MTH - 209: Data Science Lab 2

Guide: Dr. Subhajit Dutta

GROUP STATISTICA:

231080056 - MAYURESH MANDAR JOSHI

231080042 - HEMANT SANJAY NIKHADE

231080109 - VADTHYA VISHNU VARDHAN

231080082 - SHIVAM CHAURASIA

220565 - KUNAL DAYMA

A Project Report on

HOUSE PRICE PREDICTION

Department of Statistics, Indian Institute of Technology, Kanpur

TABLE OF CONTENT

Acknowledgement

We are grateful to our professor, **Dr. Subhajit Dutta**, for giving us the opportunity to work on the Data Science project on "**House Price Prediction**". With Dr. Subhajit Dutta guidance, we were able to complete the project on time and with our best efforts. This experience has been incredible for our team, allowing us to learn and work together collaborately.

1. Introduction

1.1 About House Price

House prices are influenced by factors such as location, size, amenities, and market trends. Understanding local demand, economic conditions, and demographic shifts is crucial. Factors like interest rates and housing policies also impact pricing. Accurate prediction requires robust models incorporating these variables to provide valuable insights for buyers, sellers, and investors.

The market for Houses is characterized by:

• Economic Conditions: Factors such as GDP growth, employment rates, and income levels impact housing affordability and demand.

• Demographic Trends: Changes in population size, age distribution, and household composition influence housing preferences and demand.

• Government Policies: Regulations related to zoning, taxation, and mortgage lending affect housing supply and affordability.

• Real Estate Developers and Agents: Entities involved in property development, sales, and marketing play a significant role in shaping the housing market.

1.2 Problem Statement

In this project, the focus is on developing a Model to assist consumers in finding suitable House for their living. By leveraging data science techniques, the project aims to curate a comprehensive final dataset encompassing essential attributes, including price, location, size, total sqft, bathrooms, bhk, Price Per Square Feet, and other relevant factors. This initiative is aimed at reducing the time and effort consumers spend , allowing them to access a consolidated dataset and a website expedites their decision-making process and enhances their overall experience.

1.3 Project Aim and Objective

• People looking to buy a new home tend to be more conservative with their budgets and marketstrategies.

• This project aims to analyse various parameters like average income, average area etc. andpredict the house price accordingly.

• This application will help customers to invest in an estate without approaching an agent To provide a better and fast way of performing operations.

• To provide proper house price to the customers.

• To eliminate need of real estate agent to gain information regarding house prices.

• To provide best price to user without getting cheated.

• To enable user to search home as per the budget.

• The aim is to predict the efficient house pricing for real estate customers with respect totheir budgets and priorities. By analyzing previous market trends and price ranges, and alsoupcoming developments future prices will be predicted.

• House prices increase every year, so there is a need for a system to predict house prices in the future.

• House price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house.

• We use linear regression algorithm in machine learning for predicting the house price trend.

2. About our dataset

2.1 Source of our data collection

https://www.kaggle.com/datasets/amitabhajoy/bengaluru-house-price-data

The dataset used for this model contains various features that influence house prices, including location, number of bedrooms and bathrooms, house size, and amenities. Regression algorithms are employed to 2 analyze this data and predict house prices accurately

2.2 Data Set Variables

Categorical Variables:

1) **area_type**: This variable likely represents the type of area the property is located in, such as “Super built-up Area”, “Built-up Area”, “Plot Area”, etc. It is categorical because it represents different categories or types of areas.

2) **availability**: This variable may indicate the availability status of the property, such as “Ready To Move”, “Immediate Possession”, or specific dates. It is categorical because it represents different availability categories.

3) **location**: This variable likely represents the location or area where the property is situated, such as “Whitefield”, “Electronic City”, “Sarjapur Road”, etc. It is categorical because it represents different geographical locations.

4) **size**: This variable may represent the size or dimensions of the property in terms of the number of bedrooms, such as “1 BHK”, “2 BHK”, “3 BHK”, etc. It is categorical because it represents different categories based on the number of bedrooms.

5) **society**: This variable might indicate the name or code of the housing society or community where the property is located. It is categorical because it represents different society names or codes.

Numerical Variables:

1) **total_sqft**: This variable likely represents the total area of the property in square feet. It is numerical because it represents continuous values.

2) **bath**: This variable may represent the number of bathrooms in the property. It is numerical because it represents discrete values.

3) **balcony**: This variable may represent the number of balconies in the property. It is numerical because it represents discrete values.

4) **price**: This variable likely represents the price of the property. It is numerical because it represents continuous value

TECHNOLOGY AND TOOLS USED IN PROJECT:

1 Python

2 Numpy and Pandas for data cleaning

3 Matplotlib for data visualization

4 Sklearn for model building

5 Jupyter notebook, visual studio code and pycharm as IDE

6 Python flask for http server

7 HTML/CSS/Java script for UI

LINEAR ALGEBRA, STATISTICS AND PROBABILITY USED:

- Linear Regression
- One Hot encoding
- K – fold cross validation
- Decision tree Regressor
- Lasso Regression
- Gridsearch CV
- Mean, Standard deviation and Basic Statistics
- Correlation, Histogram, Scatterplot, Barplot

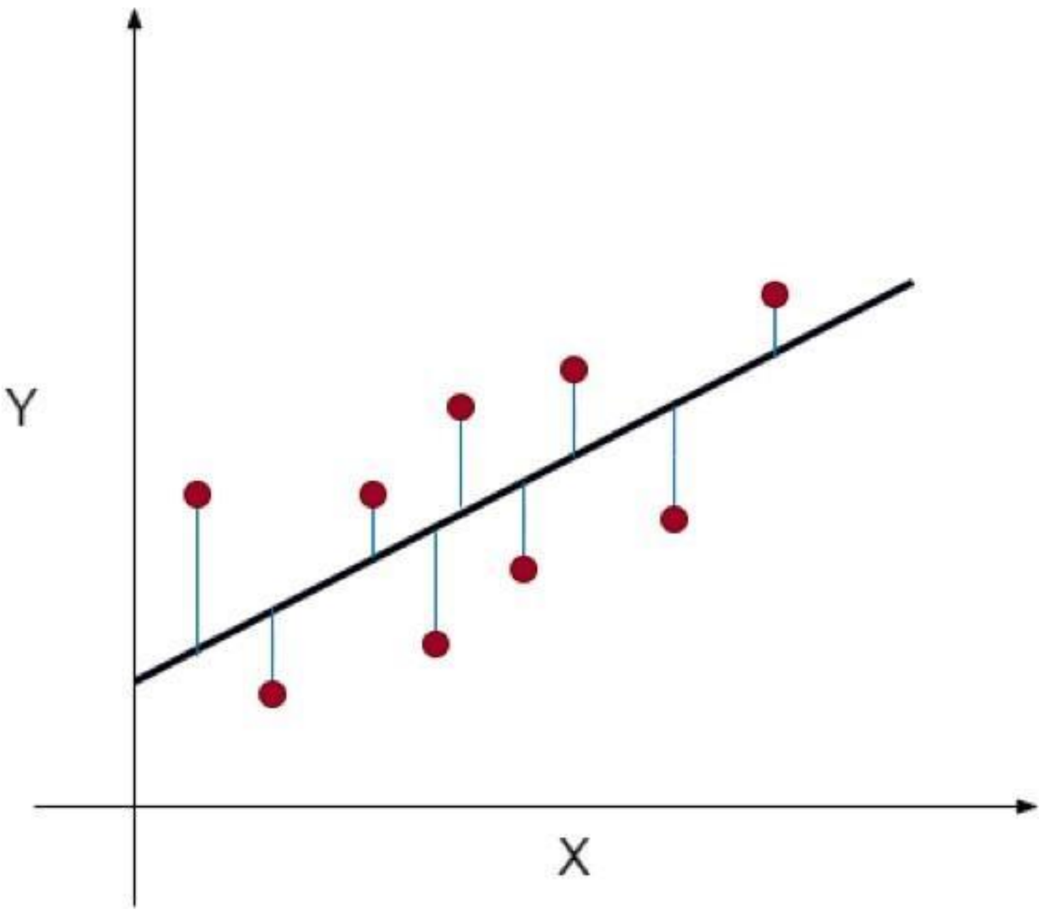
PROPOSED SYSTEM

Linear Regression is a supervised machine learning model that attempts to model a linear relationship between dependent variables (Y) and independent variables (X).

