Table Contents

Table of figures

1. Introduction

1.1 Overview of house price prediction model

Houses are a basic need of human beings, providing shelter for their families against various environmental and artificial factors and the price of the house plays a key factor in deciding whether people will buy a house or not (Mao, 2023). Forecasting the price of the house as per its demand it depends on attributes like the number of bedrooms, location of the house, road access, available amenities, and other factors that influence the price prediction of the house which creates difficulty for accurate house price prediction (Yaping Zhao, 2024). The increase in economic growth of the country has led to the demand for better houses with qualities like plenty of spaces for gardens and parking, an area with no noise and pollution, and houses facing in a certain direction due to cultural beliefs or natural light direction. Accurately predicting the house price can benefit real estate investors and financial institutions for long-term investment (Yaping Zhao, 2024). In Nepal, people buy houses for better returns on their long-term investments as the house price is increasing every year.

In machine learning, house price prediction falls under the supervised learning regression problem as the target variable which is needed to be predicted is already known to us i.e. the Price of a house and all we need to do is map all the features variables to the target variable with the help of labelled to train a different type of regression model and compare their performance on unseen data to determine the best-fit model for the project.

1.2 Linear regression model

Linear regression is a supervised machine-learning algorithm model that learns the hidden pattern from labelled data with the numerical target variable and predicts the continuous target variable on unseen data based on the mapped independent input or feature variable using optimized linear functions (geeksforgeeks, 2024). In linear regression, the best fit line is determined which minimizes the error between the predicted and actual values. The best-fit line represents the linear relationship between the feature variable and target variable where the slope of the line shows how much the target variable or dependent variable changes for a unit change in the independent or feature variable (geeksforgeeks, 2024).

It is calculated using the formula

$Y = MX + b$, where Y is the dependent variable, M is the slope of the line, X is the independent variable and b is the y-intercept.

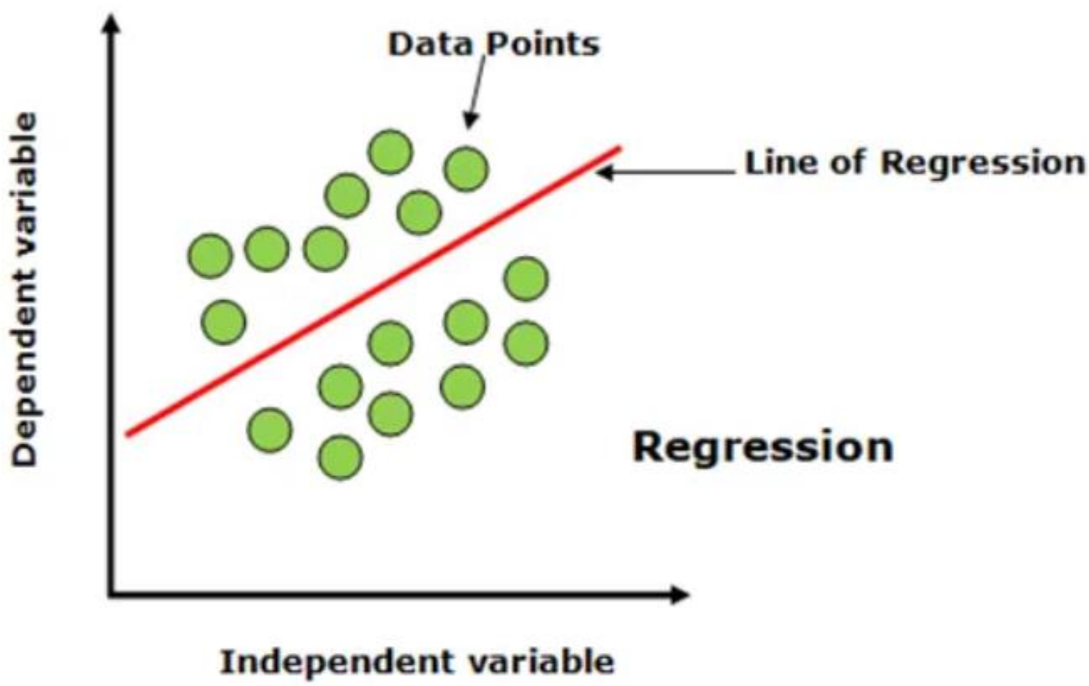

Figure 1 Linear regression (Salman, 2024)

1.3 Random Forest

Random Forest is an ensemble machine learning process which is used to predict both the numerical values and categorical values by combining the prediction made by different decision trees for accurate prediction. It handles non-linear relationships, is less sensitive to outliers and less prone to overfitting which is suitable for house price prediction projects. Random forest regressor works by combining multiple decision trees where each decision tree is built from a different subset of data and their prediction is aggregated to determine the final output (AnalytixLabs, 2023).

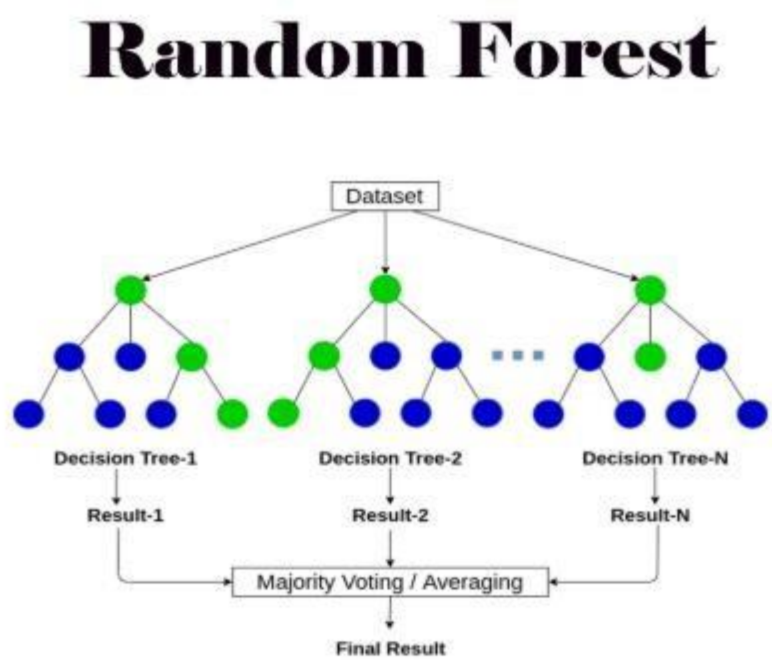

Figure 2 Random Forest tree (Brital, 2021)

1.4 Stacking Regressor

In machine learning, stacking is an ensemble learning method which combines the prediction made by several base model for getting the final prediction by the meta model for better performance of the model (Soni, 2023). It works by training multiple base model which is initialized and trained to feed their prediction as an input into a meta model which makes the final combination prediction with better accuracy and performance. The model diversity is strength by the use of stacking to reduce problem of overfitting of the model and create better generalization.

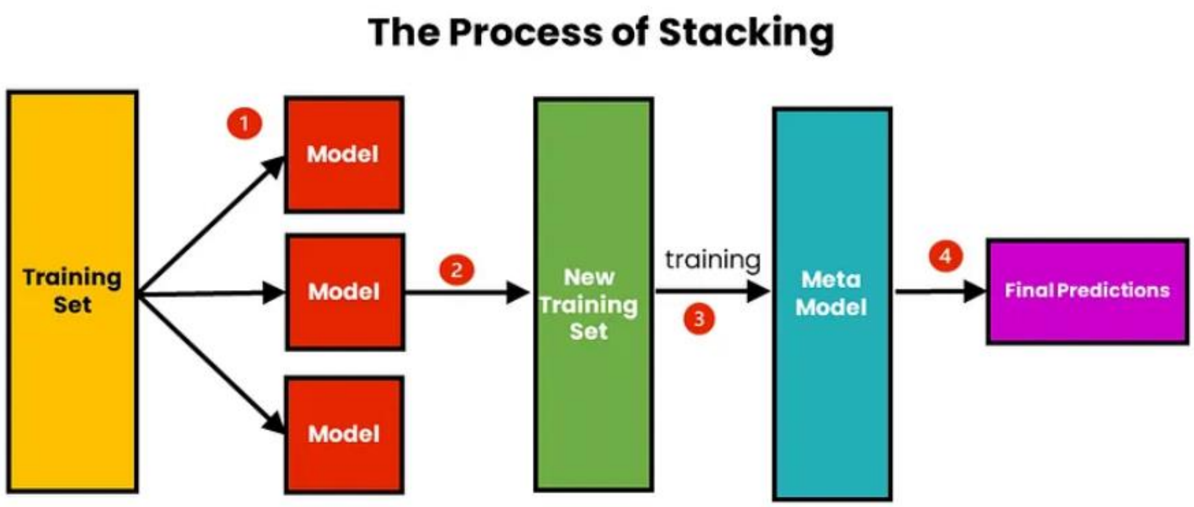

Figure 3 Stacking regressor (Soni, 2023)

1.5 R2 score

The R2 score is the evaluation metric to assess the performance of the regression model by finding out the goodness of fit which measure how well the regression model fits and predicts the dataset. Its value ranges from 0 to 1 where 0, indicate the poor performance of the model or could explain the variability of target variable 1 denote model predict the target variable correctly (Verma , 2023).

$$1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Figure 4 R2 score formula

1.6 Adjusted R2 Score

Overfitting can occur when adding many variables to the linear regression model, but the R2 score can't determine it, if the number of variables increases then the value of the r2 score also increases which is not the best metric for evaluation. Adjusted R2 score is an updated version of the R2 score where its value increases if the feature adds value for better model prediction (Abhigyan, 2020).

$$R^2_{adj} = 1 - \left[\frac{(1 - R^2)(n - 1)}{n - k - 1} \right]$$

Formula of Adjusted R-Squared

Figure 5 Adjusted r2 score formula (Abhigyan, 2020)

2. Problem Domain

2.1 Background

The housing price market is a dynamically changing market which is prone to volatility and uncertainty (Fatbardha Maloku, 2024), it is difficult for any machine learning model with high accuracy to predict the correct housing price. The price of houses in Nepal depends on different factors that influence the price prediction of the house which creates difficulty for accurate house price prediction (Yaping Zhao, 2024). Accurately predicting the price of a house requires the real-world dataset, which is not publicly available, if it is available then it might be outdated or not meaningful.

External factors like cultural beliefs, political changes, GDP fluctuation, and per capita income change, also affect the housing price which creates difficulty in estimating the exact price of the house. Some of the financial institutes buy the houses in bulk to seek profitable investment by reselling them at a higher price which is the price of the house. It also depends on the buyer's behaviour as they might not be interested in buying the house as per the attributes listed to them. Determining the right attributes, needs and interests of a buyer while choosing a house can be unpredictable (Mfazi, 2022).The changes in loan interest rate also affects the housing price as many people will have access to low-interest rate loans which leads to an increase in the number of people to afford the house which can create a market bubble often leading to the higher price of the housing market (Fatbardha Maloku, 2024).

