradient Boosting Model

- After Train test split, create an instance of the GradientBoostingRegressor, with default parameters
- Fit the model on the training data. Use the trained Gradient Boost model to make predictions on the test set
- Assess the model's performance using metrics such as R-squared, RMSE, MSE ,MAPE and MAE on the test set

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Gradient Boosting with Default Parameters	0.8727	0.1867	0.0349	0.1373	1.0558	0.8894	0.1757	0.0309	0.1295	0.9951

Table 37 Metrics of Gradient Boosting

Performance Metrics:

- R-square score: - For this Gradient Boosting Model, the R-squared value is 0.87 for test set
- and 0.89 for train set.
- RMSE: - The RMSE is 0.187 on test dataset and 0.175 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.035 and on train Dataset 0.03
- Mean Absolute Error for test data using Gradient Boosting Model is 0.137 and on train dataset is 0.129
- Mean Absolute Percentage Error for test data using Gradient Boosting Model is 1.055 and on train dataset is 0.99

Gradient Boosting Regressor with user defined Parameters

Steps in building Gradient Boosting Model

- After Train test split, create an instance of the GradientBoostingRegressor, specifying hyperparameters as n_estimators=500,max_depth = 10, min_samples_leaf=10,min_samples_split=10 ,learning_rate=0.05,max_features='sqrt',loss='squared_error',random_state=1
- Fit the model on the training data. Use the trained Gradient Boost model to make predictions on the test set
- Assess the model's performance using metrics such as R-squared, RMSE, MSE ,MAPE and MAE on the test set

	Test					Train				
Method	R-squared	RMSE	MSE	MAE	MAPE	R-squared	RMSE	MSE	MAE	MAPE
Gradient Boosting with user Parameters	0.9013	0.1645	0.027	0.1169	0.9013	0.9628	0.102	0.0104	0.0734	0.5667

Table 38 Metrics of Gradient Boosting with Parameters

Performance Metrics:

- R-square score: - For this Gradient Boosting Model with parameters, the R-squared value is 0.9013 for test set
- and 0.9628 for train set.
- RMSE: - The RMSE is 0.1645 on test dataset and 0.1020 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.0270 and on train Dataset 0.0104
- Mean Absolute Error for test data using Gradient Boosting Model is 0.1169 and on train dataset is 0.0734
- Mean Absolute Percentage Error for test data using Gradient Boosting Model is 0.9013 and on train dataset is 0.5667

ADA Boosting Regressor

- After Train test split, choose a base model, such as a Decision Tree and create an instance of it. Create an instance of the AdaBoostRegressor, specifying the base model and other hyperparameters as n_estimators=50,base_estimator=RandomForestRegressor(max_depth = 10, min_samples_leaf=5,min_samples_split=5) ,learning_rate=1,loss='square',random_state=1
- Fit the model on the training data. Use the trained AdaBoost model to make predictions on the test set
- Assess the model's performance using metrics such as R-squared, RMSE, MSE ,MAPE and MAE on the test set

  Performance Metrics:

	Test					Train				
Method	R-squared	RMSE	MSE	MAE	MAPE	R-squared	RMSE	MSE	MAE	MAPE
ADA Boosting	0.8791	0.1819	0.0331	0.1357	1.0429	0.9312	0.1386	0.0192	0.1135	0.8719

Table 39 Metrics of ADA Boosting

- R-square score: - For this ADA Boosting Model, the R-squared value is 0.88 for test set and 0.93 for train set.
- RMSE: - The RMSE is 0.182 on test dataset and 0.14 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.033 and on train Dataset is 0.02
- Mean Absolute Error for test data using ADA Boosting Model is 0.135 and on train dataset is 0.113
- Mean Absolute Percentage Error for test data using ADA Boosting Model is 1.043 and on train dataset is 0.87

XG Boosting Regressor with default parameters

- After Train test split, Create an instance of the XGBoost Regressor with default hyperparameters.
- Fit the model on the training data. Use the trained AdaBoost model to make predictions on the test set
- Assess the model's performance using metrics such as R-squared, RMSE, MSE ,MAPE and MAE on the test set.

Performance Metrics:

	Test					Train				
Method	R-squared	RMSE	MSE	MAE	MAPE	R-squared	RMSE	MSE	MAE	MAPE
XG Boosting with default Parameters	0.8914	0.1724	0.0297	0.1227	0.9439	0.9557	0.1112	0.0124	0.0822	0.6333

Table 40 Metrics of XG Boosting

- R-square score: - For this XG Boosting Model, the R-squared value is 0.89 for test set and 0. 95 for train set.
- RMSE: - The RMSE is 0.1724 on test dataset and 0.1112 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.029 and on train Dataset is 0.0124

- Mean Absolute Error for test data using XG Boosting Model is 0.122 and on train data set 0.082
- Mean Absolute Percentage Error for test data using XG Boosting Model is 0.94 and on train data set 0.633

XG Boosting Regressor with booster as Dart

- After Train test split, Create an instance of the XGBoost Regressor with hyperparameters booster as booster='dart',verbosity=1, nthread=4,n_estimators=500,max_depth=7, learning_rate=0.07, eval_metric='rmse', rate_drop=0.01,normalize_type='forest'dart '
- Fit the model on the training data. Use the trained XGBoost model to make predictions on the test set
- Assess the model's performance using metrics such as R-squared, RMSE, MSE ,MAPE and MAE on the test set.