Every evaluated observation with a model, the target (Y)'s actual value is compared to the target (Y)'s predicted value, and the major differences in these values are called residuals. The Linear Regression model aims to minimize the sum of all squared residuals. Here is the mathematical representation of the linear regression:

$$Y= a_0+a_1X+ \varepsilon$$

The values of X and Y variables are training datasets for the model representation of linear regression. When a user implements a linear regression, algorithms start to find the best fit line using $\mathbf{a_0}$ and $\mathbf{a_1}$. In such a way, it becomes more accurate to actual data points; since we recognize the value of $\mathbf{a_0}$ and $\mathbf{a_1}$**,** we can use a model for predicting the response. Linear regression is widely used in various fields, including finance, economics, healthcare, and social sciences, for tasks such as predicting stock prices, estimating the impact of advertising on sales, and analyzing the relationship between variables. Despite its simplicity, linear regression remains a powerful and interpretable method for predictive modeling and inference



- As you can see in the above diagram, the red dots are observed values for both X and Y.
- The black line, which is called a line of best fit, minimizes a sum of a squared error.
- The blue lines represent the errors; it is a distance between the line of best fit and observed values.
- The value of the $\mathbf{a_1}$is the slope of the black line.

One Hot Encoding

- One-hot encoding is a technique used in machine learning to represent categorical variables numerically. In datasets, variables like colors, countries, or product types are categorical, meaning they have discrete values with no inherent order. However, many machine learning algorithms require numerical input.
- One-hot encoding solves this by creating binary columns for each unique category in the original variable.
- Each observation is then represented by a binary vector where only one element is 1 (indicating the presence of that category) and all others are 0. For example, if a variable has three categories—such as "red," "blue," and "green"—one-hot encoding would create three binary columns, each representing one of these categories.
- This transformation enables algorithms to interpret categorical data correctly without implying false relationships between categories.
- It's important to note that while one-hot encoding provides a straightforward solution to handle categorical variables, it can lead to high-dimensional feature spaces, increasing computational complexity.
- Additionally, it's crucial to handle large categorical variables and multicollinearity issues that may arise.
- Despite these challenges, one-hot encoding is widely used and effective for preprocessing categorical data in machine learning tasks.

K-fold cross-validation

- K-fold cross-validation is a method used to evaluate the performance of a machine learning model by partitioning the dataset into k subsets or "folds."
- The model is trained on k-1 folds and validated on the remaining fold. This process is repeated k times, with each fold serving as the validation set exactly once.
- The performance metrics, such as accuracy or mean squared error, are averaged over the k iterations to obtain a robust estimate of the model's performance.
- K-fold cross-validation helps assess the model's generalization ability and reduces the variance in performance estimation compared to a single train-test split.
- It's particularly useful for small datasets where splitting into distinct training and testing sets may lead to unreliable performance estimates.
- Additionally, it ensures that each observation is used for both training and validation, maximizing the utilization of available data for model evaluation.

Lasso regression

- Lasso regression, short for Least Absolute Shrinkage and Selection Operator regression, is a regularization technique used in machine learning to prevent overfitting and perform feature selection.
- It extends traditional linear regression by adding a penalty term to the ordinary least squares (OLS) objective function.
- This penalty term is the sum of the absolute values of the coefficients multiplied by a regularization parameter (λ), also known as alpha. By penalizing the absolute size of the coefficients, lasso regression encourages sparsity in the model, meaning it forces some coefficients to be exactly zero.
- Consequently, lasso regression performs automatic feature selection by identifying and discarding irrelevant or less important features from the model.
- This property makes it particularly useful for datasets with a large number of features, where selecting the most relevant features can improve model performance and interpretability.
- The choice of the regularization parameter (λ) is crucial and typically determined through techniques such as cross-validation. Lasso regression is widely used in various domains, including finance, healthcare, and natural language processing, where interpretability and feature selection are important considerations.

Decision Tree Regressor

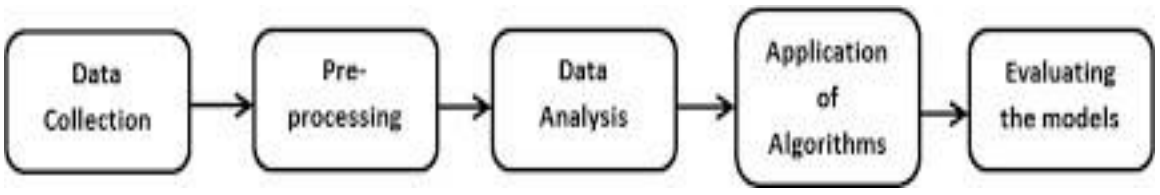
- A Decision Tree Regressor is a machine learning algorithm used for regression tasks, where the goal is to predict a continuous target variable. It operates by recursively partitioning the feature space into regions, guided by the values of input features, to create a tree-like structure. At each node of the tree, the algorithm selects the feature and the split point that maximizes the reduction in variance of the target variable. This process continues until a stopping criterion is met, such as reaching a maximum depth or minimum number of samples per leaf.
- The resulting tree structure can be visualized, making it easy to interpret and understand the decision-making process. To make predictions, new data points traverse the tree from the root node to the leaf node, and the average target value of the training instances in that leaf node is used as the prediction.
- Decision Tree Regressors are versatile and can handle both numerical and categorical features without requiring feature scaling. However, they are

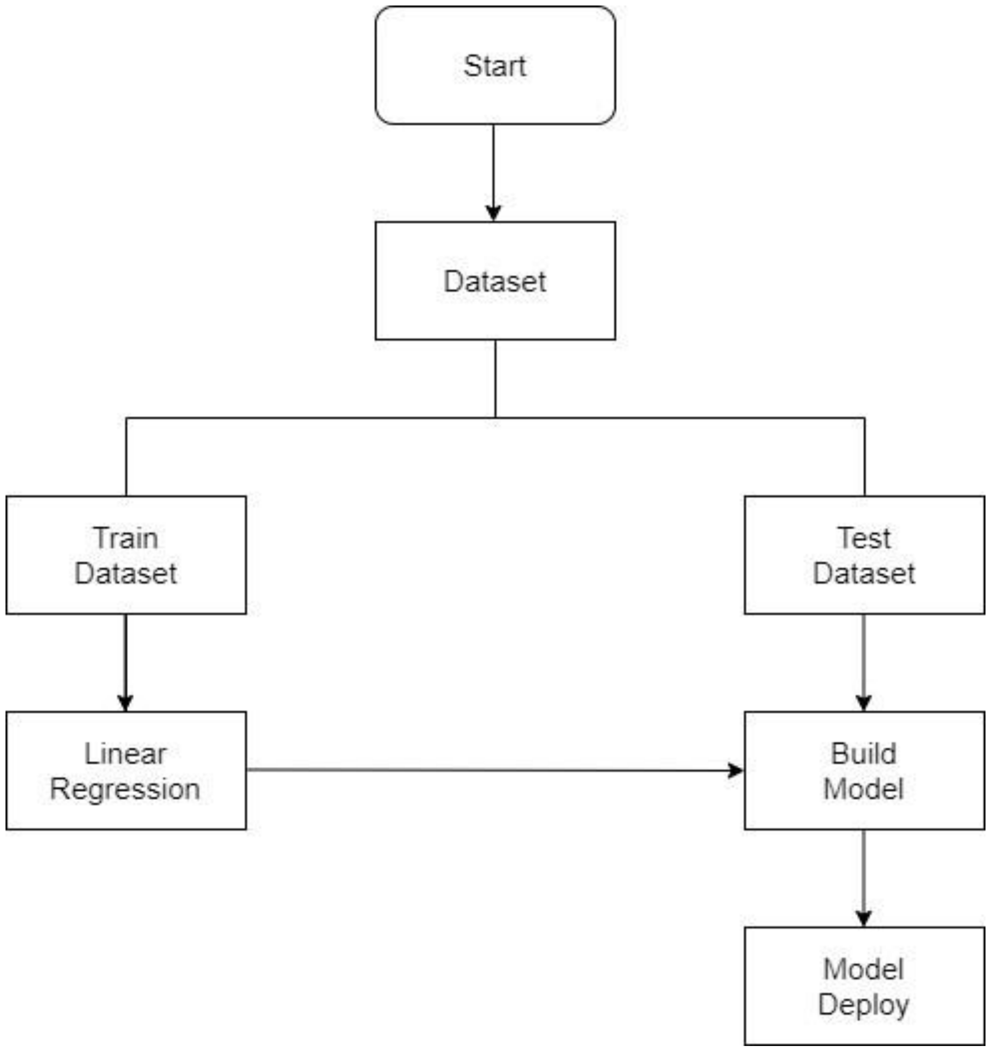
prone to overfitting, especially when the tree is deep. Techniques like pruning, limiting the maximum depth, and using ensemble methods like Random Forest can help mitigate this issue. Decision Tree Regressors are widely used in various domains, including finance, healthcare, and manufacturing, for their simplicity, interpretability, and effectiveness in capturing nonlinear relationships in data.