Figure 6 Figure 2 Factor affecting the price of house (Nguyen, 2024)

2.2 Dataset Description

The dataset is taken from Kaggle, Chennai housing price data with 7109 rows and 22 columns. The description for each column is given below:

S.N.	Table Name	Description	Data Type
1.	PRT_ID	This column indicates the unique ID of each house's details.	object
2.	AREA	The "AREA" column denotes the location of the house in Chennai. There are a total of six areas in the dataset: Chromepet, Karapakkam, KK Nagar, Velachery, Anna Nagar, Adyar, and T Nagar. The Chromepet area has the highest row count value, 1702, and T Nagar has the lowest, 501.	object
3.	INT_SQFT	These columns indicate the overall size of the house in square feet where the highest square foot of the house is 2500 and the lowest is 500.	Int64
4.	DATE_SALE	This column contains the date when the house was sold.	object
5.	DIST_MAINROAD	These columns show the distance of the house from the main road.	Int64
6.	N_BEDROOM	This column contains the number of bedrooms in the house. Most of the houses contain at most 2 bedrooms in Chennai, with a maximum number of 4 bedrooms and a minimum number of bedrooms 1.	Float64
7.	N_ROOM	This column denotes the number of rooms contained in the house. Most of the houses contain at least 4 rooms, where the	Float64

		maximum number of rooms is 6 and the minimum number of rooms is 2	
8.	N_BATHROOM	This column denotes the number of bathrooms contained in the house. Almost every house contains 1 bathroom and the maximum number of bathrooms is 2.	Float64
9.	SALE_COND	This column denotes the sale condition of the house which was sold. It contains 5 unique value Adjland, partial, normalsale, Abnormal, Family.	object
10	PARK_FACIL	This column denotes the availability of parking facilities in the house.	object
11	DATE_BUILD	This column denotes the date when the house was constructed.	object
12	BUILD TYPE	This column denotes the type of property that is sold, house, commercial or others. Others might indicate, hospital, school etc.	object
13.	UTILITY_AVAIL	This column indicates the availability of utility facilities such as water, electricity, sewer etc. in the house. There are 3 unique values Nosewer, Allpub, and ELO.	object
14.	STREET	This column indicates the street road condition of the house. With 3 unique values i.e. paved, gravel or no access to the road.	object
15.	MZZONE	This column indicates the zone where the house is located. It contains 6 unique values	object

		RL (residential low density), RH(residential high density), RM (residential medium density), C (Commercial), A (Agricultural), and I (industrial) zone.	
16	QS_ROOMS	These columns indicate the quality score of a room of the house out of 5. The Average quality score is 3.517 for the house in the dataset	Float64
17.	QS_BATHROOM	These columns indicate the quality score of the bathroom of the house out of 5. The Average quality score is 3.5 for the house in the dataset.	Float64
18.	QS_BEDROOM	These columns indicate the quality score of the bedroom of the house out of 5. The Average quality score is 3.4 for the house in the dataset.	Float64
19	QS_OVERALL	These columns indicate the overall quality score of all rooms of the house out of 5. The Average overall quality score is 3.5 for the house in the dataset.	Float64
20.	REG_FEE	These columns denote the registration fee of the house after the sale is made.	Int64
21.	COMMIS	These columns denote the commission fee paid to the agent to sell the house.	Int64
22	SALES_PRICE	This column denotes the sale price of the house which is the target variable for model training to predict the house price.	Int64

3.Solution

3.1 Flowchart

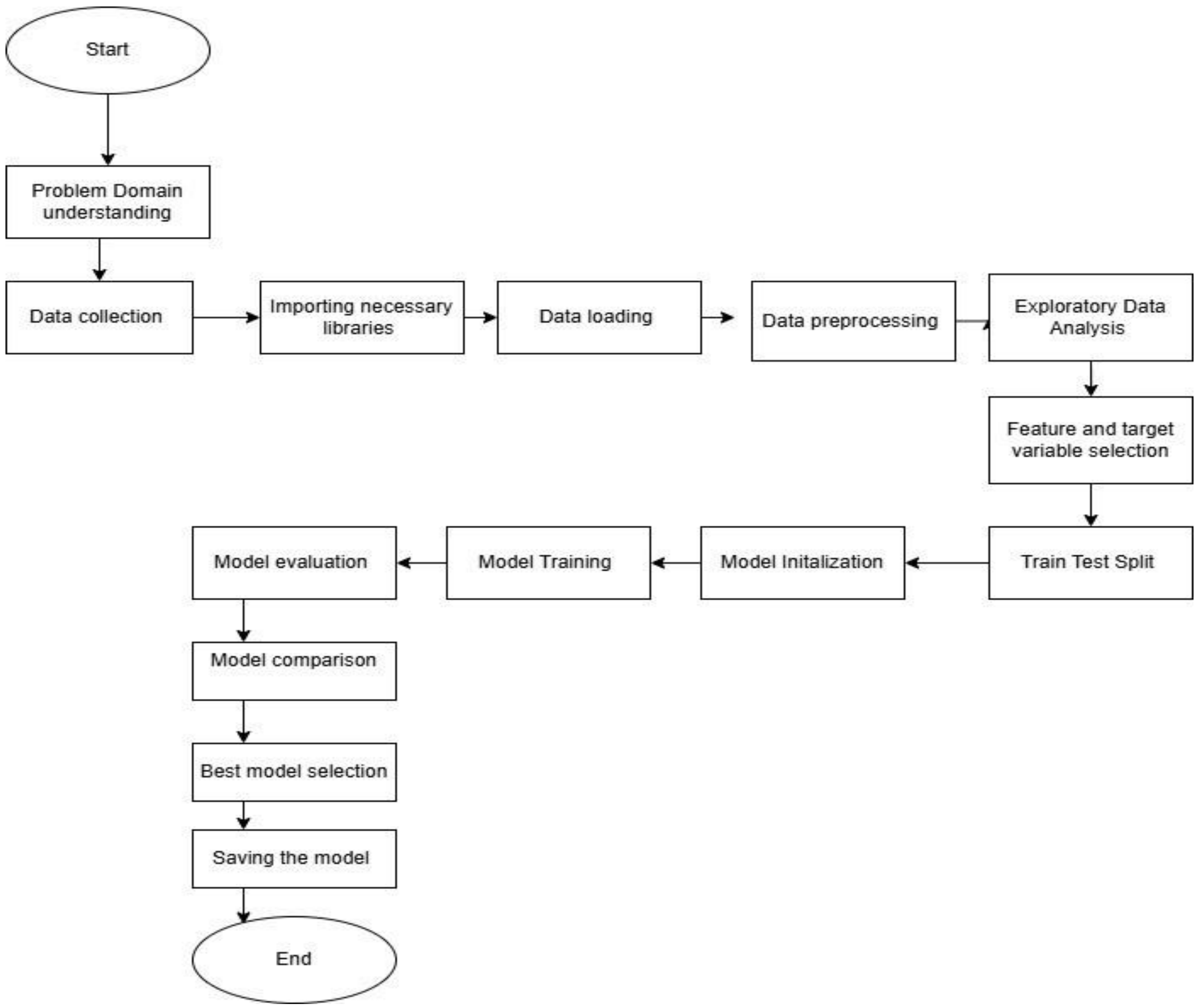

Figure 7 Flow chart of the system

3.2 Steps for Developing the Model:

3.2.1. Problem Domain Understanding

In this stage the Problem Domain that we are trying to solve is understood, the requirement is collected, and different approaches to find the best solution for the problem are studied and implemented.

3.2.2. Data collection

Good quality and quantity of data are necessary to train the machine learning model, which will improve its accuracy and performance on unseen data. The dataset for training the model was collected from the Kaggle Chennai dataset.

3.2.3. Importing necessary libraries

Performing machine learning tasks requires the tools and libraries listed below:

Pandas: It is a Python library used for manipulating, transforming and cleaning the data for analysis. In this project, pandas are used for loading the data in a data frame, data preprocessing, data transformation and Exploratory data analysis

Scikit-learn: It is an open-source Python library used for machine learning and data modelling-related tasks. It is used to create an instance of a machine-learning model for training.

NumPy: It is an open-source Python library used for working on multidimensional arrays, and numerical and mathematical operations.

Seaborn: It is an open-source library of Python used for data visualization and analysis.

Importing necessary libraries and module ¶

```
[1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
from sklearn.metrics import r2_score, mean_squared_error,mean_absolute_error
from sklearn.model_selection import train_test_split, cross_val_score,KFold,GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import StackingRegressor
```

Figure 8 Importing necessary libraires

3.2.4. Data loading

After the data is collected through web scraping, it must be loaded into the Jupiter notebook environment for further processing. Using pandas, the data is loaded as a Data Frame, which stores it in tabular format for easier processing. The data is loaded into data frame using read_csv method where the file path is passed as a parameter

```
]: #loading the csv file using pandas read_csv() method
house_price_df = pd.read_csv("Chennai houseing sale.csv")
```

Figure 9 Loading the csv file into data frame

3.2.5. Data understanding

3.2.5.1 Inspection of last 5 row of data frame

The last 5 row of data frame of the data frame is inspect using tail() with default parameters.

2.1) Inspection of 5 last row of the dataframe

```
# print out the last 5 rows of the dataframe
house_price_df.tail()
```

	PRT_ID	AREA	INT_SQFT	DATE_SALE	DIST_MAINROAD	N_BEDROOM	N_BATHROOM	N_ROOM	SALE_COND	PARK_FACIL	...	UTILITY_AVAIL	STREET	MZZC
7104	P03834	Karapakkam	598	03-01-2011	51	1.0	1.0	2	AdjLand	No	...	ELO	No Access	
7105	P10000	Velachery	1897	08-04-2004	52	3.0	2.0	5	Family	Yes	...	NoSeWa	No Access	
7106	P09594	Velachery	1614	25-08-2006	152	2.0	1.0	4	Normal Sale	No	...	NoSeWa	Gravel	
7107	P06508	Karapakkam	787	03-08-2009	40	1.0	1.0	2	Partial	Yes	...	ELO	Paved	
7108	P09794	Velachery	1896	13-07-2005	156	3.0	2.0	5	Partial	Yes	...	ELO	Paved	

5 rows × 22 columns

Figure 10 Inspection of the last 5 row of data frame

3.2.5.2 Inspection of first 5 rows of data frame

The first five row of the data frame is inspected using head() method with default parameter of pandas library.

2.2 Inspection of first five row of the dataframe

```
# print out the top 5 rows of the dataframe
house_price_df.head()
```

	PRT_ID	AREA	INT_SQFT	DATE_SALE	DIST_MAINROAD	N_BEDROOM	N_BATHROOM	N_ROOM	SALE_COND	PARK_FACIL	...	UTILITY_AVAIL	STREET	MZZONE
0	P03210	Karapakkam	1004	04-05-2011	131	1.0	1.0	3	AbNormal	Yes	...	AllPub	Paved	A
1	P09411	Anna Nagar	1986	19-12-2006	26	2.0	1.0	5	AbNormal	No	...	AllPub	Gravel	RH
2	P01812	Adyar	909	04-02-2012	70	1.0	1.0	3	AbNormal	Yes	...	ELO	Gravel	RL
3	P05346	Velachery	1855	13-03-2010	14	3.0	2.0	5	Family	No	...	NoSewr	Paved	I
4	P06210	Karapakkam	1226	05-10-2009	84	1.0	1.0	3	AbNormal	Yes	...	AllPub	Gravel	C

5 rows × 22 columns

Figure 11 Inspection of the first 5 row of data frame

3.2.5.3 Dimension of the data frame

The dimension of the data frame represents a number of rows and columns which is checked using. shape method. There are a total of 7109 rows and 22 columns in the data frame.

```
10]: #Display the dimension of the dataframe .i.e number of rows and columns
house_price_df.shape
```

```
10]: (7109, 22)
```

Figure 12 Inspection of the number of rows and columns in a data frame

3.2.5.4 Columns of data frame

The column of a data frame is inspected using the .columns method.

```
# Display the columns of dataframe using .columns attribute
house_price_df.columns
```

```
Index(['PRT_ID', 'AREA', 'INT_SQFT', 'DATE_SALE', 'DIST_MAINROAD', 'N_BEDROOM',
       'N_BATHROOM', 'N_ROOM', 'SALE_COND', 'PARK_FACIL', 'DATE_BUILD',
       'BUILDTYPE', 'UTILITY_AVAIL', 'STREET', 'MZZONE', 'QS_ROOMS',
       'QS_BATHROOM', 'QS_BEDROOM', 'QS_OVERALL', 'REG_FEE', 'COMMIS',
       'SALES_PRICE'],
      dtype='object')
```