Performance Metrics:

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
XG Boosting with Dart	0.9002	0.1653	0.0273	0.1166	0.8983	0.9711	0.0898	0.0081	0.0653	0.5043

Table 41 Metrics of XG Booster with Dart

- R-square score: - For this XG Boosting Dart Model, the R-squared value is 0.90 for test set and 0.97 for train set.
- RMSE: - The RMSE is 0.165 on test dataset and 0.09 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.027 and on train Dataset is 0.008
- Mean Absolute Error for test data using XG Boosting dart Model is 0.116 and on train d dataset is 0.06
- Mean Absolute Percentage Error for test data using XG Boosting dart Model is 0.89 and on train dataset is 0.504

XG Boosting Regressor with booster as gbtree

- After Train test split, Create an instance of the XGBoost Regressor with hyperparameters booster as booster='gbtree',verbosity=1, nthread=4,n_estimators=200,max_depth=5, learning_rate=0.07, eval_metric='rmse', rate_drop=0.01,normalize_type='forest'Fit the model on the training data. Use the trained XGBoost model to make predictions on the test set
- Assess the model's performance using metrics such as R-squared, RMSE, MSE ,MAPE and MAE on the test set.

Performance Metrics:

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
XG Boosting with Gbtree	0.8948	0.1698	0.0288	0.1221	0.94	0.9256	0.1441	0.0208	0.1055	0.8124

Table 42 Metrics of XG Boosting with gbtree

- R-square score: - For this XG Boosting gbtree Model, the R-squared value is 0.89 for test set and 0.92 for train set.
- RMSE: - The RMSE is 0.17 on test dataset and 0.14 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.23 and on train Dataset 0.02
- Mean Absolute Error for test data using XG Boosting gbtree Model is 0.122 and on train dataset is 0.105
- Mean Absolute Percentage Error for test data using XG Boosting gbtree Model is 0.94 and on train dataset is 0.81

Appendix E

Hyperparameter tuning and cross-validation are crucial for optimizing the performance of regression and ensemble models. GridSearchCV and RandomsearchCV are hyperparameter tuning in machine learning. They both perform a search over a hyperparameter space to find the best combination of hyperparameters for a given model. GridSearchCV exhaustively searches over a predefined set of hyperparameter values and Requires specifying a grid of hyperparameter values to explore whereas RandomizedSearchCV randomly samples a fixed number of hyperparameter combinations from the specified hyperparameter space.

Ridge and Lasso are regularization techniques used to prevent overfitting by adding a penalty term to the linear regression objective function

In Ridge and Lasso the hyperparameter alpha controls the strength of the regularization in both Ridge and Lasso. The GridSearchCV object is used to perform a cross-validated grid search, and it returns the best hyperparameters and their corresponding scores. After finding the best hyperparameters, you can use the trained models to make predictions on new data.

GridSearchCV does an exhaustive search over specified parameter values for an estimator

GridSearchCV implements a "fit" and a "score" method. It also implements "score_samples", "predict", "predict_proba", "decision_function", "transform" and "inverse_transform" if they are implemented in the estimator used.

The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid.

Hyper parameter tuning of Ridge Model

Steps in Grid search CV

- In the case of Ridge regression with GridSearchCV, we are using the Ridge regressor as the estimator, and the hyperparameter space to be explored is defined in param_grid. The param_grid dictionary contains different values for the hyperparameter Alpha
- The hyperparameter values that we want to explore are as follows {"alpha":[10,20,30,50,100]}
- Create a GridSearchCV object, providing the Ridge regressor, the parameter grid
- Fit the GridSearchCV object on your training data. This will perform an exhaustive search over the specified hyperparameter values
- After fitting, you can access the best hyperparameters found by the grid search.
- We then retrieve the best estimator (model) with the optimal hyperparameters.
- Use the best model to make predictions on the test set and evaluate its performance.
- The hyperparameter being tuned is the regularization strength alpha in Ridge regression. Th e grid search will perform cross-validation for each alpha value specified in the parameter gri d and select the one that optimizes the specified scoring metric.

The best hyperparameters found by the grid search for Ridge is alpha value 10 .

Performance Metrics with best value for alpha as 10

	Test					Train				
Method	R-squared	RMSE	MSE	MAE	MAPE	R-squared	RMSE	MSE	MAE	MAPE
Ridge with GridsearchCV	0.8750	0.1850	0.0342	0.1366	1.0502	0.8810	0.1822	0.0332	0.1343	1.0318

Table 69 Metrics of Ridge Model with Tuning

- R-square score :- For this Ridge Model with Tuning Model , the R-squared value is 0.875 for test set and 0.88 for train set .
- RMSE :- The RMSE is 0.1850 on test dataset and 0.1822 on train dataset .
- Mean Squared Error: The MSE on test Dataset is 0.0342 and on train Dataset is 0.0332
- Mean Absolute Error for test data using Ridge Model with Tuning is 0.1366 and on train dataset is 0.1343
- Mean Absolute Percentage Error for test data using Ridge Model with Tuning is 1.0502 and on train dataset is 1.0318

Hyper parameter tuning of Lasso Model

Steps in Grid search CV

- In the case of Lasso regression with GridSearchCV, we are using the Lasso regressor as the estimator, and the hyperparameter space to be explored is defined in param_grid. The param_grid dictionary contains different values for the hyperparameter Alpha
- The hyperparameter values that we want to explore are as follows {"alpha":[0.1,0.2,0.5,1]}
- Create a GridSearchCV object, providing the Lasso regressor, the parameter grid
- Fit the GridSearchCV object on your training data. This will perform an exhaustive search over the specified hyperparameter values
- After fitting, you can access the best hyperparameters found by the grid search.
- We then retrieve the best estimator (model) with the optimal hyperparameters.
- Use the best model to make predictions on the test set and evaluate its performance.