Gridsearch CV

- GridSearchCV is a technique used in machine learning to tune hyperparameters by exhaustively searching through a specified grid of parameter values. It is implemented as part of the scikit-learn library in Python and is widely used for optimizing models.
- The process involves defining a grid of hyperparameters to be tuned and specifying the possible values for each hyperparameter. GridSearchCV then evaluates the model performance using cross-validation for each combination of hyperparameters in the grid. By default, it uses k-fold cross-validation, where the dataset is divided into k subsets (or folds), and the model is trained and evaluated k times, each time using a different fold as the validation set.
- GridSearchCV selects the combination of hyperparameters that results in the best performance, typically measured by a specified evaluation metric such as accuracy, precision, recall, or F1 score. Once the best hyperparameters are determined, the final model is trained on the entire dataset using these optimal values.
- GridSearchCV helps automate the process of hyperparameter tuning, saving time and effort compared to manual tuning. However, it can be computationally expensive, especially for large datasets or models with many hyperparameters. Techniques like randomized search (RandomizedSearchCV) or Bayesian optimization (BayesSearchCV) can be used as alternatives to GridSearchCV to improve efficiency in hyperparameter tuning.

BLOCK DIAGRAM





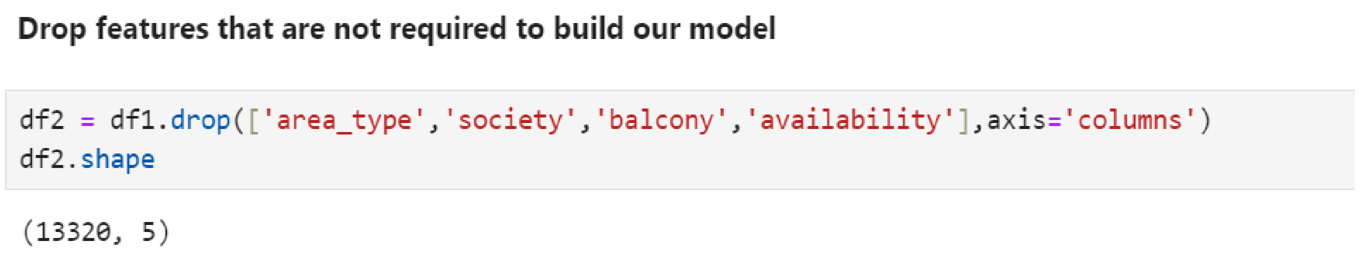
There are 13320 observations in our dataset. There are a total of 9 columns/attributes in our dataset. The all 9 columns are area type, availability, location, size, total sqft, bath, society, balcony and price.

Data Cleaning:

The initial phase of our project involved rigorous data cleaning to ensure the quality and integrity of the dataset before proceeding with further analysis and modeling. In this section, we detail the steps taken to clean the dataset, including the removal of unnecessary columns, treatment of missing values, and handling of outliers.

Removal of Unnecessary Columns:

We began by identifying columns that did not contribute significantly to the analysis or were redundant. Utilizing the following code snippet, we removed these columns from the dataset:



Drop features that are not required to build our model

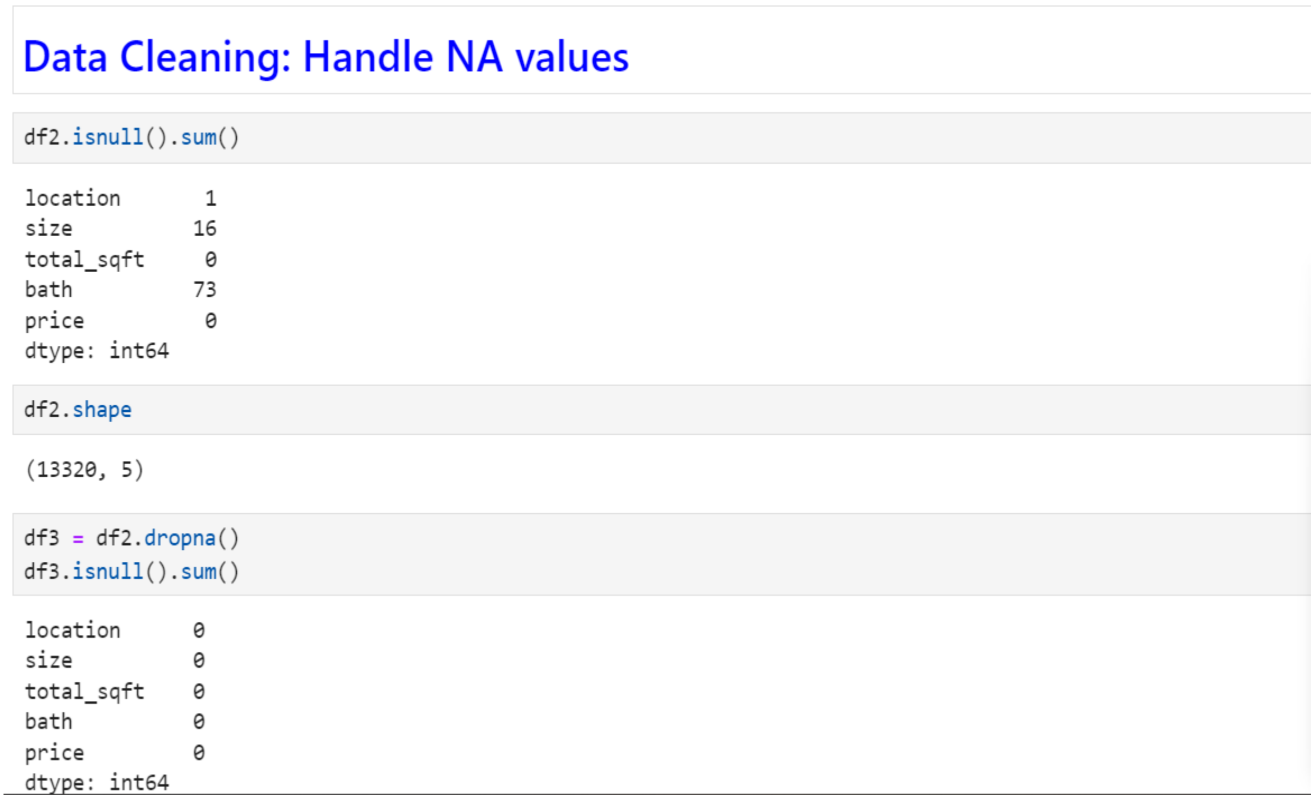
```
df2 = df1.drop(['area_type','society','balcony','availability'],axis='columns')
df2.shape
```

```
(13320, 5)
```

The removal of these columns streamlined the dataset and eliminated potential sources of noise in our analysis.