Figure 13 Column of the data frame

3.2.5.5 Data type of each column

The data type of each column of data frame is inspected using. dtypes method of pandas.

```
[14]: # Display the data type of each columns using .dtypes attributes
      house_price_df.dtypes

[14]: PRT_ID              object
      AREA                object
      INT_SQFT             int64
      DATE_SALE           object
      DIST_MAINROAD        int64
      N_BEDROOM          float64
      N_BATHROOM         float64
      N_ROOM               int64
      SALE_COND           object
      PARK_FACIL          object
      DATE_BUILD          object
      BUILDTYPE           object
      UTILITY_AVAIL       object
      STREET              object
      MZZONE              object
      QS_ROOMS           float64
      QS_BATHROOM        float64
      QS_BEDROOM         float64
      QS_OVERALL         float64
      REG_FEE              int64
      COMMIS               int64
      SALES_PRICE          int64
      dtype: object
```

Figure 14 Inspection of data type of each column

3.2.5.6 Information about each column

The information about each column is view using the info() method of pandas which shows the non-null count, datatype, range index, memory usage of the columns of the data frame

```
[16]: # Display the information about each columns like datatypes, rowcount, null value, no.of columns and rows. using info() method
      house_price_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7109 entries, 0 to 7108
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   PRT_ID         7109 non-null   object
 1   AREA           7109 non-null   object
 2   INT_SQFT       7109 non-null   int64
 3   DATE_SALE      7109 non-null   object
 4   DIST_MAINROAD  7109 non-null   int64
 5   N_BEDROOM      7108 non-null   float64
 6   N_BATHROOM     7104 non-null   float64
 7   N_ROOM         7109 non-null   int64
 8   SALE_COND      7109 non-null   object
 9   PARK_FACIL     7109 non-null   object
 10  DATE_BUILD     7109 non-null   object
 11  BUILDTYPE      7109 non-null   object
 12  UTILITY_AVAIL  7109 non-null   object
 13  STREET         7109 non-null   object
 14  MZZONE         7109 non-null   object
 15  QS_ROOMS       7109 non-null   float64
 16  QS_BATHROOM    7109 non-null   float64
 17  QS_BEDROOM     7109 non-null   float64
 18  QS_OVERALL     7061 non-null   float64
 19  REG_FEE        7109 non-null   int64
 20  COMMIS         7109 non-null   int64
 21  SALES_PRICE    7109 non-null   int64
dtypes: float64(6), int64(6), object(10)
memory usage: 1.2+ MB
```

Figure 15 Information about each column of the data frame

3.2.5.7 Statistical description of each numerical column

The statistical description like mean, quartiles, count, maximum and minimum values and standard deviation of the each column is inspected using describe() method of pandas.

```
# Statistical descrption of numerical columns including mean,standard deviation, quartile range,
# maximum and minimum values using  describe() method
house_price_df.describe()
```

	INT_SQFT	DIST_MAINROAD	N_BEDROOM	N_BATHROOM	N_ROOM	QS_ROOMS	QS_BATHROOM	QS_BEDROOM	QS_OVERALL	REG_FEE	COMMIS	SALES_PRICE
count	7109.000000	7109.000000	7108.000000	7104.000000	7109.000000	7109.000000	7109.000000	7109.000000	7061.000000	7109.000000	7109.000000	7.109000e+03
mean	1382.073006	99.603179	1.637029	1.213260	3.688704	3.517471	3.507244	3.485300	3.503254	376938.330708	141005.726544	1.089491e+07
std	457.410902	57.403110	0.802902	0.409639	1.019099	0.891972	0.897834	0.887266	0.527223	143070.662010	78768.093718	3.768603e+06
min	500.000000	0.000000	1.000000	1.000000	2.000000	2.000000	2.000000	2.000000	2.000000	71177.000000	5055.000000	2.156875e+06
25%	993.000000	50.000000	1.000000	1.000000	3.000000	2.700000	2.700000	2.700000	3.130000	272406.000000	84219.000000	8.272100e+06
50%	1373.000000	99.000000	1.000000	1.000000	4.000000	3.500000	3.500000	3.500000	3.500000	349486.000000	127628.000000	1.033505e+07
75%	1744.000000	148.000000	2.000000	1.000000	4.000000	4.300000	4.300000	4.300000	3.890000	451562.000000	184506.000000	1.299390e+07
max	2500.000000	200.000000	4.000000	2.000000	6.000000	5.000000	5.000000	5.000000	4.970000	983922.000000	495405.000000	2.366734e+07

Figure 16 Statistical description of each column

3.2.5.8 Inspection of null values

The null values are inspected using .isna() method of pandas along with sum() method to calucluate the count of null values in each column of the data frame.

```
[20]: #.isna() method display the bool value for missing values and .sum() method is used to count the no. of missing values
house_price_df.isna().sum()
```

```
N_BEDROOM        1
N_BATHROOM       5
N_ROOM           0
SALE_COND        0
PARK_FACIL       0
DATE_BUILD       0
BUILDTYPE        0
UTILITY_AVAIL    0
STREET           0
MZZONE           0
QS_ROOMS         0
QS_BATHROOM      0
QS_BEDROOM       0
QS_OVERALL      48
REG_FEE          0
COMMIS           0
SALES_PRICE      0
dtype: int64
```

Figure 17 Inspection of null values

3.2.5.9 Frequency count of each categorical columns unique value

The unique values with their frequency count of each categorical column are viewed using .value_counts() of the pandas. Firstly the categorical columns were classified and the loop was created to view all the unique values of each column

```
228]:
category_columns = house_price_df.select_dtypes(include=['object']).columns

# Loop to print value counts for each categorical column
for columns in category_columns:
    if not columns in  ['PRT_ID','DATE_SALE','DATE_BUILD'] :
        print(f"Frequency count of unique value  for '{columns}' columns:")
        print(house_price_df[columns].value_counts())
        print("\n")
```

```
Frequency count of unique value  for 'AREA' columns:
AREA
Chromepet     1702
Karapakkam    1366
KK Nagar       997
Velachery      981
Anna Nagar     788
Adyar          774
T Nagar        501
Name: count, dtype: int64


Frequency count of unique value  for 'SALE_COND' columns:
SALE_COND
AdjLand       1439
Partial       1433
NormalSale    1423
AbNormal      1411
```

Figure 18 Frequency count of unique value of a categorical column in data frame

3.2.5.10 Inspection of duplicate values.

The duplicated values in the data frame were inspected using duplicated() method with sum() method to calculate the total count of duplicated values.

```
]:  house_price_df.duplicated().sum()

]:  np.int64(0)
```

Figure 19 Inspection of duplicate values

3.2.6 Data Cleaning

Data preprocessing is the process of preparing the data for analysis and model training by cleaning and transforming the data which includes steps like removing the missing values, duplicate values, removal of unwanted features, outlier removal, data scaling and normalization, data encoding in case of a categorical variable and feature engineering.

3.2.6.1 Removing Null Values

The missing values of QS_OVERALL, N_BATHROOM, and N_BEDROOM were imputed by the median values of that column since it is a discrete variable

```python
# removing the rows with the missing values
house_price_df['QS_OVERALL']= house_price_df['QS_OVERALL'].fillna(house_price_df['QS_OVERALL'].median())
house_price_df['N_BATHROOM']= house_price_df['N_BATHROOM'].fillna(house_price_df['N_BATHROOM'].median())
house_price_df['N_BEDROOM']= house_price_df['N_BEDROOM'].fillna(house_price_df['N_BEDROOM'].median())
```

Figure 20 Imputation of missing values using median

3.2.6.2 Cleaning inconsistency in columns

Categorical columns consist of inconsistent data such as extra white space, incorrect naming of values, and duplicate categories which are cleaned as shown below

STREET column

```python
house_price_df['STREET'] = house_price_df['STREET'].replace({'Pavd':'Paved','NoAccess':'No Access' })
house_price_df['STREET'].unique()
```

```
array(['Paved', 'Gravel', 'No Access'], dtype=object)
```

Figure 21 data cleaning STREET column

UTILITY_AVAIL column

```python
house_price_df['UTILITY_AVAIL'] = house_price_df['UTILITY_AVAIL'].replace({'NoSewr ':'NoSewer','All Pub':'AllPub','NoSeWa':'NoSewer'})
house_price_df['UTILITY_AVAIL'].unique()
```

```
array(['AllPub', 'ELO', 'NoSewer'], dtype=object)
```

Figure 22 Data Cleaning UTILITY_AVAIL column

BUILD TYPE column

```python
house_price_df['BUILDTYPE'] = house_price_df['BUILDTYPE'].replace({'Comercial':'Commercial','Other':'Others'})
house_price_df['BUILDTYPE'].unique()
```

```
array(['Commercial', 'Others', 'House'], dtype=object)
```

Figure 23 Data cleaning BUILD TYPE column

PARK_FACIL column

```
house_price_df['PARK_FACIL'] = house_price_df['PARK_FACIL'].replace({'Noo':'No'})
house_price_df['PARK_FACIL'].unique()
```

```
array(['Yes', 'No'], dtype=object)
```

Figure 24 Data cleaning PARK_FACIL column

SALE_COND column

```
house_price_df['SALE_COND'] = house_price_df['SALE_COND'].str.strip().str.replace(' ', '')
house_price_df['SALE_COND'] = house_price_df['SALE_COND'].replace({'Partiall':'Partial','PartiaLl':'Partial'})
house_price_df['SALE_COND'].unique()
```

```
array(['AbNormal', 'Family', 'Partial', 'AdjLand', 'NormalSale'],
      dtype=object)
```

Figure 25 Data cleaning SALE_COND column

AREA column

```
AREA_correct_mapping = {
    'Chrompt': 'Chromepet',
    'Chrmpet': 'Chromepet',
    'Chormpet': 'Chromepet',
    'Chrompet' : 'Chromepet',
    'Karapakam': 'Karapakkam',
    'KKNagar': 'KK Nagar',
    'Velchery': 'Velachery',
    'Ann Nagar': 'Anna Nagar',
    'Ana Nagar': 'Anna Nagar',
    'Adyr': 'Adyar',
    'TNagar': 'T Nagar'
}
house_price_df['AREA'] = house_price_df['AREA'].replace(AREA_correct_mapping)
house_price_df['AREA'].unique()
```

```
array(['Karapakkam', 'Anna Nagar', 'Adyar', 'Velachery', 'Chromepet',
       'KK Nagar', 'T Nagar'], dtype=object)
```

Figure 26 Data cleaning AREA column

3.2.6.3 Data type conversion

The DATE_SALE and DATE_BUILD columns were converted into datetime object using to_datetime method of pandas, whereas N_BEDROOM and N_BATHROOM were converted into integer data type using astype() with int64 as a parameter because the number of room or bathroom can't be float values it is a discrete value.

```
house_price_df['DATE_SALE'] = pd.to_datetime(house_price_df['DATE_SALE'],dayfirst=True)
house_price_df['DATE_BUILD'] = pd.to_datetime(house_price_df['DATE_BUILD'],dayfirst=True)
house_price_df['N_BEDROOM'] = house_price_df['N_BEDROOM'].astype(int)
house_price_df['N_BATHROOM'] = house_price_df['N_BATHROOM'].astype(int)
```

Figure 27 Data type conversion

3.2.7. Exploratory Data Analysis

In Exploratory Data Analysis, we analyze, summarize, and investigate the data to find out the hidden patterns, detect anomalies, insights and characteristics of the data by visualizing it, performing univariate and bivariate analysis, and statistical analysis.

3.2.7.1 Distribution of Sale Conditions

The overall distribution of the sale condition of the house that is already sold is displayed in a visualized manner using a seaborn count plot.