- The hyperparameter being tuned is the regularization strength alpha in Lasso regression. Th e grid search will perform cross-validation for each alpha value specified in the parameter gri d and select the one that optimizes the specified scoring metric.

The best hyperparameters found by the grid search for Lasso is alpha value 0.1 .

Performance Metrics with best value for alpha as 0.1

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Lasso with GridsearchCV	0.6352	0.3161	0.0999	0.2460	1.8839	0.6360	0.3187	0.1016	0.2490	1.9062

Table 70 Metrics of Lasso Model with Tuning

- R-square score :- For this Lasso Model with Tuning Model , the R-squared value is 0.635 for test set and 0.636 for train set .
- RMSE :- The RMSE is 0.3161 on test dataset and 0.3187 on train dataset .
- Mean Squared Error: The MSE on test Dataset is 0.0999 and on train Dataset is 0.1016
- Mean Absolute Error for test data using Lasso Model with Tuning is 0.246 and on train dataset is 0.249
- Mean Absolute Percentage Error for test data using Lasso Model with Tuning is 1.8839 and on train dataset is 1.906

Hyper parameter tuning of KNN Model

- In the case of KNN (K-Nearest Neighbors) regression with GridSearchCV, we are using the KNN regressor as the estimator, and the hyperparameter space to be explored is defined in param_grid. The param_grid dictionary contains different values for the hyperparameters 'n_neighbors', 'weights', and 'metric'.

- The hyperparameter values that we want to explore are as follows

  'n_neighbors': [3, 5, 7, 9], # values for the number of neighbors

  'weights': ['uniform', 'distance'], # values for the weight function

  'metric': ['euclidean', 'manhattan'] # values for the distance metric

  }

- Create a GridSearchCV object, providing the KNN regressor, the parameter grid, and the scoring metric you want to optimize (e.g., mean squared error for regression tasks).
- Fit the GridSearchCV object on your training data. This will perform an exhaustive search over the specified hyperparameter values
- After fitting, you can access the best hyperparameters found by the grid search.
- We then retrieve the best estimator (model) with the optimal hyperparameters.
- Use the best model to make predictions on the test set and evaluate its performance.
- The GridSearchCV object takes care of performing cross-validation for each combination of hyperparameters, and the scoring metric specified (in this case, negative mean squared error) is used to select the best combination. Cross-validation helps ensure that the model's performance is not overly dependent on a particular train-test split.

Performance Metrics with best hyper parameters {'metric': 'manhattan', 'n_neighbors': 7, 'weights': distance
}

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
KNN with GridsearchCV	0.8613	0.1949	0.0380	0.1380	1.0595	0.9996	0.0104	0.0001	0.0004	0.0035

Table 71 Metrics of KNN Model with Tuning

- R-square score :- For this KNN Model with Tuning Model , the R-squared value is 0.86 for test set and 1 for train set .
- RMSE :- The RMSE is 0.1949 on test dataset and 0.0104 on train dataset .
- Mean Squared Error: The MSE on test Dataset is 0.0380 and on train Dataset is 0.0001
- Mean Absolute Error for test data using KNN Model with Tuning is 0.1380 and on train dataset is 0.0004
- Mean Absolute Percentage Error for test data using KNN Model with Tuning is 1.0595 and on train dataset is 0.0035

Hyper parameter tuning of Ensemble Models

Hyper parameter tuning of Random Forest Regressor using GridSearchCV

In the case of a Random Forest Regressor, hyperparameters such as the number of trees in the forest (n_estimators), the number of features to consider at every split (max_features), the maximum number of levels in each tree (max_depth), and the minimum number of samples required to be at a leaf node (min_samples_leaf) can significantly impact the model's performance.

- We will define the Hyperparameter Grid by specifying a range of values for each hyperparameter that you want to tune n_estimators can take values from the list [50, 60, 70, 80, 90, 100]. max_features can take values from the list ['auto', 'sqrt']. max_depth can take values from the list [2, 4, 6, 8, 10, 15]. min_samples_leaf can take values from the list [1, 2, 5, 10].
- These values form a grid of possible hyperparameter combinations.
- We then create an instance of the machine learning model you want to tune, in this case, a RandomForestRegressor.
- We then create an instance of GridSearchCV and pass the model, the hyperparameter grid, and other

parameters like the number of cross-validation folds (cv), verbosity level (verbose), and the number of parallel jobs (n_jobs).