Treatment of Missing Values:

Missing values pose a challenge to data analysis, potentially skewing results if left unaddressed. To handle missing values, we employed the following code snippet:


Data Cleaning: Handle NA values

```
df2.isnull().sum()
```

```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

```
df2.shape
```

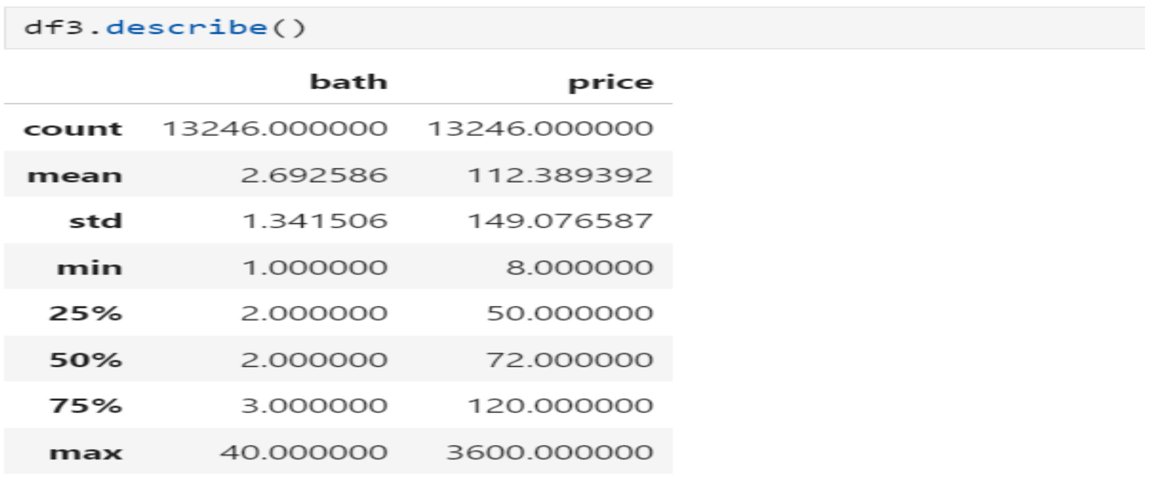
```
(13320, 5)
```

```
df3 = df2.dropna()
df3.isnull().sum()
```

```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

This simple yet effective approach involved removing rows with missing values, ensuring that our analysis was based on complete data points.

After every step we did basic exploratory data analysis such that finding mean and other basic statistics

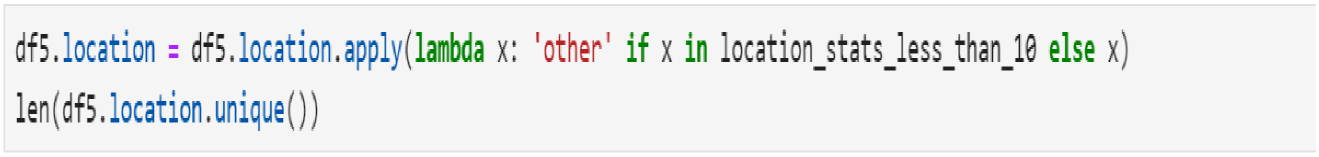


```
df3.describe()
```

	bath	price
count	13246.000000	13246.000000
mean	2.692586	112.389392
std	1.341506	149.076587
min	1.000000	8.000000
25%	2.000000	50.000000
50%	2.000000	72.000000
75%	3.000000	120.000000
max	40.000000	3600.000000

Dimensionality Reduction:

can be reduced by huge amount. Later on when we do one hot encoding, it will help us with having fewer dummy columns Any location having less than 10 data points should be tagged as "other" location. This way number of categories.

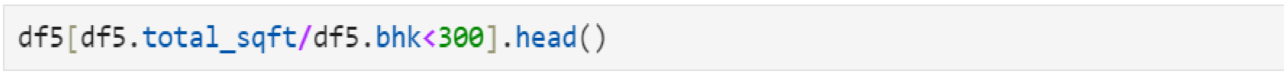


```
df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
len(df5.location.unique())
```

```
241
```

Outlier Removal Using Business Logic:

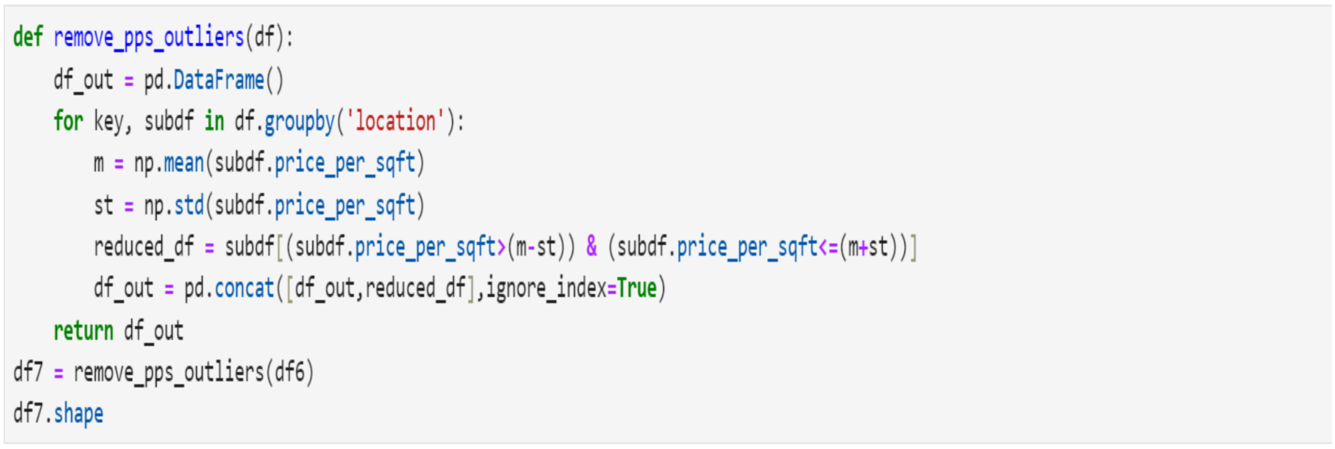
As a data scientist when you have a conversation with your business manager (who has expertise in real estate), he will tell you that normally square ft per bedroom is 300 (i.e. 2 bhk apartment is minimum 600 sqft. If you have for example 400 sqft apartment with 2 bhk than that seems suspicious and can be removed as an outlier. We will remove such outliers by keeping our minimum thresold per bhk to be 300 sqft.



```
df5[df5.total_sqft/df5.bhk<300].head()
```

Outlier Removal Using Standard Deviation and Mean:

Here we find that min price per sqft is 267 rs/sqft whereas max is 12000000, this shows a wide variation in property prices. We should remove outliers per location using mean and one standard deviation