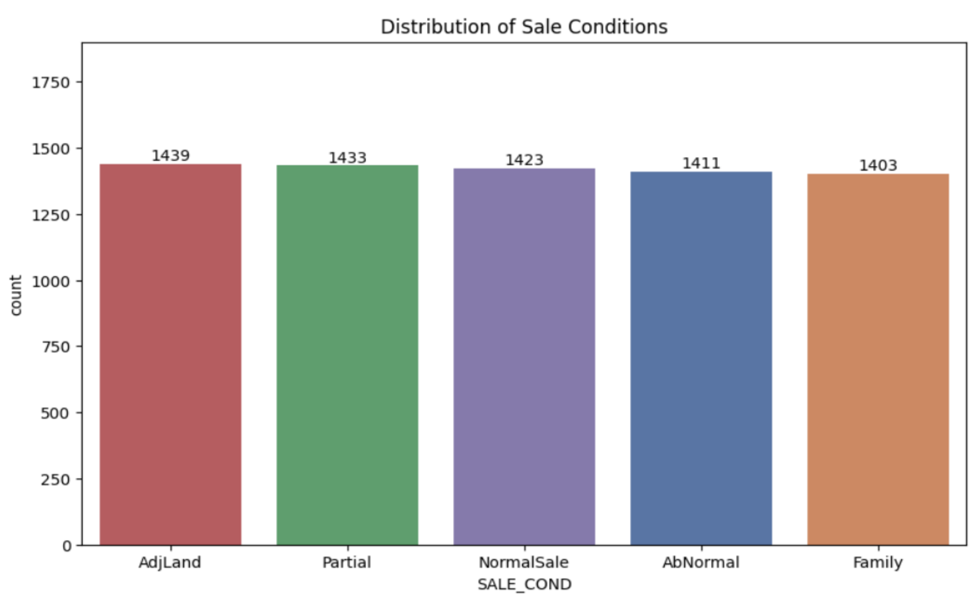

Figure 28 Distribution of Sale condition of house that is sold

3.2.7.2 Distribution of Utility Availability

The availability of utility facilities i.e. no sewer, Allpub and ELO frequency count is displayed using a count plot. Most of the houses that are sold have no sewage facility.

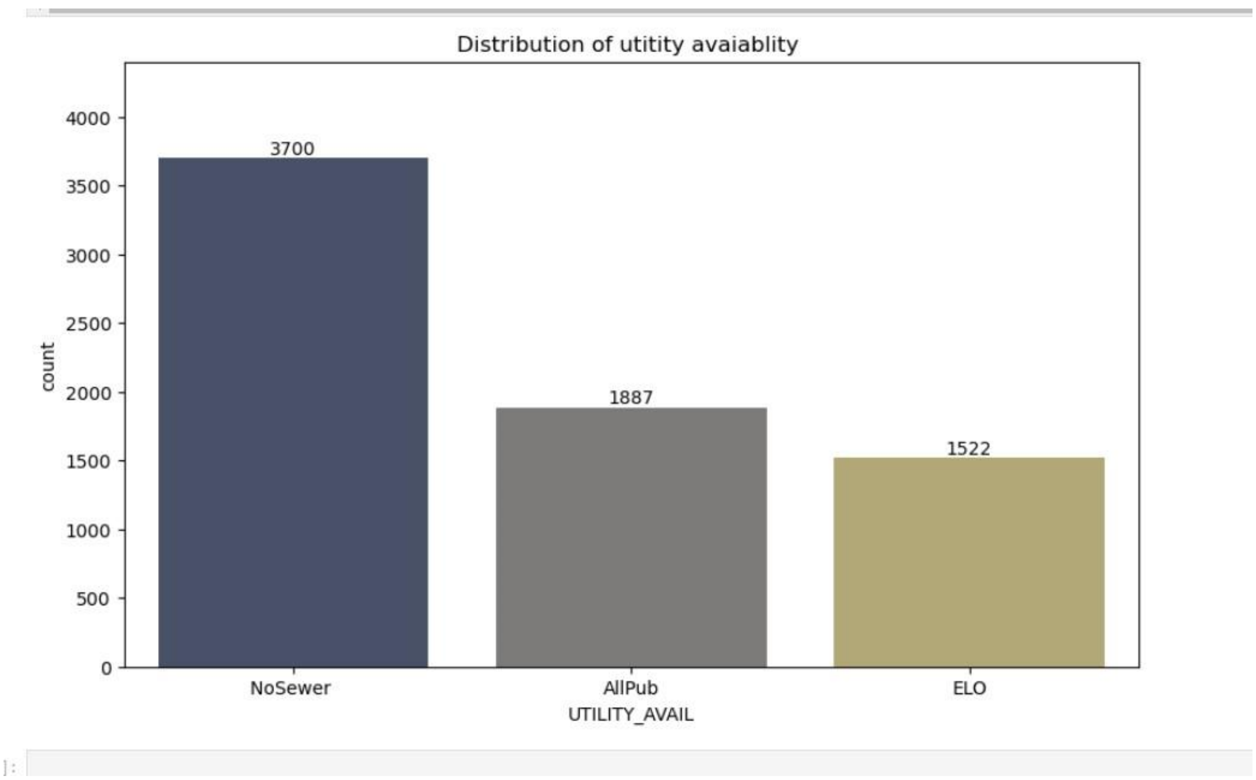

Figure 29 Distribution of utility availability

3.2.7.3 Distribution of Parking Facilities

The parking facilities of house distribution are visualized using a seaborn count plot.

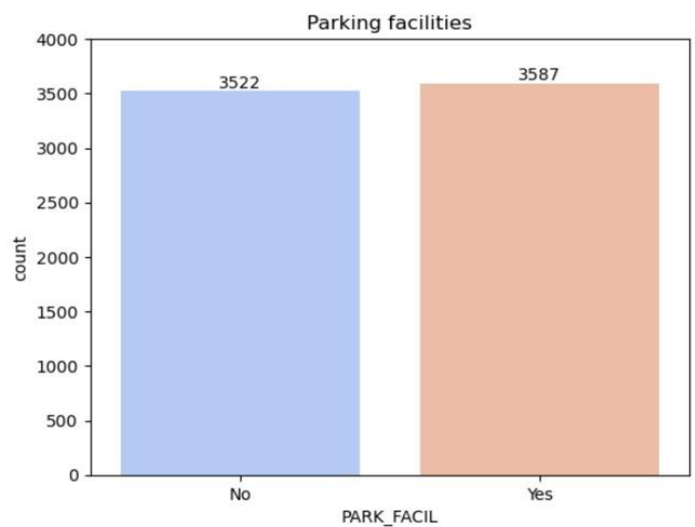

Figure 30 Distribution of Parking facilities

3.2.7.4 Distribution of the number of bathrooms

Most of the houses consist of at least 1 bathroom in Chennai. Around 1515 houses contain 2 bathrooms and 5594 houses consist of 2 bathrooms

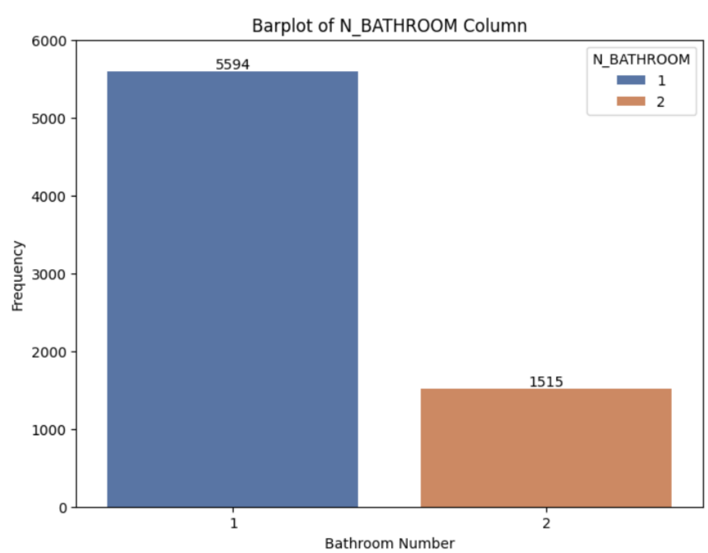

Figure 31 Distribution of number of bathrooms in house

3.2.7.5 Distribution of Street Road condition

The condition of the street road paved, gravel and no access to the road is visualized using barplot of seaborn below.

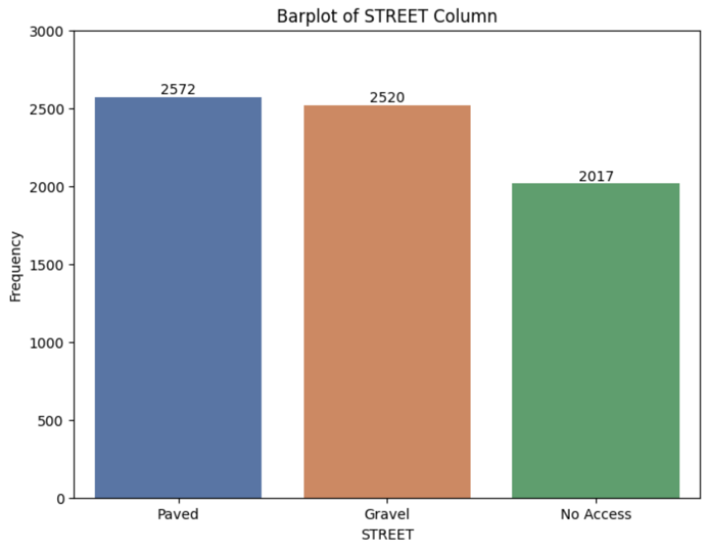

Figure 32 Bar plot of Street condition

3.2.7.6 Distribution of N_bedroom

From the below visualization the number of bedrooms of a house count is shown where the most of the house consist of 1(3796) bedrooms and the highest number of bedrooms in the house is 4 (254) .

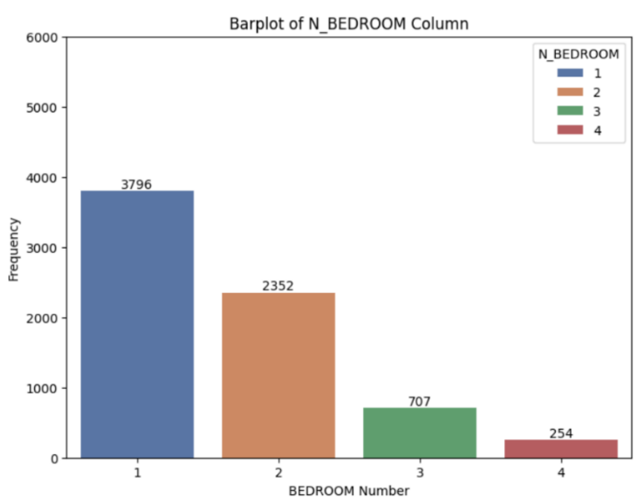

Figure 33 Barplot of Number of bedroom in house

3.2.7.7 Histogram of the sale price of the house.

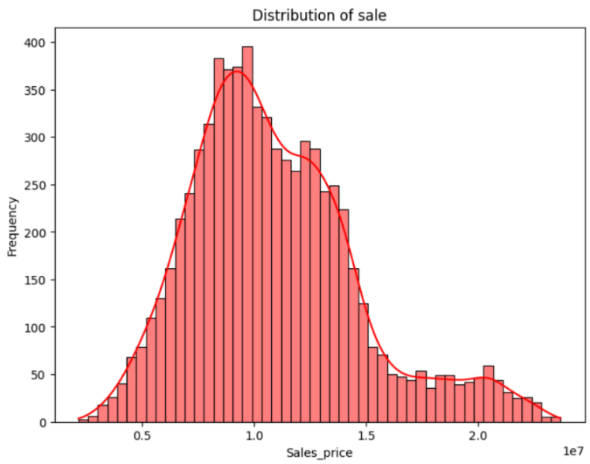

Figure 34 Histogram of target variable

3.2.7.8 Boxplot of all the numerical columns

There are some outliers in the target variable SALE_PRICE which will be removed.