- GridSearchCV then performs an exhaustive search over all possible combinations of hyperparameters using cross-validation. For each combination, it trains the model on a subset of the training data and evaluates its performance using cross-validation.
- After evaluating all combinations, GridSearchCV selects the set of hyperparameters that resulted in the best performance based on a scoring metric (specified by the scoring parameter).
- You can access the best hyperparameters using the best_params_ attribute of the fitted GridSearchCV object.
- We can retrain our model using the best hyperparameters on the entire training dataset.
- We then use the best model to make predictions on the test set and evaluate its performance.
- The goal is to find the hyperparameter values that yield the best generalization performance on unseen data. This process helps in optimizing the model and avoiding overfitting or underfitting.

With GridSearchCV we get the params for best model as

{'max_depth': 15, 'max_features': 'auto', 'min_samples_leaf': 1, 'n_estimators': 100}

Let's see Performance Metrics with best parameters

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
RF with GridsearchCV	0.8812	0.1804	0.0325	0.1286	0.9886	0.9688	0.0934	0.0087	0.0689	0.5305

Table 43 Metrics of RF Model with Tuning

- R-square score: - For this RF Model with Tuning, the R-squared value is 0.88 for test set and 0.97 for train set.
- RMSE: - The RMSE is 0.1804 on test dataset and 0.0934 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.0325 and on train Dataset is 0.0087
- Mean Absolute Error for test data using RF Model with Tuning is 0.1286 and on train dataset is 0.0689
- Mean Absolute Percentage Error for test data using RF Model with Tuning is 0.996 and on train dataset is 0.5305

Hyper parameter tuning of random XG Booster Regressor using RandomizedSearchCV

- We create an instance of the XGBoost Regressor with default hyperparameters. The goal is to find the optimal values for these hyperparameters
- parameters = {'learning_rate': [.03, 0.05, .07], 'max_depth': [2,5,7,8,10], 'n_estimators': [50,100,150,200]}
- These values form a grid of possible hyperparameter combinations.
- Here, you've specified a set of hyperparameters and their possible values. The learning rate, max depth, and the number of estimators (trees) are parameters we want to tune.
- **RandomizedsearchCV** is a method for hyperparameter tuning that randomly samples a specified number of combinations from the hyperparameter space. In this case, it will explore 100 different combinations of hyperparameters

- We create an instance of **RandomizedsearchCV** and pass the model hyperparameter grid, and other parameters like the number of cross-validation folds (cv), verbosity level (verbose), and the number of parallel jobs (n_jobs).

- Once the **RandomizedsearchCV** object is configured, you can fit it to the training data During this process, it evaluates different combinations of hyperparameters using cross-validation (as specified by **cv=3**). The best combination of hyperparameters is determined based on the performance metric (e.g., mean squared error) on the validation sets.

With **RandomizedsearchCV** we get the params for best model as
{'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.07}

	Test					Train				
Method	R-squared	RMSE	MSE	MAE	MAPE	R-squared	RMSE	MSE	MAE	MAPE
XGB with RandomSearchCV	0.8967	0.1682	0.0283	0.1192	0.9177	0.9513	0.1166	0.0136	0.0852	0.6568

Table 44 Metrics of XG Booster Model with Random Search CV Tuning

Performance Metrics with best parameters as below:

- R-square score: - For this XG Booster Regressor using RandomizedSearchCV with tuning the R-squared value is 0.896 for test set and 0.951 for train set.
- RMSE: - The RMSE is 0.1682 on test dataset and 0.1166 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.0283 and on train Dataset is 0.0136
- Mean Absolute Error for test data using XG Booster Regressor using RandomsearchCV is 0.1192 and on train dataset is 0.0852
- Mean Absolute Percentage Error for test data using XG Booster Regressor using is 0.9177 and on train dataset is 0.6568

Hyper parameter tuning of random XG Booster Regressor using GridSearchCV

Similar to XGB Regressor we created using **RandomizedSearchCV**, we now create model using **GridSearchCV** with same parameter grid. The best parameters with **GridSearchCV** are

{'learning_rate': 0.07, 'max_depth': 2, 'n_estimators': 3000}

Performance Metrics with best parameters as below

	Test					Train				
Method	R-squared	RMSE	MSE	MAE	MAPE	R-squared	RMSE	MSE	MAE	MAPE
XGB with GridsearchCV	0.9	0.1655	0.0274	0.1185	0.9131	0.9253	0.1444	0.0208	0.1051	0.8093

Table 45 Metrics of XG Booster with Tuning using GridSearchCV

- R-square score: - For this XG Booster Regressor using GridSearchCV, the R-squared value is 0.90 for test set and 0.92 for train set.
- RMSE: - The RMSE is 0.1655 on test dataset and 0.1444 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.0274 and on train Dataset is 0.0208
- Mean Absolute Error: The MAE on test Dataset is 0.1185 and on train dataset is 0.1051
- Mean Absolute Percentage Error: The MAPE on test Dataset is 0.9131 and on train dataset is 0.8093

Hyper parameter tuning of Gradient Booster Regressor using GridSearchCV

- Create an instance of the GradientBoostingRegressor, which will be used as the base model.
- Specify the hyperparameter grid to be searched. The grid contains different values for max_depth, max_features, learning_rate, min_samples_leaf, min_samples_split, and n_estimators as below
- param_grid = {'max_depth': [10,15], 'max_features': ['auto','sqrt'],'learning_rate': [0.05,0.1], 'min_samples_leaf': [10,20], 'min_samples_split': [10,15], 'n_estimators': [250,500,1000],}.
- Create a GridSearchCV object with the GradientBoostingRegressor as the estimator, the specified parameter grid, 3-fold cross-validation (cv=3), 2 parallel jobs (n_jobs=2), and verbose output (verbose=1).
- Perform the grid search with cross-validation to find the best hyperparameters for the model.
- Retrieve the best hyperparameters and the best model from the grid search.
- Use the best model to make predictions on the validation set and calculates the mean squared error

(MSE) to evaluate the model's performance.