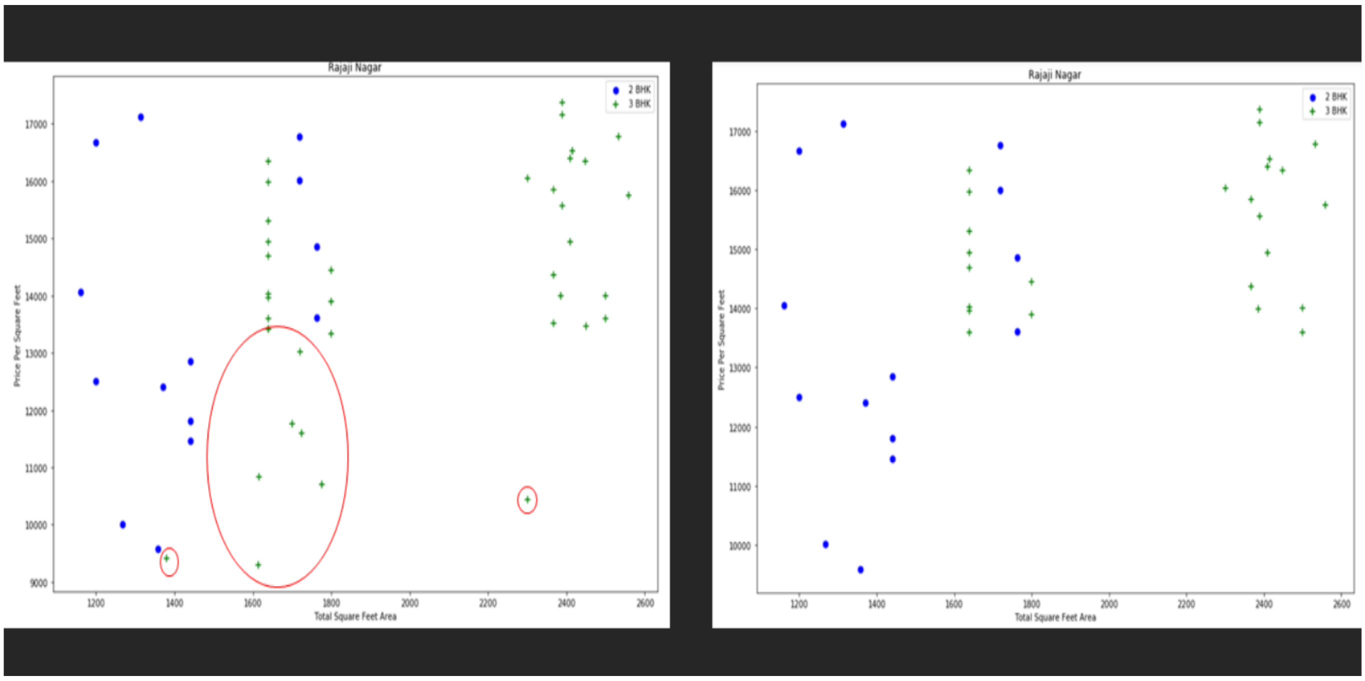


```
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key, subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)
        st = np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m+st))]
        df_out = pd.concat([df_out,reduced_df],ignore_index=True)
    return df_out
df7 = remove_pps_outliers(df6)
df7.shape
```

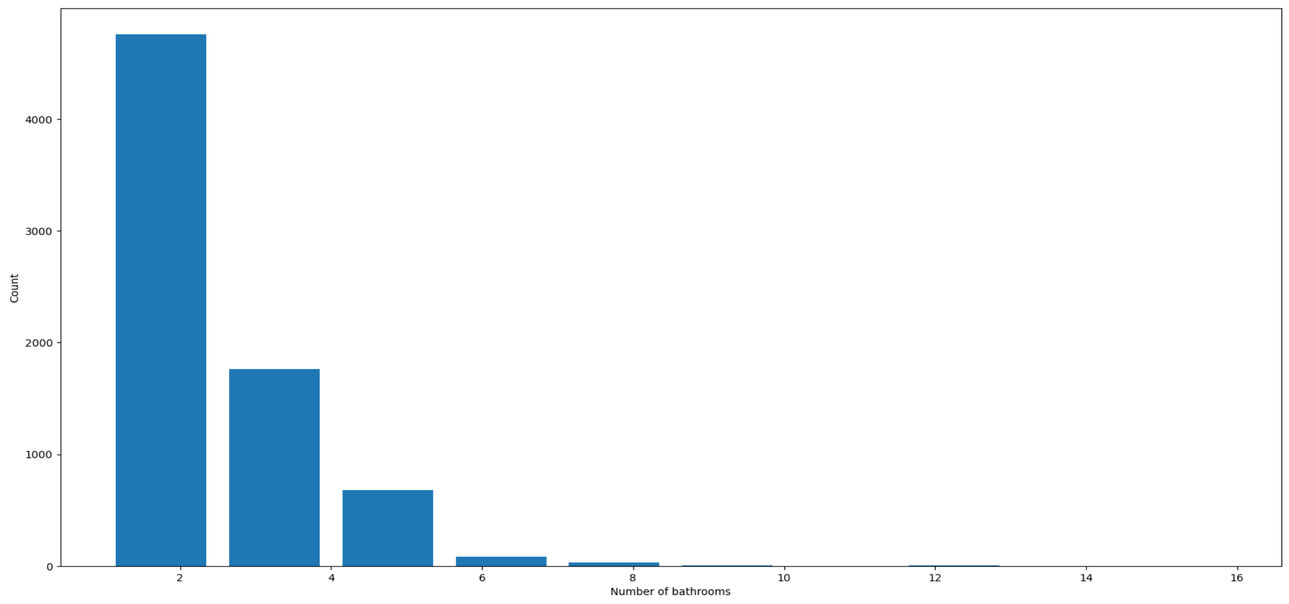
```
(10242, 7)
```

We should also remove properties where for same location, the price of (for example) 3 bedroom apartment is less than 2 bedroom apartment (with same square ft area). So what we did is for given location we build dictionary of stats per bhk so we can remove those 2 BHK apartments whose price_per_sqft is less than mean price_per_sqft of 1 BHK apartment.

Before and after outlier removal: Rajaji Nagar



It is unusual to have more bathrooms than the no of rooms, if you have 4 bedroom home and even if you have bathroom in all 4 rooms plus one guest bathroom, you will have total bath = total bed + 1 max. Anything above that is an outlier or a data error and can be removed.



```
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
```

```
(7239, 7)
```

Why not used PCA for dimensionality reduction:

During the data preprocessing phase, one common technique for reducing the dimensionality of the dataset is Principal Component Analysis (PCA). However, after careful consideration, we chose not to employ PCA for dimensionality reduction in our project.

As a statistician, the decision not to utilize PCA stemmed from a thorough assessment of our dataset's characteristics and the specific requirements of our analysis. Several factors influenced this decision, outlined below:

Feature Importance

Our dataset comprises a relatively small number of features, each of which contributes significantly to the predictive task at hand. Through exploratory data analysis and domain knowledge, we determined that all features were relevant and carried valuable information for modeling house prices in Bangalore.

Preservation of Information

PCA operates by projecting the original features onto a new orthogonal basis, represented by principal components. While this technique can effectively reduce dimensionality by capturing the maximum variance in the data, it inherently involves a loss of information. Given the importance of each feature in our dataset, we prioritized the preservation of information and interpretability over dimensionality reduction.

Interpretability and Transperency

By retaining all original features without dimensionality reduction, our analysis maintains transparency and interpretability. Each feature retains its inherent meaning and can be readily interpreted in the context of the problem domain, facilitating clearer insights and actionable conclusions.

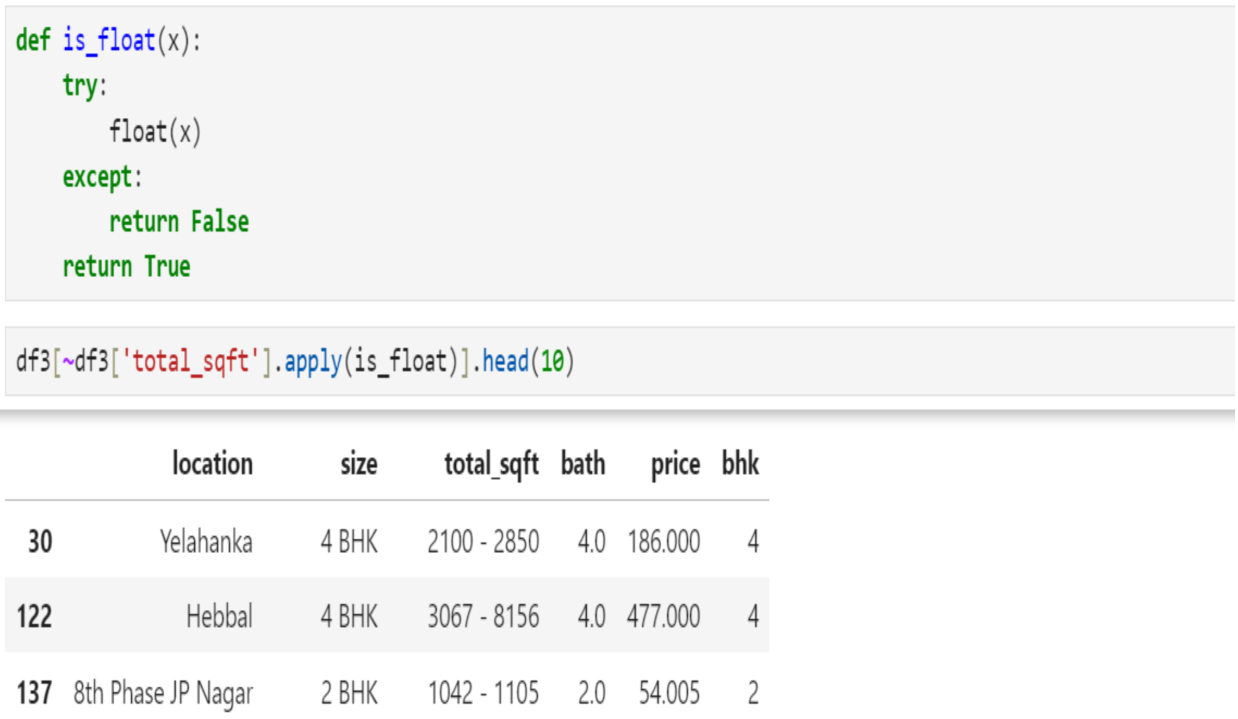
Instead of employing PCA for dimensionality reduction, we opted for a more straightforward approach to feature selection and data cleaning. This involved the careful identification and removal of redundant or irrelevant features through exploratory analysis and domain expertise.