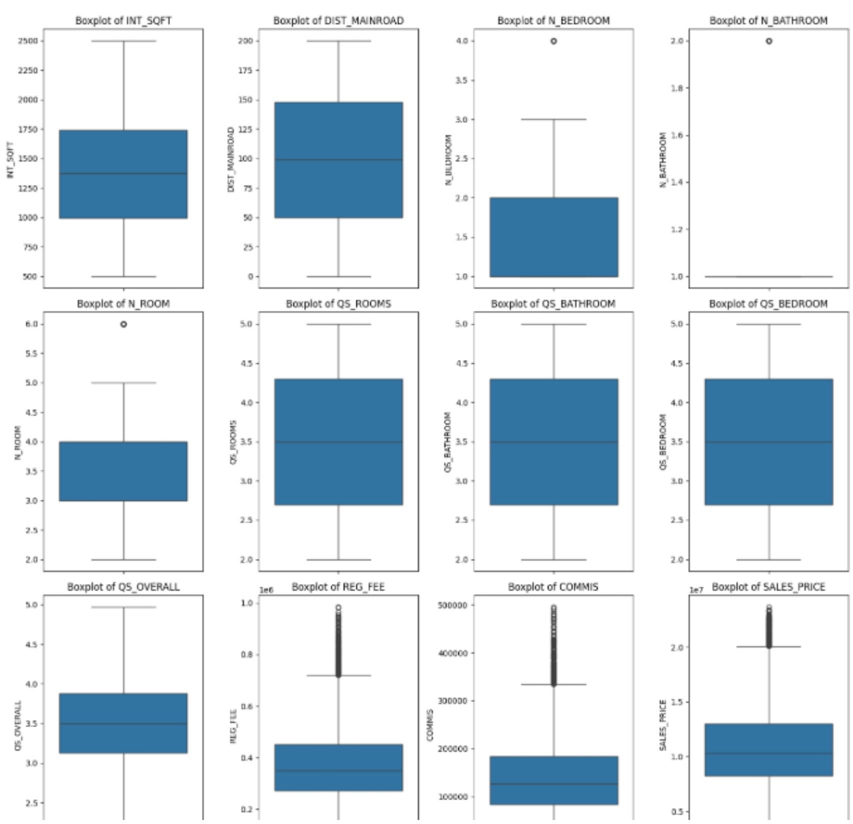

Figure 35 Box plot of numerical columns

3.2.7.9 Average price of the house based on area in Chennai

The average price of the house based on the area is visualized below which indicates us that prices of house in Anna Nagar and T nagar are expensive compared to other areas and the karapakkam area house are cheaper than other areas.

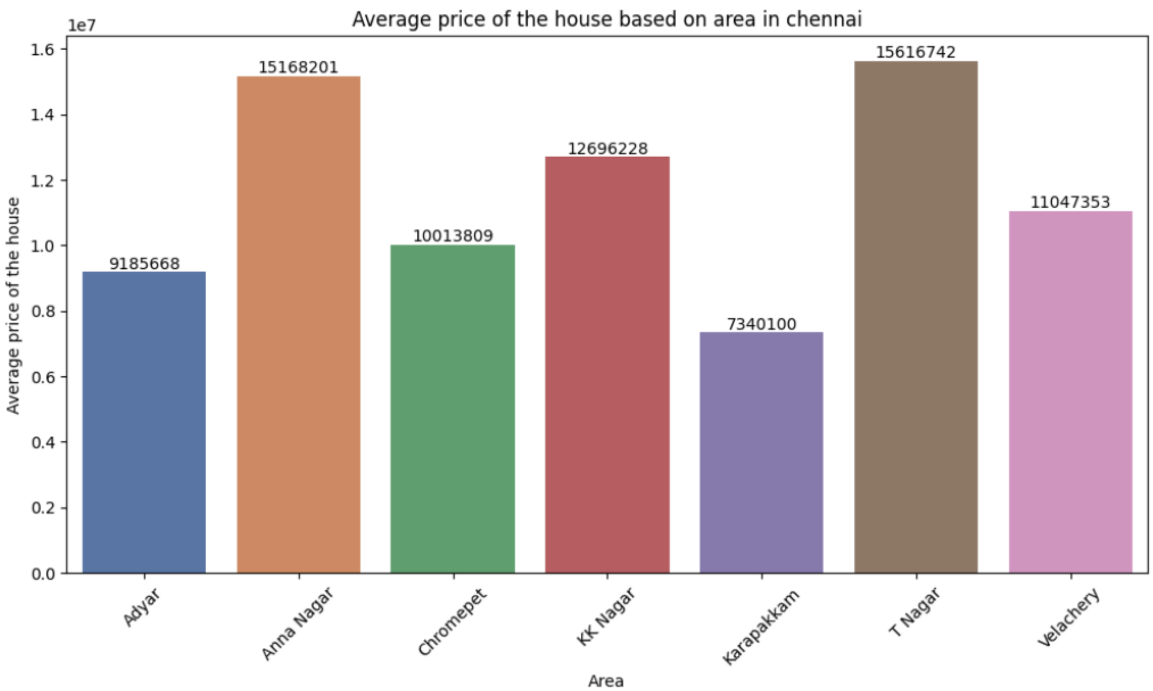

Figure 36 Average price of house based on area

3.2.7.10 Average price of the house based on utility status and street road condition

Houses with Gravel, paved road and all utilities available are more expensive than others.

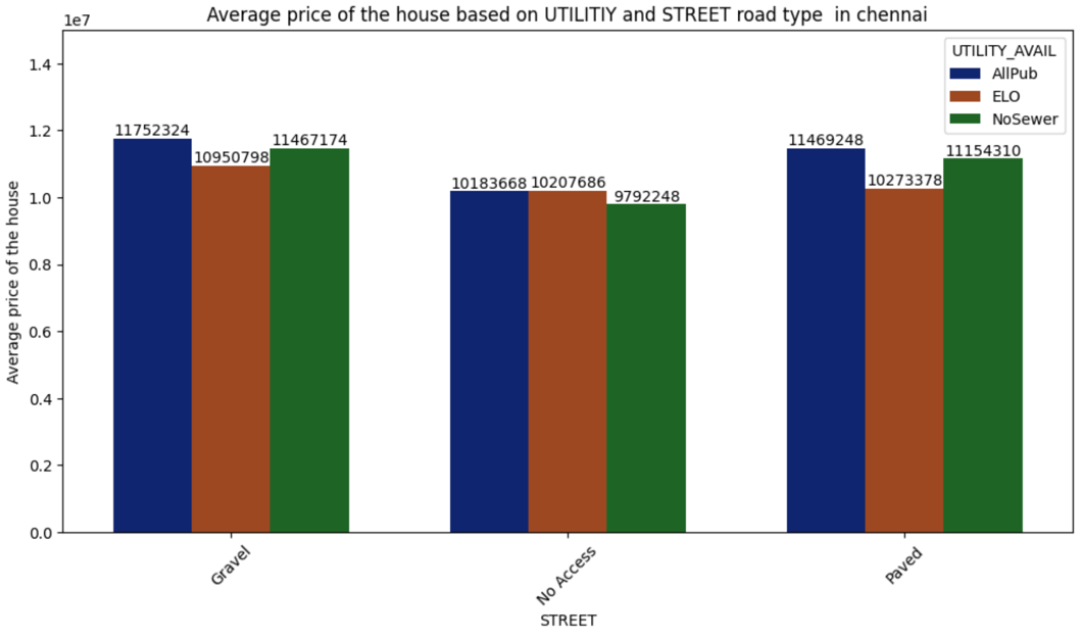

Figure 37 Average price of house based on utility status and street condition

3.2.7.11 Average price of a house based on area and parking facilities.

Houses with parking facilities are more expensive than no parking facilities in all the area of Chennai.

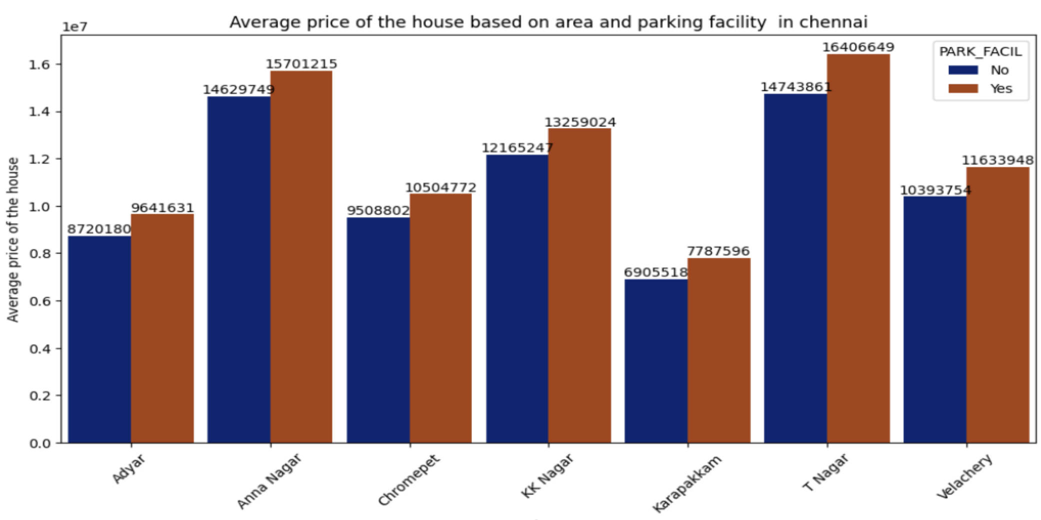

Figure 38 Average price of house based on area and parking facilities

3.2.7.12 Average price of a house based on MZZONE

The residential area houses are more expensive than the other zone areas like agricultural, commercial and industrial zone houses in Chennai.