- Best parameters are a below
  {'learning_rate': 0.05, 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 20, 'min_samples_split': 10, 'n_estimators': 500}

Performance Metrics with best parameters as below:

	Test					Train				
Method	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**	**R-squared**	**RMSE**	**MSE**	**MAE**	**MAPE**
Gradient Boosting with GridsearchCV	0.9011	0.1646	0.0271	0.1172	0.9029	0.9514	0.1165	0.0136	0.0837	0.6453

Table 46 Metrics of Gradient Booster with Tuning using GridSearchCV

- R-square score: - For this Gradient Booster using GridSearchCV, the R-squared value is
- 0.90 for test set and 0.95 for train set.
- RMSE: - The RMSE is 0.165 on test dataset and 0.1165 on train dataset.
- Mean Squared Error: The MSE on test Dataset is 0.027 and on train Dataset is 0.0136
- Mean Absolute Error for test data using Gradient Booster using GridSearchCV with Tuning is 0.1172 and on train dataset is 0.0837
- Mean Absolute Percentage Error for test data using Gradient Booster using GridSearchCV with Tuning is 0.9029 and on train dataset is 0.6453

Appendix F

Distribution of Target Variable

The target variable of this data set is the "price" as stated in the description of the data set. Figure below shows the histogram of the target variable.

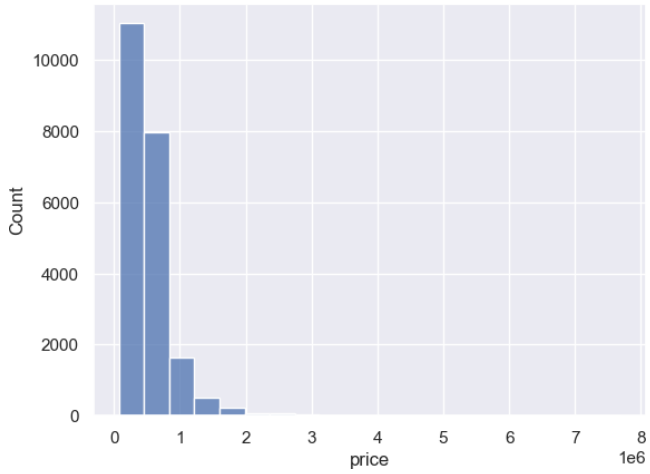


Figure 51: Histogram of target variable price

In the optimal case, the distribution of the target variable would be normally distributed. But, as we can see, this is clearly not the case for the target variable of this data set.

The mean is around 540182.16 and median is 450000.0.There is a large right tail of higher price values. This could lead to the problem, that the model better predicts the price values around the mean, but is quite bad at predicting the price values from the right tail. This is because most of the time the model sees values around the mean and is therefore biased towards these price values.

A statistical test can also be executed in order to verify that this distribution is not a normal distribution. Null hypothesis states that distribution is normal. Alternate hypothesis states that

distribution is not normal. In practice, a p-value smaller than 0.05 means that the null-hypothesis can be rejected. Therefore, a p-value of 0.05 or greater means that the distribution is a normal distribution conversely a p-value less than 0.05 means that the distribution is not a normal distribution.

We conduct test on the target variable and we see that p-value for this statistical test is at 0 and less than 0.05 , hence we reject the null hypothesis that distribution is normal . So, we can conclude the distribution of Price variable is not normal

Transformation of Target Variable

As we saw the target variable is highly skewed. Let's see different transformations that can be applied to the target variable. The goal of these transformations is to get the distribution of the target variable closer to a normal distribution, thus eliminating the imbalance.

Square Root Transformation

- The first transformation is the square root transformation. This transformation simply computes the square root of each entry of the target variable (Equation 1).

$$y_{sqrt} = \sqrt{y}$$

Equation 1: Square root transformation applied on input target variable

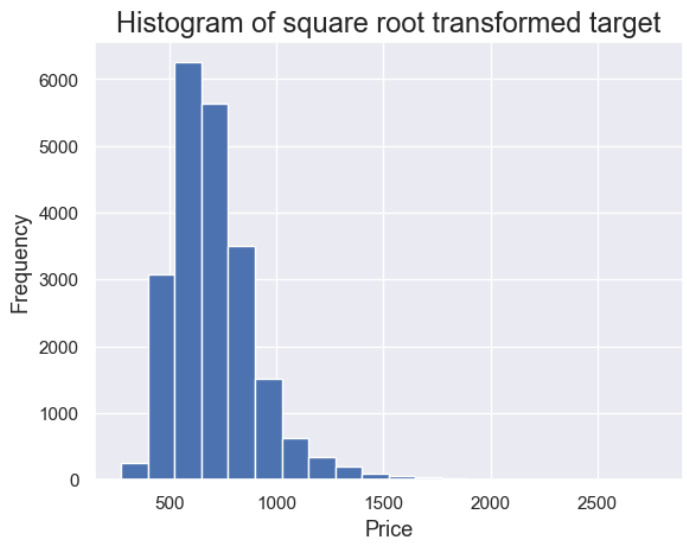


Figure 63 Histogram of Target Variable after square root transformation

Figure above shows the histogram of the target variable after applying this transformation.