Feature Engineering:

We added new feature integer for bhk and price per sqft.

In the process of feature engineering, we sought to enhance the predictive power of our model by deriving new features from the existing dataset. One key feature we introduced was the 'BHK' (Bedrooms, Hall, and Kitchen) feature, which represents the total number of bedrooms and living spaces in each property. This aggregated feature provides a more holistic view of the property's size and layout, capturing essential information for price prediction.

Additionally, we engineered the 'Price per Sqft' feature, calculated by dividing the total price of the property by its total square footage. This feature offers insights into the pricing dynamics based on the area of the property, enabling more granular analysis and prediction.

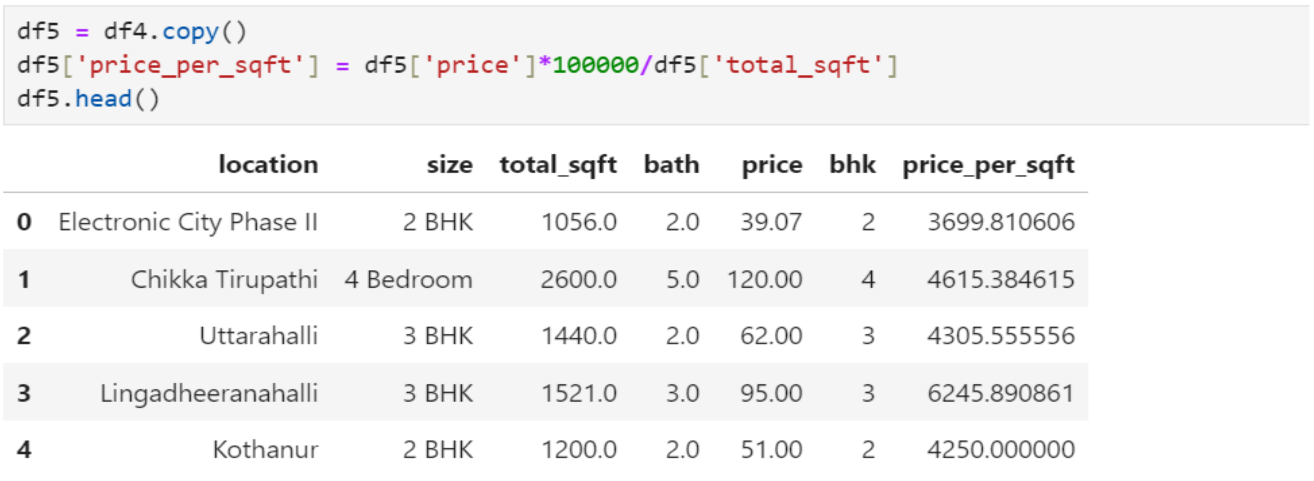


```python
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

```python
df3[~df3['total_sqft'].apply(is_float)].head(10)
```

	location	size	total_sqft	bath	price	bhk
30	Yelahanka	4 BHK	2100 - 2850	4.0	186.000	4
122	Hebbal	4 BHK	3067 - 8156	4.0	477.000	4
137	8th Phase JP Nagar	2 BHK	1042 - 1105	2.0	54.005	2

Above shows that total_sqft can be a range (e.g. 2100-2850). For such case we can just take average of min and max value in the range. There are other cases such as 34.46Sq. Meter which one can convert to square ft using unit conversion. We just dropped such corner cases to keep things simple.