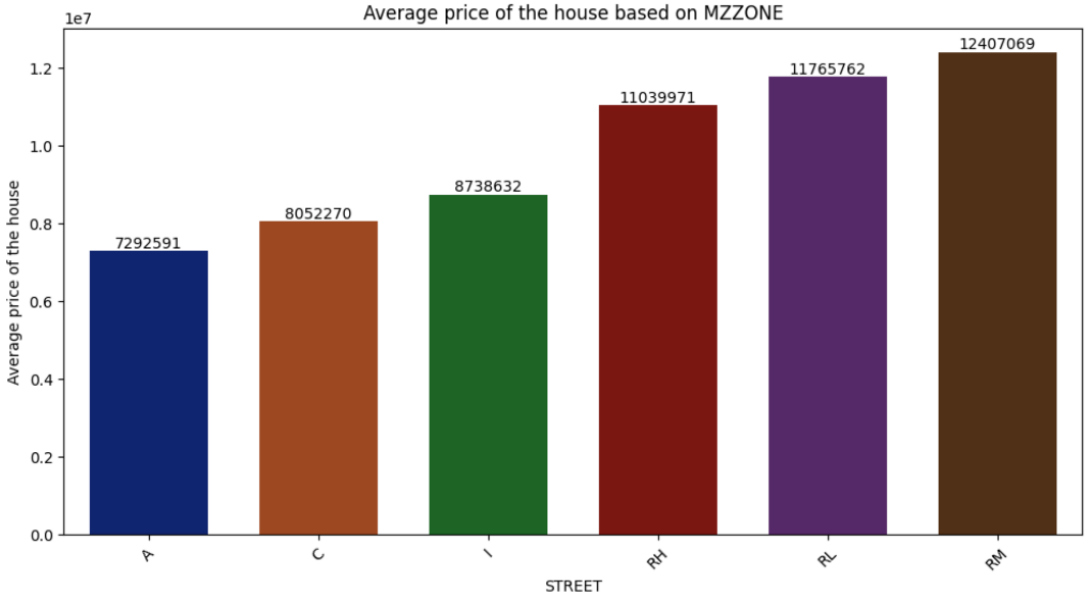

Figure 39 average price of a house base on the mzzone

3.2.7.13 Numerical variable vs sale price target variable regression plot

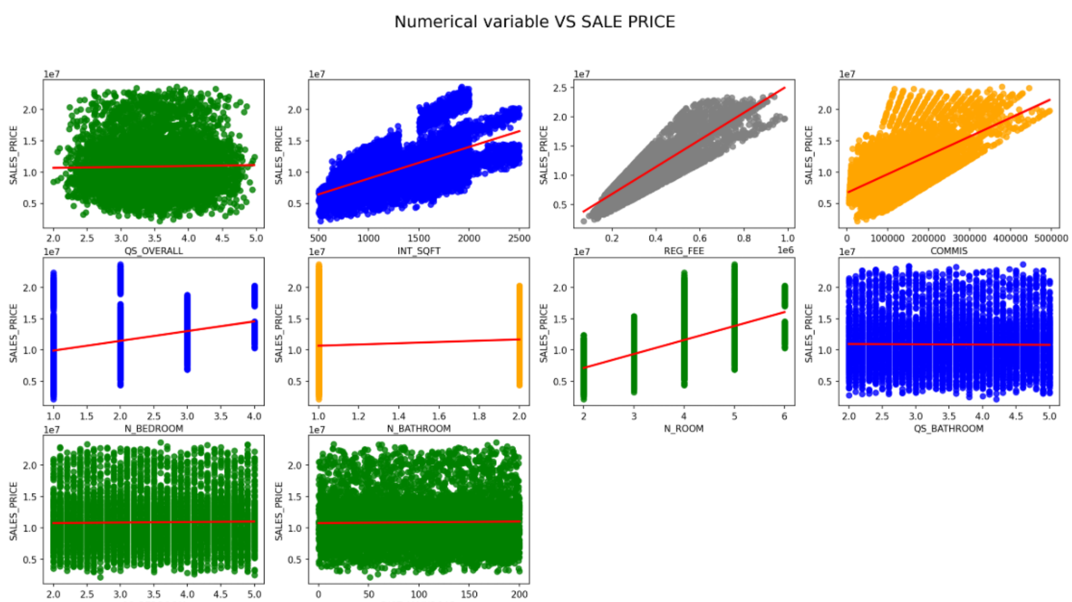

Figure 40 regression plot for numerical columns vs target column

The overall quality score of the room, the quality score of the bathroom and bedroom, and the distance from the main road columns have weak correlations with the target variable which suggests that these features doesn't affect the price of the house.

3.2.7.14 Heatmap of all the numerical columns

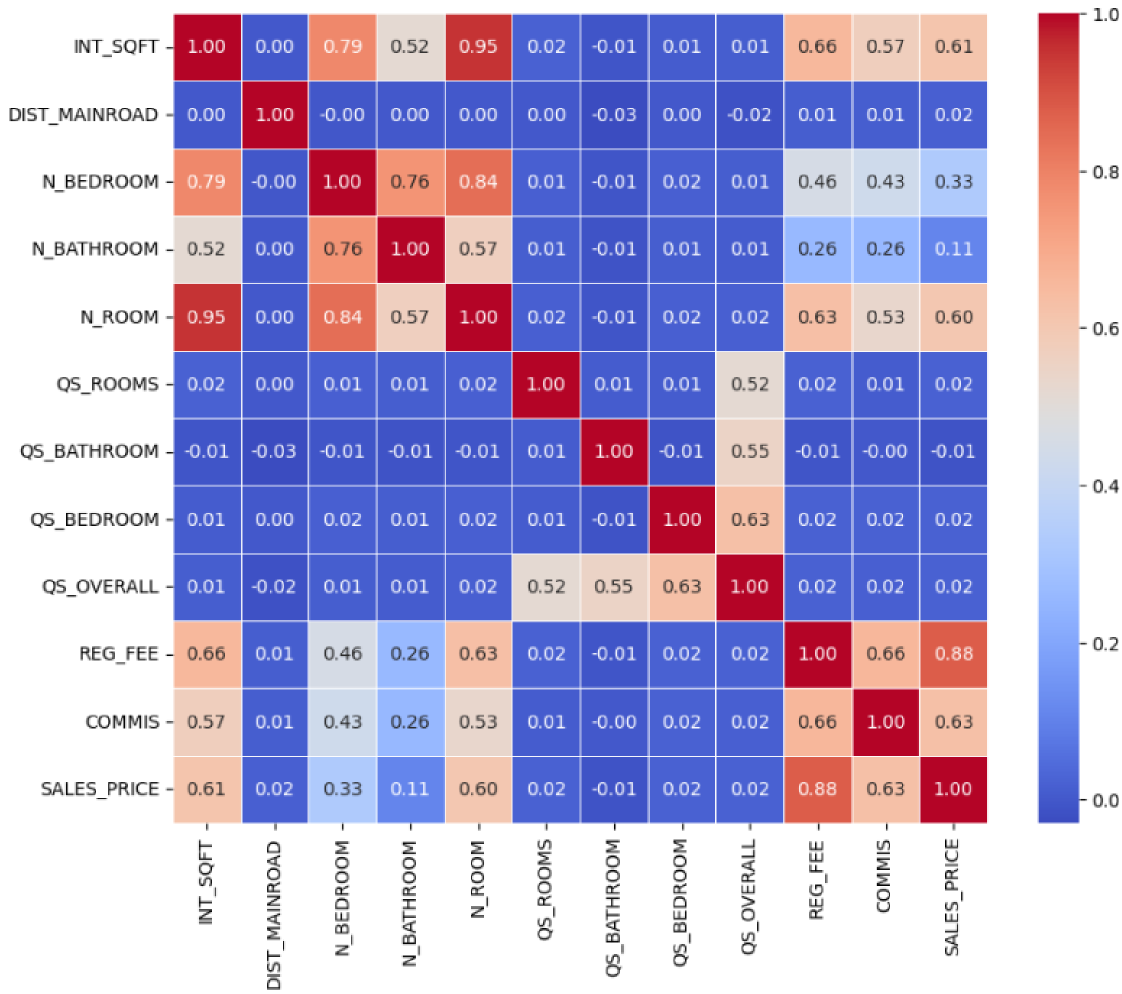

Figure 41 Heat map of all the numerical columns

The registration fee and commission fee have a strong correlation with the target variable, but to determine the price of the house this feature is not required because it doesn't determine the actual price of the house, it is the extra fee paid to sale the house.

3.2.8 Feature engineering

In the machine learning lifecycle, feature engineering is performed to create new or extra features derived from the older feature or using domain knowledge to enhance the performance of the machine learning model. Feature transformation like data scaling, encoding, data transformation is performed in this step.

3.2.8.1 Feature creation

The new feature age of the property is calculated by subtracting the date of sale year and date of build year feature.

```
[232]: house_price_df['AGE_PROPERTY'] = house_price_df['DATE_SALE'].dt.year - house_price_df['DATE_BUILD'].dt.year
       house_price_df['AGE_PROPERTY']

[232]: 0       44
       1       11
       2       20
       3       22
       4       30
               ..
       7104    49
       7105     9
       7106    28
       7107    32
       7108    44
       Name: AGE_PROPERTY, Length: 7109, dtype: int32
```

Figure 42 Feature creation age of the house

3.2.8.2 Removal of unwanted features

The unnecessary features are removed from the data frame using the drop method and specifying the column's names to remove.

```
house_price_df = house_price_df.drop(columns=["DATE_SALE","DATE_BUILD","PRT_ID","REG_FEE","COMMIS"])
```

Figure 43 Removal of unwanted features

3.2.8.3 Feature transformation label encoding

UTILITY_AVAIL column label encoding

```
[121]: label_encoder = LabelEncoder()
```

```
[123]: house_price_df['UTILITY_AVAIL'] = house_price_df['UTILITY_AVAIL'].map({'NoSeWa': 0, 'ELO': 1, 'AllPub': 2})
       house_price_df['UTILITY_AVAIL'].unique()
```

```
[123]: array([2, 1, 0])
```

PARK_FACIL column label encoding

```
[128]: house_price_df['PARK_FACIL'] = house_price_df['PARK_FACIL'].map({'Yes': 1, 'No': 0})
       house_price_df['PARK_FACIL'].unique()
```

```
[128]: array([1, 0])
```

Figure 44 Label encoding

3.2.8.4 Feature transformation one hot encoding

One hot encoding transforms the categorical data into new columns for each category the binary representation is carried out which improve the overall performance of the machine learning model

```
house_price_df_encode = pd.get_dummies(house_price_df, columns=['AREA', 'BUILDTYPE', 'STREET','SALE_COND','MZZONE'])
house_price_df_encode
```

Figure 45 one hot encoding

3.2.8.5 Feature scaling

Feature scaling is performed to make sure that the values range of each feature lies within the same range because the huge difference in the value can cause biases toward the large value column which affects the generalization of the model. Standardization is performed which scales the data to a mean 0 and a standard deviation of 1

```
standardscaler = StandardScaler()
```

```
scaling_df = house_price_df.drop(columns=['SALES_PRICE'])
house_price_scale_df = standardscaler.fit_transform(scaling_df)
```

Figure 46 Feature scaling using standardization

3.2.9. Feature and Target variable selection

For a supervised learning model, it is essential to differentiate feature variables and target variables. Predicting the correct target variable depends on the relevant feature selected for the model to train the data.

```
x = house_price_scale_df
y = house_price_df['SALES_PRICE'] # Target columns
```

Figure 47 Feature and target varable selection

3.2.10. Train test split

The dataset is split into training and testing sets before fitting the model for evaluation purposes (Joseph, 2022). The overfitting and underfitting can be determined by comparing the accuracy of the model in training and testing data. The training set is for only model training whereas the testing data set is used to evaluate the machine learning model performance on unseen data. The data set is commonly split in the ratio of 80 percent training and 20 percent testing (Joseph, 2022).

The data is split into training and testing sets by utilizing the train_test_split method from module model selection of scikit-learn library.

Splitting data into training and testing set

```
# Splitting the data into 80 percent for training and 20 percent for testing
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

Figure 48 Train test split the dataset

3.2.11 Model Initialization

In model initialization, the instance of the machine learning model is created, and its required parameter is set for training it on the preprocessed data using the Scikit learn library. The linear regression model and random forest regressor model are initialized for model training.

3.2.12. Model training

After the required machine learning model is initialized, it is trained on the training dataset so that the model can learn the hidden pattern from the dataset for correct prediction. The fit() method from the scikit learn library is used for training the instance of the machine learning model.

3.2.13 Model Evaluation

The model evaluation is the process of evaluating the performance and accuracy of the machine learning model on unseen data using different types of evaluation metrics (Fatmanurkutlu, 2024). The overfitting and underfitting of the model are also evaluated.

3.2.14. Best model selection

The performance metric or the benchmark of two different models that are trained on the same training dataset are compared and the best-performing machine learning model is determined for the house price prediction problem.

3.3 Pseudocode

START

IMPORT necessary libraries

LOAD dataset

PERFORM Data understanding

DO Data Cleaning

PERFORM removal of missing values

PERFORM duplicates value removal

PERFORM columns value text cleaning

END DO

DO Exploratory data analysis

CREATE univariate analysis include visualization

CREATE bivariate analysis including visualization

CREATE correlation analysis including heatmap

PERFORM Outlier detection including boxplot

END DO

DO Feature engineering

CREATE new feature

REMOVE unwanted feature

PERFORM one hot encoding and label encoding

PERFROM outlier removal

PERFROM feature scaling

DECLARE feature variable

DECLARE target variable

CONDUCT train test split

INITIALIZE random forest regressor model

INITIALIZE linear regression model

INITIALIZE stacking regressor model

TRAIN random forest regressor, linear regression, and stacking regressor model

GENERATE Evaluation metric for random forest classifier model, support vector classifier model and logistic regression model

EVALULATE metric

PEFROM hyperparameter tuning

COMPARE performance of the model

END

4. Result

4.1) Linear Regression

4.1.1) linear regression model initialization

The linear regression model is initialized using the scikit learn linear model package and using the method LinearRegression()

Base Model initialization (Linear Regression) ¶

```
LinearRegressionmodel = LinearRegression() #model initalization
```

Figure 49 Linear model initialization

4.1.2) linear regression model fitting the model into training data

The linear regression model is trained using the training data set by fit() method

Fitting the base model (linear regression)

```
LinearRegressionmodel.fit(X_train, y_train) #Fitting the model to traing data
```

```
▾ LinearRegression
LinearRegression()
```

Figure 50 fitting the linear regression model in train dataset

4.1.3) Linear regression model predicting the target value on test data

```
y_pred_linear_regression = LinearRegressionmodel.predict(X_test) # Predict the data on test data
r2_linear_regression = r2_score(y_test, y_pred_linear_regression) # evaluate the r2 score in test data
r2_linear_regression
```

```
0.9616053692957656
```

Figure 51 Linear regression Prediction on test data

```
y_pred_linear_regression_train = LinearRegressionmodel.predict(X_train)
r2_linear_regression = r2_score(y_train, y_pred_linear_regression_train)
r2_linear_regression
```

```
0.9611436124157611
```

Figure 52 linear regression prediction on train data

Since the r2 score in both the train and test data set is similar, the model is not over fitted.

4.1.4) Evaluation metrics for linear regression with label encoder.