The new distribution already looks less skewed, but still not really like a normal distribution.

The p-value of the new distribution is at 0, so still lower than the 0.05.

Log Transformation

Another useful transformation is computing the natural logarithm of each entry of the input target variable (Equation 2).

$$y_{log} = \ln y$$

- *Equation 2:* Log transformation applied on input target variable

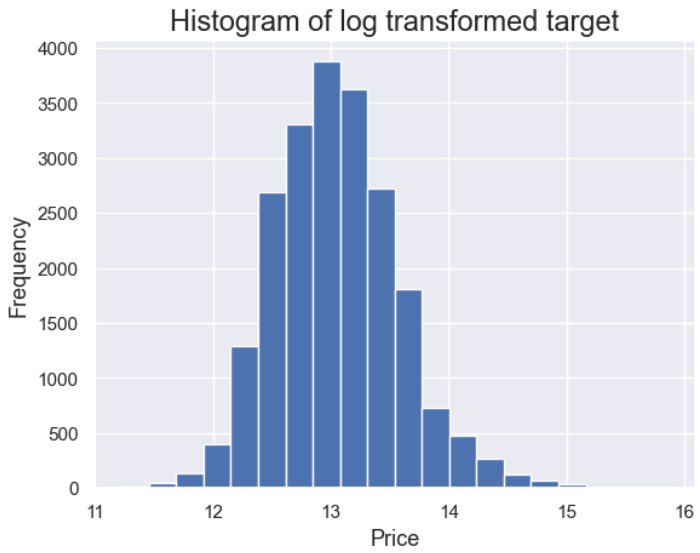


Figure 64 Histogram of target variable after log transformation is applied

- Figure above shows the histogram after the log transformation is applied. This distribution looks better than the original, but still not like a normal distribution. The p-value of this distribution is at 2.308725847410902e-186, which is still lower than 0.05.

Boxcox Transformation

Another useful transformation is Boxcox.

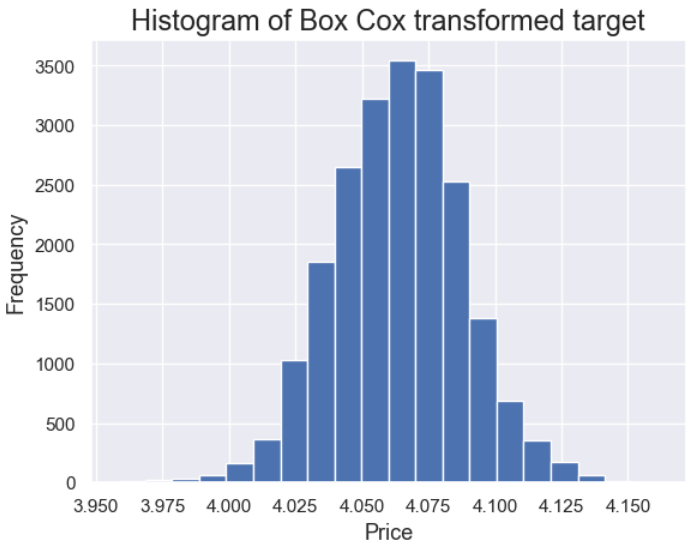


Figure 65 Histogram of target variable after boxcox *transformation is applied*

Figure above shows the histogram after the Boxcox transformation is applied. This distribution looks better than the original, but still not like a normal distribution. The p-value of this distribution is at 1.5422148070630482e-15 which is still lower than 0.05.

We can apply any of these transformation on the target variables and see the performance of the model .

Appendix G

(R-squared):

- R^2 a measure of how well the predicted values from the model matches the actual values.
- It ranges from 0 to 1, where 1 indicates perfect predictions, and 0 indicates that the model does not explain any variability in the target variable.
- R^2 is often interpreted as the proportion of the variance in the dependent variable that is predictable from the independent variables.

Mean Squared Error (MSE):

- MSE is the average of the squared differences between predicted and actual values.
- It gives more weight to larger errors and is sensitive to outliers.
- MSE is calculated as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

where n is the number of data points, yi is the actual value, and y^i is the predicted value.

Root Mean Squared Error (RMSE):

- RMSE is the square root of the MSE and has the same unit as the dependent variable.
- It provides a measure of the average magnitude of errors in the predicted values.
- RMSE is calculated as: RMSE= $\sqrt{\text{MSE}}$

Mean Absolute Error (MAE):

- MAE is the average of the absolute differences between predicted and actual values.
- It is less sensitive to outliers compared to MSE and RMSE.MAE is calculated as:

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

where n is the number of data points, yi is the actual value, and y^i is the predicted value.

- These metrics help in assessing the accuracy and goodness of fit of a regression model. When evaluating a model, it's often useful to consider multiple metrics to gain a comprehensive understanding of its performance. R^2 provides an overall measure of goodness of fit, while MSE, RMSE, and MAE provide insights into the magnitude and direction of errors.