```
df5 = df4.copy()
df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
df5.head()
```

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250.000000

Exploratory Data Analysis (EDA):

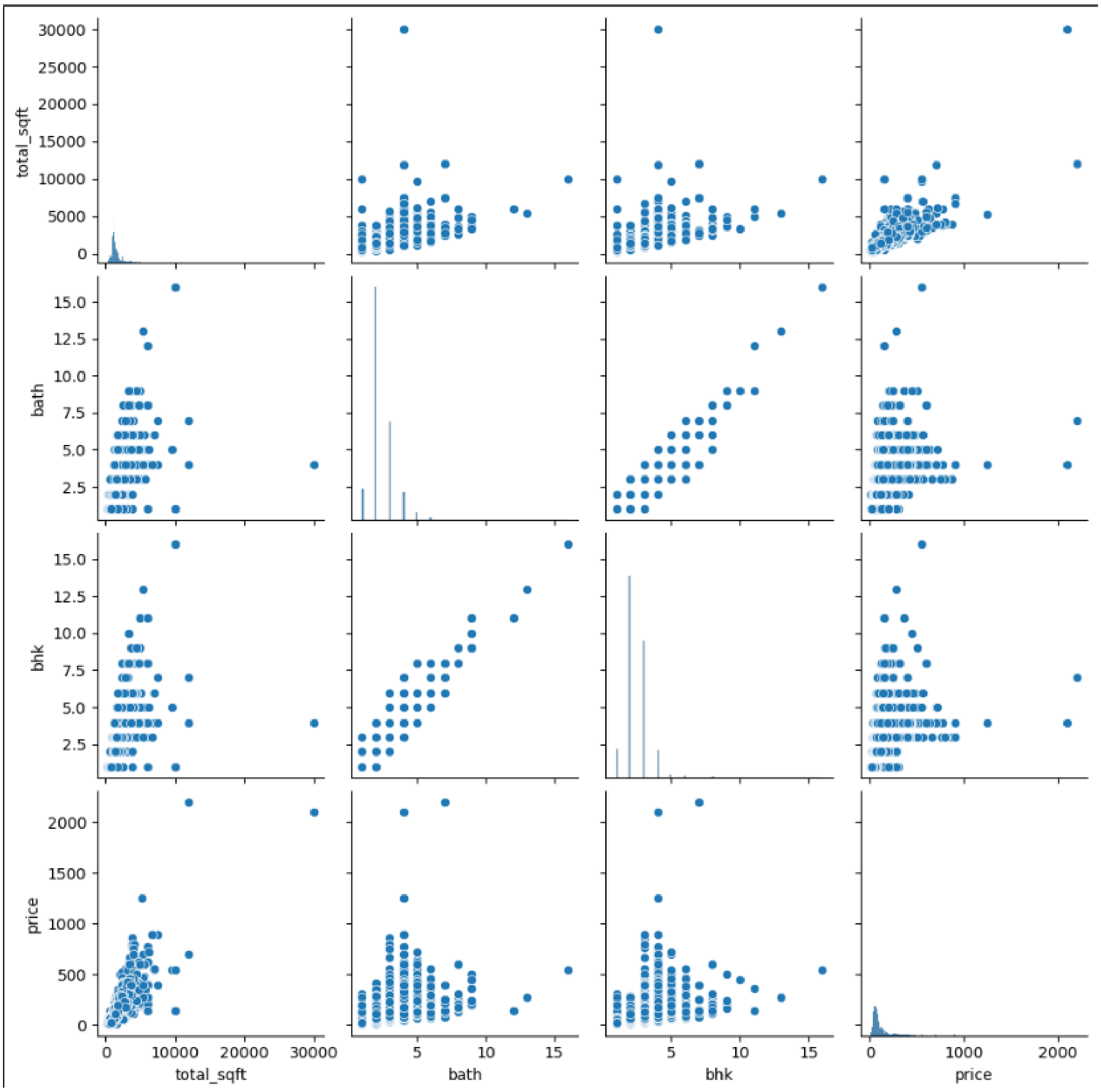
Exploratory Data Analysis (EDA) is a crucial step in understanding the underlying patterns and relationships within the dataset. In this section, we delve into two fundamental visualization techniques employed during EDA: scatter plots and correlation matrices.

Scatter Plot:

A scatter plot is a simple yet powerful visualization tool used to explore the relationship between two continuous variables. In our project, scatter plots were instrumental in understanding the association between various features and the target variable, i.e., house prices in Bangalore.

Purpose:

The primary purpose of using scatter plots in our analysis was to identify any discernible patterns or trends between independent variables (features) and the dependent variable (house prices). By plotting house prices against individual features, we could visually inspect the nature of their relationship, whether it be linear, non-linear, or lack thereof.



- The scatter plot analysis reveals compelling insights into the relationships between various features. Notably, all features exhibit a linear relationship with each other, indicating a strong correlation.
- Particularly striking is the high linearity observed between the number of bedrooms (BHK) and bathrooms, suggesting a proportional increase in bathrooms with an increase in the number of bedrooms.
- The price demonstrates a clear linear relationship with other factors, underscoring their impact on property valuation.

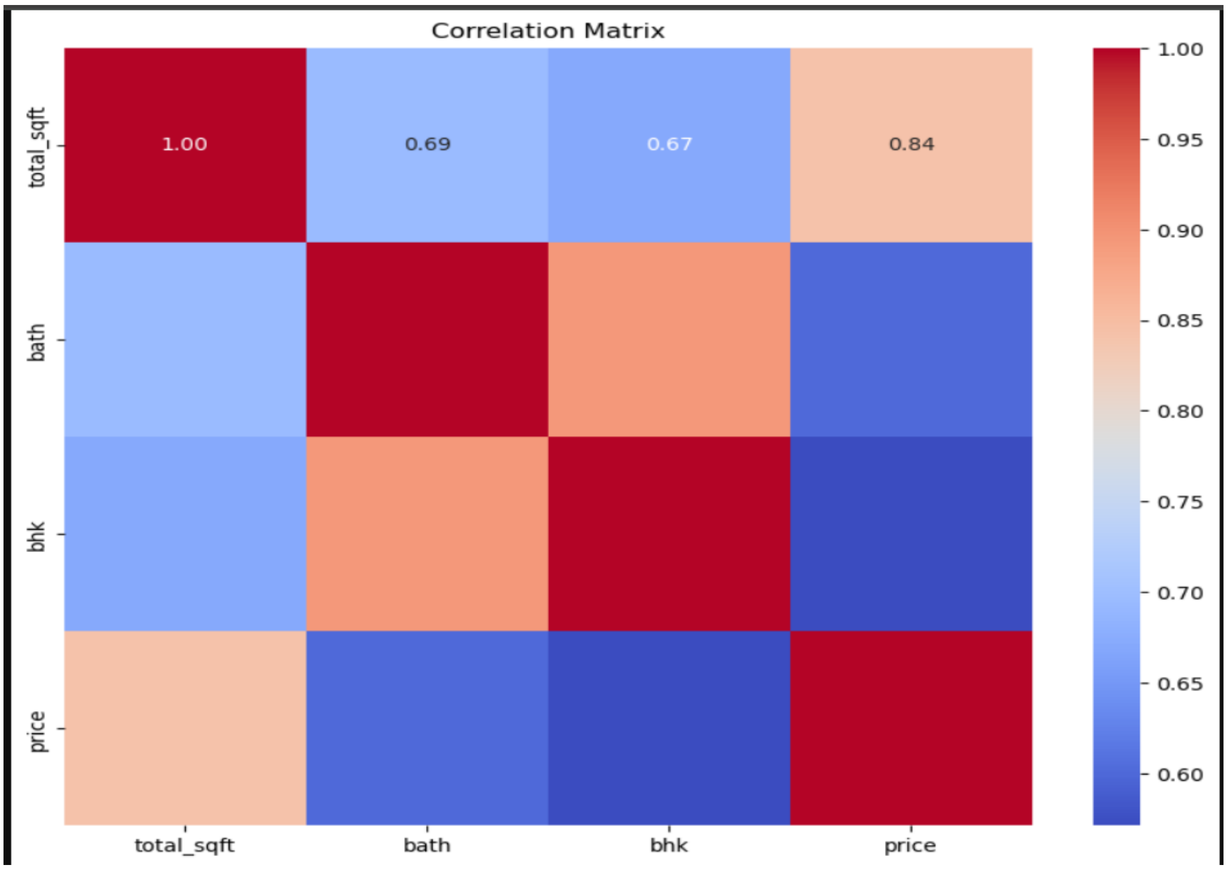
- This comprehensive understanding of linear relationships underscores the importance of these features in predicting house prices accurately, guiding informed decision-making processes in the real estate domain.

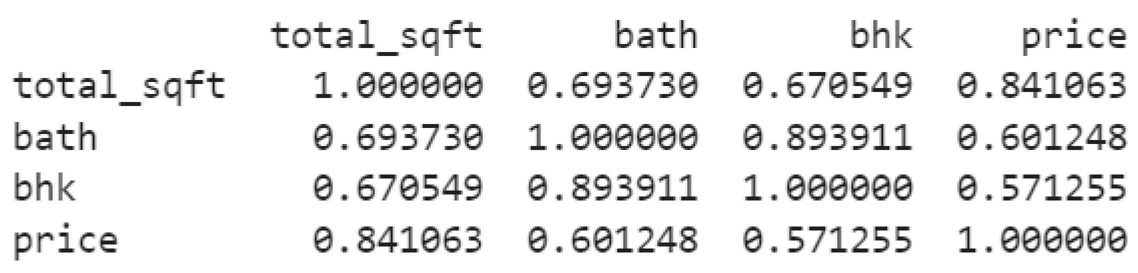
Correlation Matrix:

A correlation matrix is a tabular representation of the pairwise correlations between variables in the dataset. It provides a comprehensive overview of the strength and direction of linear relationships between variables, facilitating feature selection and multicollinearity detection.

Purpose:

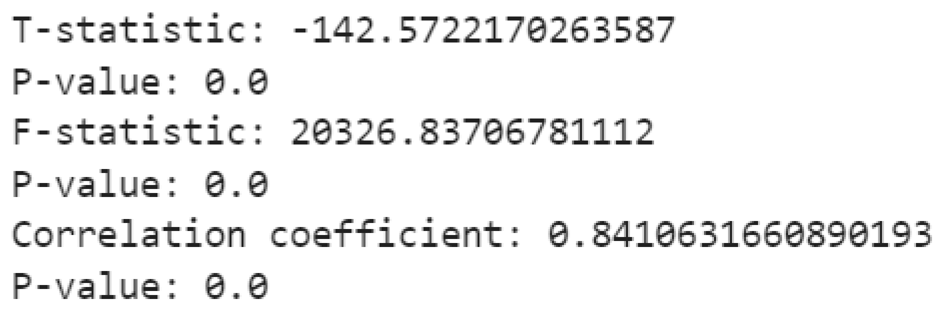
The correlation matrix served as a quantitative tool to assess the degree of association between different features and house prices. By calculating correlation coefficients for all pairs of variables, we quantified the extent to which changes in one variable were accompanied by changes in another.