```
r2_linear_regression = r2_score(y_test, y_pred_linear_regression) # R2 score
mse_linear_regression = mean_squared_error(y_test, y_pred_linear_regression) # mean square error
rmse_linear_regression = np.sqrt(mse_linear_regression) # root mean square error
mae_linear_regression = mean_absolute_error(y_test, y_pred_linear_regression) # mean absolute error
n = len(y_test)  # Number of rows
p = X_test.shape[1]  # Number of faeture variables
r2_adjusted_linear_regression = 1 - (1 - r2_linear_regression) * (n - 1) / (n - p - 1)
print(f"linear_regression  R²: {r2_linear_regression:.4f}")
print(f"linear_regression MSE: {mse_linear_regression:.4f}")
print(f"linear_regression RMSE: {rmse_linear_regression :.4f}")
print(f"linear_regression MAE: {mae_linear_regression :.4f}")
print(f"linear_regression Adjusted R²: {r2_adjusted_linear_regression:.4f}")
```

```
linear_regression  R²: 0.7665
linear_regression MSE: 2448124140752.6479
linear_regression RMSE: 1564648.2483
linear_regression MAE: 1287678.9981
linear_regression Adjusted R²: 0.7636
```

Figure 53 Evaluation of linear regression with label encoder

4.1.5) Evaluation metrics for linear regression with one hot encoder.

```
r2_linear_regression = r2_score(y_test, y_pred_linear_regression)
mse_linear_regression = mean_squared_error(y_test, y_pred_linear_regression)
rmse_linear_regression = np.sqrt(mse_linear_regression)
mae_linear_regression = mean_absolute_error(y_test, y_pred_linear_regression)
n = len(X_train)  # Number of samples
p = 1  # Number of predictors (since we have only one feature: epochs)
adjusted_r2 = 1 - (1 - r2_linear_regression) * (n - 1) / (n - p - 1)
print(f"linear regression R²: {r2_linear_regression:.4f}")
print(f"linear regression  MSE: {mse_linear_regression:.4f}")
print(f"linear regression  RMSE: {rmse_linear_regression:.4f}")
print(f"linear regression  MAE: {mae_linear_regression  :.4f}")
print(f"linear regression  adjusted R²: {adjusted_r2 :.4f}")
```

```
linear regression R²: 0.9616
linear regression  MSE: 402569879254.6489
linear regression  RMSE: 634483.9472
linear regression  MAE: 474100.9694
linear regression  adjusted R²: 0.9616
```

Figure 54 Evaluation of linear regression with one hot encoding

4.1.6 Evaluation metric of linear regression (label encoded data vs one hot encoded data)

Evaluation metric	Linear regression with label encoder	Linear regression with one hot encoder (final)
R2 score	0.7665	0.9616
Mean square error	2448124140752.6479	402569879254.6489
Root mean square error	1564648.2483	634483.9472
Mean absolute error	1287678.9981	474100.969
Adjusted R2 score	0.7636	0.9613

Using one hot encoding technique for linear regression significantly increase the model performance as it transforms the categories into binary column for each value so that model can interpret it correctly

4.1.5) Cross-validation

```
# define the number fold for the data set to conduct cross validation
cross_validation_fold = KFold(n_splits=7, shuffle=True, random_state=42)
cross_validation_score = cross_val_score(LinearRegressionmodel, X_train, y_train, cv=cross_validation_fold)
print(f"cross_validation scores: {cross_validation_score}")
print(f"Average cross_validation score: {cross_validation_score.mean():.3f} ")
```

```
cross_validation scores: [0.95989495 0.96038841 0.96027795 0.95893304 0.95856174 0.96169975
 0.96445375]
Average cross_validation score: 0.961
```

Figure 55 Linear regression cross validation score

4.2) Random Forest Regressor

4.2.1) Random Forest Regressor initialization

The random forest regressor is initially initiated with default value for the parameter for testing purposes. RandomForestRegressor() method from sklearn.ensemble module is used for initialization

Random Forest Regressor

```
random_forest_model = RandomForestRegressor()
```

Figure 56 Model initialization of random forest regressor

4.2.2) Fitting the random forest regressor model using training dataset.

After, the random forest regressor model was initialized the model was trained in training data set using fit() method with two positional arguments X_train features and y_train target values

```
[128]: random_forest_model.fit(X_train, y_train)
[128]: RandomForestRegressor
       RandomForestRegressor(random_state=42)
```

Figure 57 Training random forest regressor model in training data

4.2.3) Prediction on test data for random forest regressor

The prediction was carried out in testing data for random forest regressor model using predict () method.

Prediction of Random forest regressor

```
0]: y_pred_random_forest = random_forest_model.predict(X_test)
```

Figure 24 prediction on test data for random forest regressor

4.2.4) Evaluation for random forest regressor model

Model evaluation Random forest regressor

```
r2_random_forest = r2_score(y_test, y_pred_random_forest)
mse_random_forest = mean_squared_error(y_test, y_pred_random_forest)
rmse_random_forest = np.sqrt(mse_random_forest)
mae_random_forest = mean_absolute_error(y_test, y_pred_random_forest)
print(f"Random Forest R²: {r2_random_forest:.4f}")
print(f"Random Forest MSE: {mse_random_forest:.4f}")
print(f"Random Forest RMSE: {rmse_random_forest :.4f}")
print(f"Random Forest MAE: {mae_random_forest :.4f}")
```

```
Random Forest R²: 0.9803
Random Forest MSE: 268627461626.3336
Random Forest RMSE: 518292.8339
Random Forest MAE: 398185.2071
```

Figure 25 Model evaluation for Random Forest regressor

4.2.5 Evaluation after hyper parameter tuning

Random forest hyperparameter tunning

```
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'max_features': ['log2', 'sqrt'],
    'bootstrap': [True]

}
grid_search = GridSearchCV(estimator=random_forest_model, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)
print(f"Best Parameters: {grid_search.best_params_}")
```

```
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Best Parameters: {'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
```

After hyper tunning random forest

```
r2_random_forest = r2_score(y_test, y_pred_random_forest_tuned_train)
mse_random_forest = mean_squared_error(y_test, y_pred_random_forest_tuned_train)
rmse_random_forest = np.sqrt(mse_random_forest)
mae_random_forest = mean_absolute_error(y_test, y_pred_random_forest_tuned_train)
n = len(X_train)
p = 1
adjusted_r2_random = 1 - (1 - r2_random_forest) * (n - 1) / (n - p - 1)
print(f"Random Forest R²: {r2_random_forest:.4f}")
print(f"Random Forest MSE: {mse_random_forest:.4f}")
print(f"Random Forest RMSE: {rmse_random_forest :.4f}")
print(f"Random Forest MAE: {mae_random_forest :.4f}")
print(f"Random Forest ajusted r2: {adjusted_r2_random :.4f}")
```

```
Random Forest R²: 0.9720
Random Forest MSE: 293643655843.6227
Random Forest RMSE: 541888.9700
Random Forest MAE: 425144.4023
Random Forest ajusted r2: 0.9720
```

Figure 58 Evaluation after hyper parameter tuning random forest regressor

4.2.6 Comparison of before and after the hyper parameter tuning

Evaluation metric	Random forest regressor	After hyper parameter tunning
R2 score	0.9766	0.9720
Mean square error	2448124140752.6479	293643655843.6227
Root means square error	495783.2144	541888.9700
Mean absolute error	380388.4777	
Adjusted R2 score	0.9766	

4.3) Stacking model (Ensemble learning)

4.3.1) Base model for stacking

For the base model the linear regression and random forest regressor were initialized.

```
# Define base models for stacking
base_learners = [
    ('Random_forest', random_forest_model),
    ('linear_regression', LinearRegressionmodel)
]
```

Figure 59 Base model initialization for stacking method

4.3.2) Stacking Regressor model initialization

Stacking regressor model initialization was conducted by using the StackingRegressor() method where the estimator is both the linear regression and random forest model for training independently and final estimator is the random forest model which learn the prediction of the base model and combines it to produce the final best predictions.

StackingRegressor initialization

```
# Create the ensemble Stacking Regressor using random forest and linear regression model as estimator and
# final estimator model as random forest
stacking_regressor = StackingRegressor(estimators=base_learners, final_estimator=random_forest_model)
```

Figure 60 Stacking Regressor model initializaiton

4.3.3) Fitting the stack regressor in training dataset.

The stack regressor model is trained in training dataset with the help of fit () method

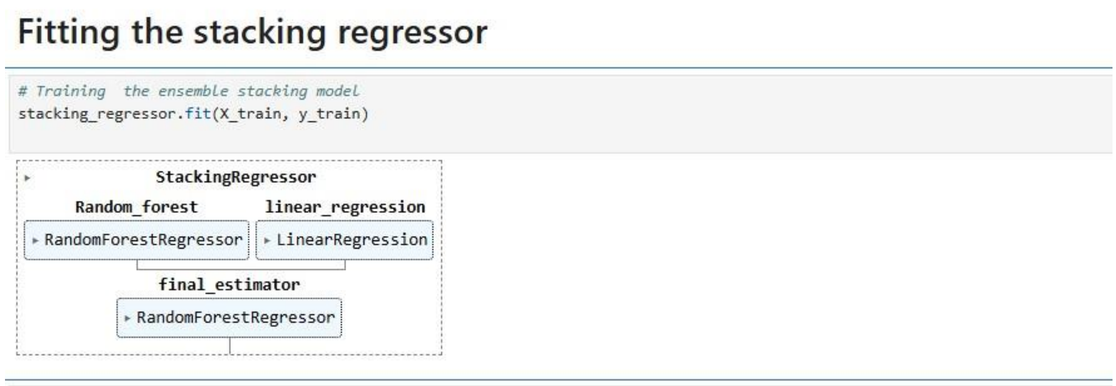

Figure 61 Fitting the stacking regressor model

4.3.4) Prediction on test data for stacking regressor

The stacking regressor was used to predict the target feature in test dataset by calling predict() method.

Figure 62 prediction on test data for stacking regressor

4.3.5) Evaluation for stacking regressor

```
r2_stacking_model = r2_score(y_test, y_pred_stacking)
mse_stacking_model = mean_squared_error(y_test, y_pred_stacking)
rmse_stacking_model = np.sqrt(mse_random_forest)
mae_stacking_model = mean_absolute_error(y_test, y_pred_stacking)
n = len(X_train)
p = 39
adjusted_r2_stack = 1 - (1 - r2_stacking_model) * (n - 1) / (n - p - 1)
print(f"Ensemble stacking model R²: {r2_stacking_model :.4f}")
print(f"Ensemble stacking model MSE: {mse_stacking_model:.4f}")
print(f"Ensemble stacking model RMSE: {rmse_stacking_model :.4f}")
print(f"Ensemble stacking modelMAE: {mae_stacking_model :.4f}")
print(f"Ensemble stacking adjusted r2 : {adjusted_r2_stack :.4f}")
```