MAPE (Mean Absolute Percentage Error)

- MAPE measures the average absolute percentage difference between predicted and actual values.
- The formula for calculating MAPE is as follows:

where: $MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - P_i}{A_i}\right| \times 100$

n is the number of observations.
Ai is the actual value.
Pi is the predicted value.

- MAPE expresses the error as a percentage of the actual values, making it easy to interpret and compare across different datasets or models. A lower MAPE indicates better accuracy, while a higher MAPE suggests greater prediction errors.

Appendix H

Clustering can help identify distinct segments in the housing market based on various features such as square footage, number of bedrooms, bathrooms, and overall condition. This segmentation can assist in tailoring marketing and pricing strategies for different market segments-means clustering is a popular unsupervised machine learning algorithm used for partitioning a dataset into distinct, non-overlapping groups or clusters. The algorithm works by iteratively
assigning data points to clusters based on their similarity and updating the cluster centroids until convergence. Before performing clustering, we treat the outliers and scale the data because K-means is sensitive to outliers because it uses the mean to calculate cluster centroids. Outliers can disproportionately influence the centroid positions.
We also scale the features before applying K-means to ensure that all features contribute equally to the clustering process.
The number of clusters, K, needs to be specified before running the algorithm. Techniques like the elbow method or silhouette analysis are often employed to determine an optimal K.

Let’s look at the Elbow plot

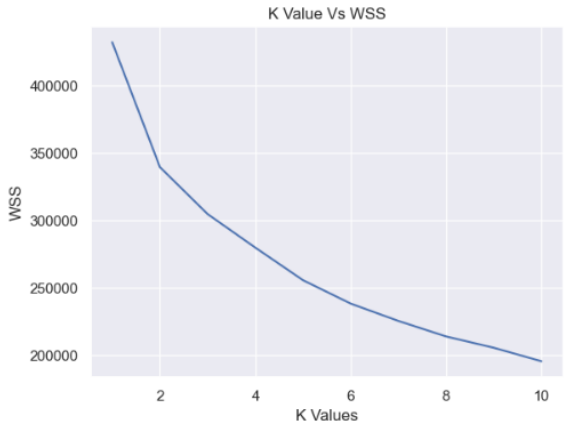


Figure 66 Elbow Plot

Elbow plot for K-means clustering shows a clear elbow at 2 clusters and another potential elbow at 5 clusters
Elbow at 2 Clusters
There is a clear and pronounced elbow at 2 clusters, it suggests that using 2 clusters explains a significant amount of the variance in the data. This simplicity can be advantageous, especially if the two clusters capture meaningful distinctions in your dataset.
Elbow at 5 Clusters
There is a noticeable but less prominent elbow at 5 clusters, it could indicate that adding more clusters provides additional refinement in capturing nuances within the data. However, a larger number of clusters may also introduce complexity and may not necessarily lead to more interpretable or actionable results.

Let’s have a look at the silhouette plot of Silhouette scores versus number of clusters

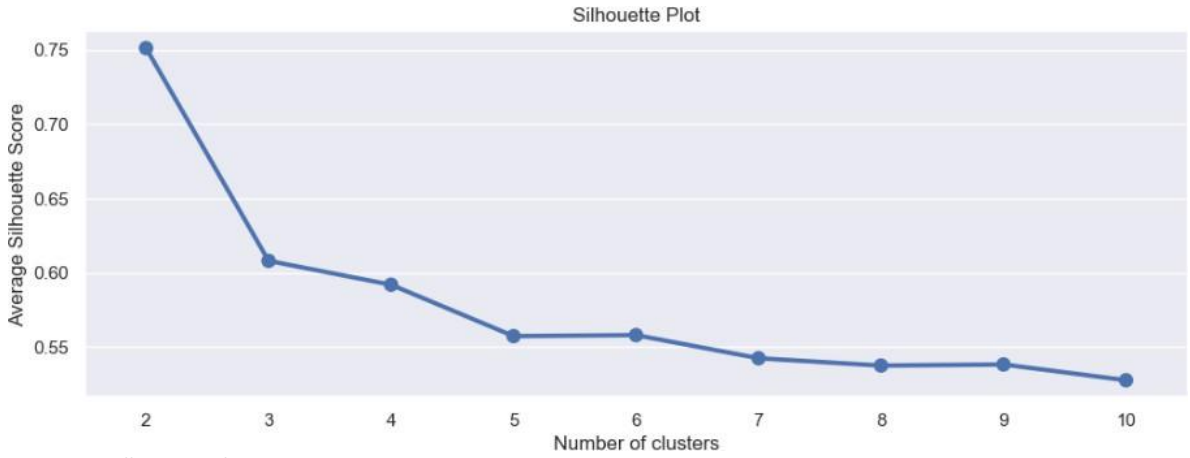


Figure 67 Silhouette plot

Silhouette Score:

let’s calculate the silhouette score for both 2 and 3 clusters. A higher silhouette score generally indicates better-defined clusters

```
The Average Silhouette Score for 2clusters is 0.75087
The Average Silhouette Score for 3clusters is 0.60746
The Average Silhouette Score for 4clusters is 0.59113
The Average Silhouette Score for 5clusters is 0.55596
The Average Silhouette Score for 6clusters is 0.55659
The Average Silhouette Score for 7clusters is 0.54056
The Average Silhouette Score for 8clusters is 0.53507
The Average Silhouette Score for 9clusters is 0.53562
The Average Silhouette Score for 10clusters is 0.52546
```

Silhouette score is slightly higher for 2 clusters, so it supports the choice of 2 clusters. Here's a snapshot of the dataset along with 2 clusters created using K-means clustering.