```
Ensemble stacking model R²: 0.9885
Ensemble stacking model MSE: 120160979146.0757
Ensemble stacking model RMSE: 497376.1968
Ensemble stacking modelMAE: 257868.2779
Ensemble stacking adjusted r2 : 0.9885
```

Figure 63 Model evaluation for stacking regressor

4.4) Model Comparison

4.4.1) R2 score of all the models

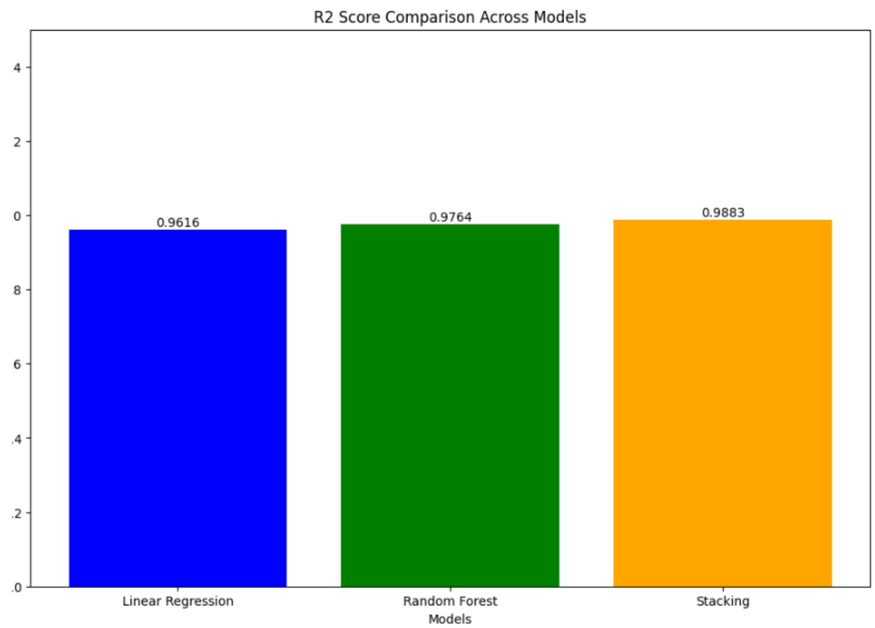

Figure 64 R2 score of all the model

4.4.2) RMSE score of all models

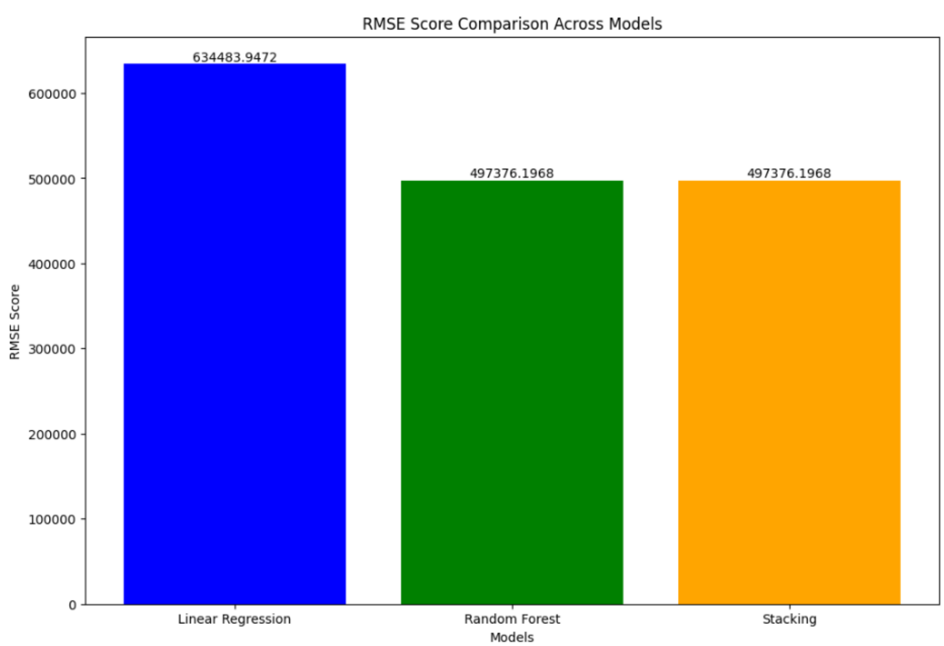

Figure 65 RMSE score of all the mo

4.4.3) MSE score of all models

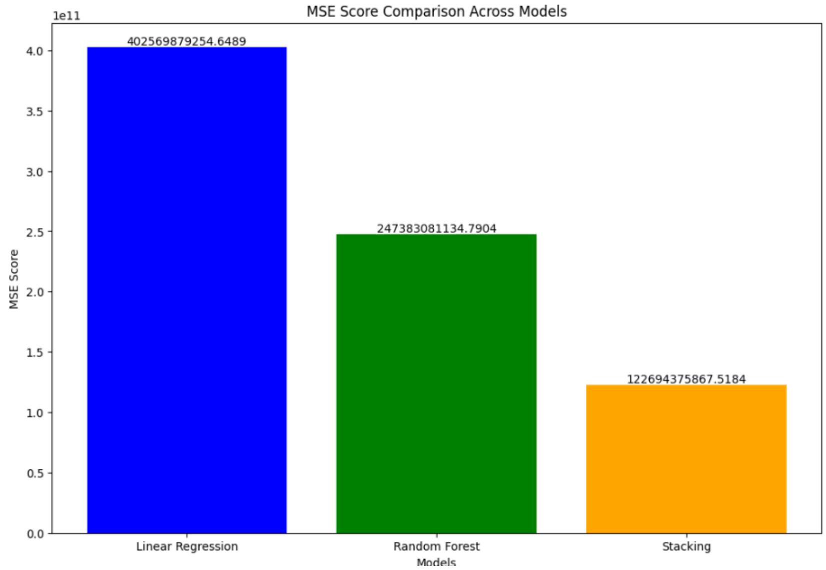

Figure 66 MSE score of all the model

4.4.4) MAE Score of all models

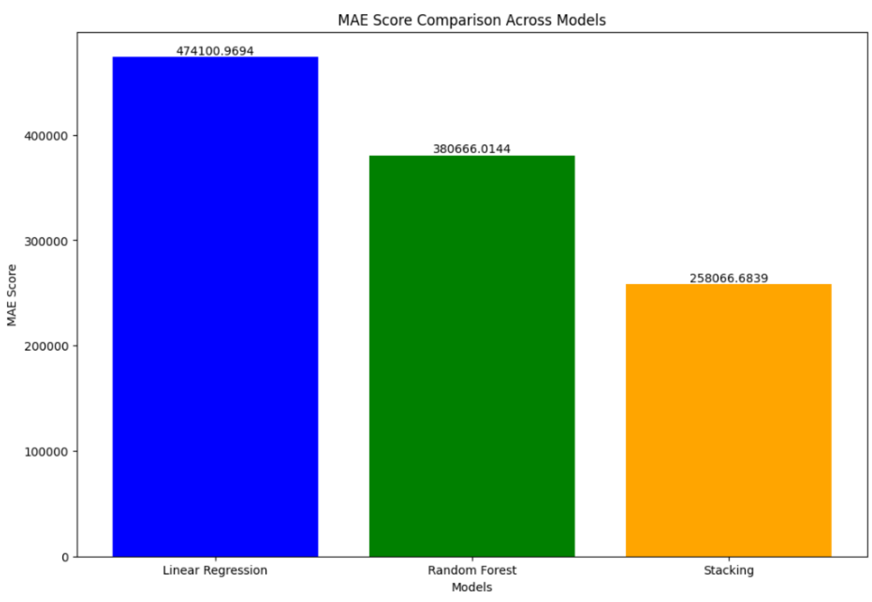

Figure 67 MAE score of all models

5) Conclusion

5.1) Analysis of work done

The linear regression model and random forest regressor along with ensemble learning stacking method was used to evaluate the performance of the model and select the best performing model. Different evaluations metric like root mean square, mean absolute error, r2 score and mean square error were used to evaluate the best model. Both Random Forest regressor and Stacking Regressor performs well on regression task as there is slight difference in evaluation metric. This is because random forest regressors was selected as a final estimate model for stacking regressors The stacking regressor adjusted r2 score is 0.9885, random forest regressor is 0.9720 and linear regression is 0.96. All the models perform well in predicting the house price in Chennai. When the label encoding was used the performance of the linear regression was about 0.76, after the application of one hot encoding the performance boosted to 0.96 which indicate the importance of one hot encoding for categorical data for nominal data.

RMSE, mean absolute error and mean square error score of random forest regressors and stacking regressors was almost same and less than the linear regressor. Which suggests better model performance in prediction as the difference between the actual and predict value is comparable lower than the linear regressor model.

5.2) Application addressing real world problem and further work

The better model performance for predicting the house price of the Chennai by considering different factor like bedroom sale condition, utilities etc. can help the potential buyer to estimate the price of the house which enables better decision making. The property buyer can prevent from fraudulent deal by estimating if the price of the house is overpriced or underpriced. The banking and real estate agent can use the machine learning approach to estimate the price and invest in right property, bank can assess the house valuation for the loan approval or credit approval to the customer to mitigate the risk.

Further work can include, creating the API with friendly user interface to estimate the house price, improving the model with the real time correct data so it performs well in future.

6) References

AnalytixLabs, 2023. *Random Forest Regression — How it Helps in Predictive Analytics?.* [Online]
Available at: https://medium.com/@byanalytixlabs/random-forest-regression-how-it-helps-inpredictive-analytics-01c31897c1d4 [Accessed 22 December 2024].

Brital, A., 2021. *Random Forest Algorithm Explained.* [Online] Available at: https://anasbrital98.github.io/blog/2021/Random-Forest/ [Accessed 22 December 2024].

Fatbardha Maloku, B. M. a. A. A. D. K., 2024. House Price Prediction Using Machine Learning and Artificial. *Journal of Artificial Intelligence & cloud computing,* 3(4), p. 1.

Fatmanurkutlu, 2024. *Model Evaluation Techniques in Machine Learning.* [Online] Available at: https://medium.com/@fatmanurkutlu1/model-evaluation-techniques-in-machinelearning-8cd88deb8655 [Accessed 22 Dec 2024].

geeksforgeeks, 2024. *Linear Regression in Machine learning.* [Online]
Available at: https://www.geeksforgeeks.org/ml-linear-regression/
[Accessed 22 December 2024].

Joseph, V. R., 2022. *Optimal ratio for data splitting.* [Online] Available at:
https://www.researchgate.net/publication/359714114_Optimal_ratio_for_data_splitting
[Accessed 22 December 2024].

Mao, T., 2023. Real Estate Price Prediction Based on Linear Regression and. *Real Estate Price Prediction Based on Linear Regression and,* Volume 38, p. 400.

Mfazi, S., 2022. *16-Days Challenge: Personal House Price Prediction Project - Beginner.* [Online]
Available at: https://www.linkedin.com/pulse/16-days-challenge-personal-house-priceprediction-project-mfazi/
[Accessed 22 December 2024].

Yaping Zhao, J. Z. a. E. Y. L., 2024. House Price Prediction: A Multi-Source Data Fusion Perspective. *BIG DATA MINING AND ANALYTICS,* 7(3), p. 603.

Abhigyan, 2020. *R-Squared and Adjusted R-Squared.* [Online]
Available at: https://medium.com/analytics-vidhya/r-squared-and-adjusted-r-squared-408aaca84fd5
[Accessed 14 January 2024].
Salman, M., 2024. *Linear Regression for Machine Learning: A Practical Approach.* [Online]
Available at: https://medium.com/@mahnoorsalman96/linear-regression-for-machine-learning-a-practical-approach-84e447afa188
[Accessed 14 January 2025].
Soni, B., 2023. *Stacking to Improve Model Performance: A Comprehensive Guide on Ensemble Learning in Python.* [Online]
Available at: https://medium.com/@brijesh_soni/stacking-to-improve-model-performance-a-comprehensive-guide-on-ensemble-learning-in-python-9ed53c93ce28
[Accessed 14 January 2024].
Verma , D., 2023. *R2 Score: Linear Regression.* [Online]
Available at: https://medium.com/@deependra.verma00/r2-score-linear-regression-e095a1188e87
[Accessed 14 January 2024].