	0	1	2	3	4
price	600000.00	190000.00	735000.00	257000.00	450000.00
room_bed	4.00	2.00	4.00	3.00	2.00
room_bath	1.75	1.00	2.75	2.50	1.00
living_measure	3050.00	670.00	3040.00	1740.00	1120.00
lot_measure	9440.00	3101.00	2415.00	3721.00	4590.00
ceil	1.00	1.00	2.00	2.00	1.00
coast	0.00	0.00	1.00	0.00	0.00
sight	0.00	0.00	4.00	0.00	0.00
condition	3.00	4.00	3.00	3.00	3.00
quality	8.00	6.00	8.00	8.00	7.00
ceil_measure	1800.00	670.00	3040.00	1740.00	1120.00
zipcode	98034.00	98118.00	98118.00	98002.00	98118.00
lat	47.72	47.55	47.52	47.34	47.57
long	-122.18	-122.27	-122.26	-122.21	-122.28
living_measure15	2020.00	1660.00	2620.00	2030.00	1120.00
lot_measure15	8660.00	4100.00	2433.00	3794.00	5100.00
furnished	0.00	0.00	0.00	0.00	0.00
years old	49.00	67.00	48.00	5.00	91.00
is_renovated	0.00	0.00	0.00	0.00	0.00
basement_present	1.00	0.00	0.00	0.00	0.00
Cluster	0.00	0.00	1.00	0.00	0.00

Table 47 dataset with Clusters

Let's do a cluster profiling to understand the clusters

Cluster	price	living_measure	lot_measure	quality	room_bath	room_bed	ceil	condition	years old
0	419399.24	1676.66	7672.14	7.14	1.85	3.15	1.36	3.47	49.95
1	864873.85	3083.41	11482.38	9.04	2.77	3.92	1.85	3.25	25.48

Table 48 Cluster and Variables

Cluster 0:

Cluster 0, characterized by a mean price of $ 419399.24 , represents the cluster with the lowest mean price. This cluster also features the mean quality at 7 and the smallest mean living_measure, lot_measure among the 2 clusters. It's also the cluster with older properties with mean age of properties in this cluster being 50 years .
Cluster 1:

Cluster 1 comprises the highest-priced properties, boasting a mean price of $ 864873.85. Within this cluster, properties exhibit the highest mean living measure , the highest mean lot measure. Additionally, Cluster 1 properties showcase the highest quality of 9 .Also the number of bathrooms and bedrooms in this cluster are higher. The condition of the property is not very different for both clusters . However, properties in this cluster are the newer properties with average age of property being 25 years . This is a cluster of high-priced properties with larger living spaces and superior quality. We can see the impact of renovations on house prices. Properties in this cluster showing higher prices might be associated with recent renovations (e.g., living_measure15 and lot_measure15)

Clustering based on geographical coordinates (lat and long) revealed spatial patterns in house prices. It helped identify areas with higher demand and higher prices . We saw that cities near coast had highest prices compared to other cities

Appendix I

Feature	Importance from Gradient Boost Model
Lat	0.238521629
Living Measure	0.128039949
Furnished	0.081736883
Living Measure15	0.072185858
Quality	0.069817546
Room Bath	0.061293418
Ceil Measure	0.059164676
Long	0.028954201
Sight	0.02812136
Ceil	0.024201733
Years Old	0.023107258
Lot Measure15	0.022644032
Lot Measure	0.021125769
Zipcode 98004:	0.019193965
Room Bed	0.018573788
Basement Present	0.015855383
Coast	0.009949493
Condition	0.008135787
zipcode_98040:	0.007671554
zipcode_98023:	0.005891207
zipcode_98112:	0.005357858
zipcode_98039:	0.00396989
zipcode_98033:	0.003099166
zipcode_98042:	0.002315819
is_renovated:	0.001983289
zipcode_98106:	0.001960246
zipcode_98115:	0.001905075
zipcode_98117:	0.001772146
zipcode_98168:	0.001643222
zipcode_98092:	0.001602794
zipcode_98014:	0.000713089
zipcode_98166:	0.000707398
zipcode_98003:	0.000649993
zipcode_98070:	0.000614841
zipcode_98038:	0.000612032
zipcode_98056:	0.000553686
zipcode_98102:	0.000541077
zipcode_98030:	0.000482045
zipcode_98122:	0.000471661
zipcode_98045:	0.000453295
zipcode_98027:	0.000440193
zipcode_98065:	0.000436628
zipcode_98011:	0.000230281
zipcode_98010:	0.000200132
zipcode_98178:	0.000191489
zipcode_98125:	0.000184735
zipcode_98034:	0.000163714

zipcode_98008:	0.000131769
zipcode_98177:	0.000129035
zipcode_98072:	0.000106314
zipcode_98007:	9.13E-05
zipcode_98024:	7.44E-05
zipcode_98188:	7.33E-05
zipcode_98077:	3.37E-05
zipcode_98148:	9.26E-06