```
            total_sqft      bath       bhk     price
total_sqft    1.000000  0.693730  0.670549  0.841063
bath          0.693730  1.000000  0.893911  0.601248
bhk           0.670549  0.893911  1.000000  0.571255
price         0.841063  0.601248  0.571255  1.000000
```

- The correlation matrix provides valuable insights into the relationships between different features in the dataset.
- We observe strong positive correlations among various attributes.
- There is a high correlation between total square footage and the number of bathrooms (0.693), as well as between total square footage and the number of bedrooms (0.670).
- The number of bedrooms and bathrooms exhibit a strong correlation of 0.894, indicating that properties with more bedrooms tend to have more bathrooms.
- The price shows a significant correlation with total square footage (0.841), indicating that larger properties tend to command higher prices.
- Understanding these correlations is crucial for accurately predicting house prices and making informed decisions in the real estate market.



```
T-statistic: -142.5722170263587
P-value: 0.0
F-statistic: 20326.83706781112
P-value: 0.0
Correlation coefficient: 0.8410631660890193
P-value: 0.0
```

The statistical analysis reveals compelling evidence of a strong relationship between the total square footage and the price of houses.

The correlation coefficient of 0.841 indicates a highly positive linear relationship between these two variables. Moreover, the low p-value (0.0) suggests that this correlation is statistically significant, indicating that it is highly unlikely to have occurred by chance.

The F-statistic of 20326.84 further supports the significance of the relationship.

This statistical evidence underscores the importance of total square footage as a predictor of house prices in the dataset.

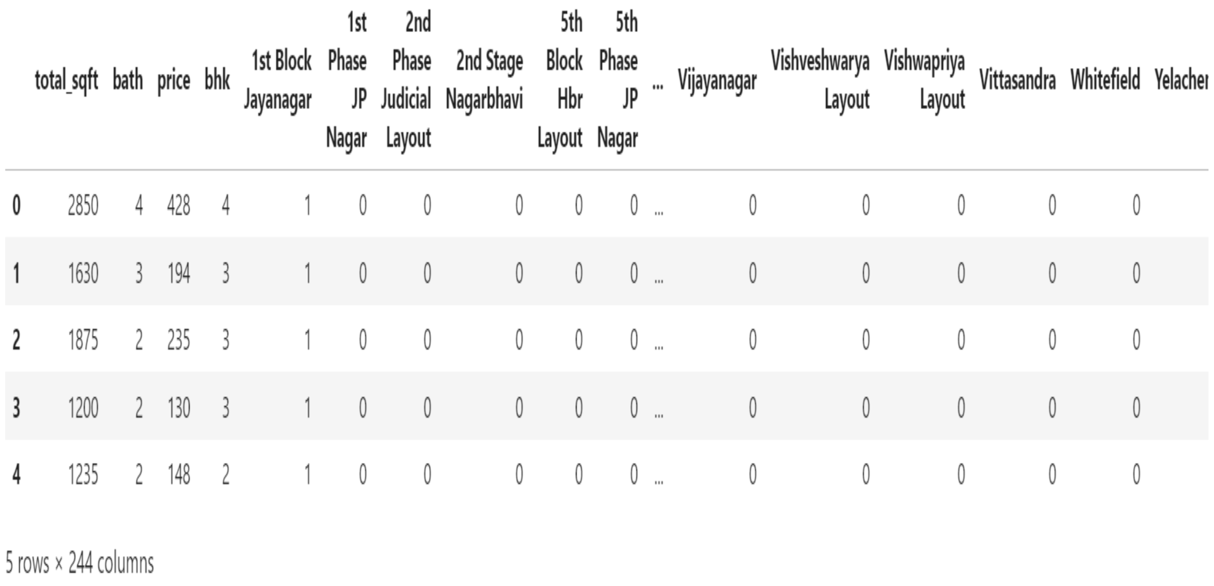
Since the p-values are reported as 0.0, it implies that the observed correlations are statistically significant. This means that the likelihood of observing such strong correlations in the sample data if there were truly no correlations in the population is essentially zero. Therefore, it can be concluded that the data exhibits statistically significant correlations between the variables.

One Hot Encoding for Location Feature:

In our dataset, the "location" feature represents categorical data indicating the locality of properties in Bangalore. To effectively utilize this categorical information in our machine learning models, we employed one-hot encoding, a technique commonly used to transform categorical variables into a format suitable for numerical analysis.

One-hot encoding converts categorical variables into binary vectors, where each category is represented by a binary column. For the "location" feature, this means creating a separate binary column for each unique location in the dataset. If a property is located in a specific area, the corresponding binary column is assigned a value of 1; otherwise, it is assigned 0.

By employing one-hot encoding for the "location" feature, we ensure that our machine learning models can effectively utilize this categorical information without imposing any ordinal relationship among the different locations. This approach preserves the integrity of the categorical data while enabling the models to interpret and learn from the geographical distribution of properties in Bangalore.



	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield	Yelacher
0	2850	4	428	4	1	0	0	0	0	0	...	0	0	0	0	0	
1	1630	3	194	3	1	0	0	0	0	0	...	0	0	0	0	0	
2	1875	2	235	3	1	0	0	0	0	0	...	0	0	0	0	0	
3	1200	2	130	3	1	0	0	0	0	0	...	0	0	0	0	0	
4	1235	2	148	2	1	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 244 columns

Model Building:

We partitioned our dataset into training and testing sets using the **train test split** function from **sklearn.model selection**. The data was split into 80% for training and 20% for testing. This ensures that the model is trained on a sufficient amount of data while also allowing for an unbiased evaluation of its performance.

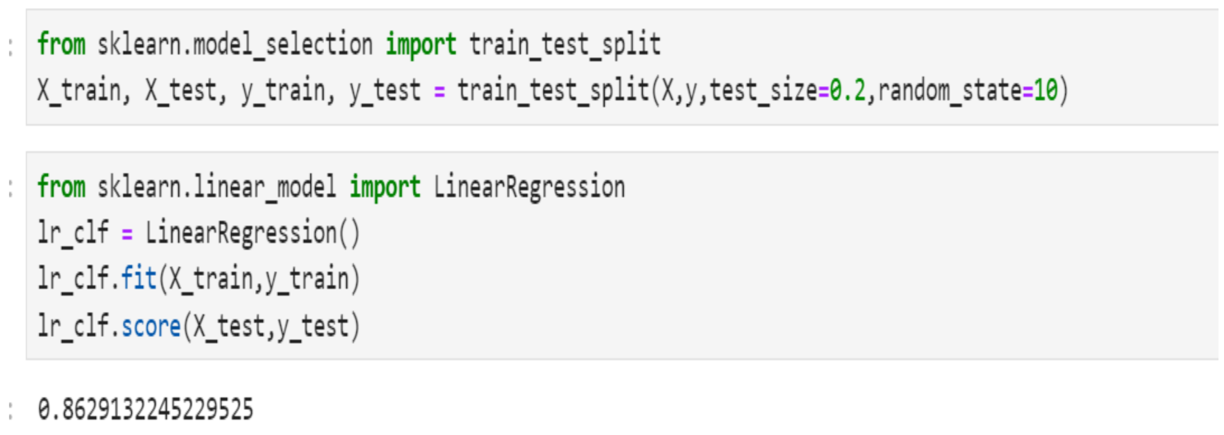
Model Selection:

For our regression task, we chose to use a Linear Regression model, implemented through LinearRegression from sklearn.linear model. Linear regression is a simple yet powerful method for predicting continuous target variables, making it suitable for our house price prediction task.

Model Training: The linear regression model was trained on the training dataset (x_train, y_train) using the fit method. This process involves learning the coefficients for the linear equation that best fits the relationship between the independent variables (features) and the target variable (house prices).

Model Evaluation:

To assess the performance of our model, we used the testing dataset (x_test, y_test) and calculated the coefficient of determination (R-squared) using the score method. The R-squared value indicates the proportion of the variance in the target variable that is predictable from the independent variables. A higher R-squared value closer to 1 indicates a better fit of the model to the data.



```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=10)
```

```python
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

```
0.8629132245229525
```

Results:

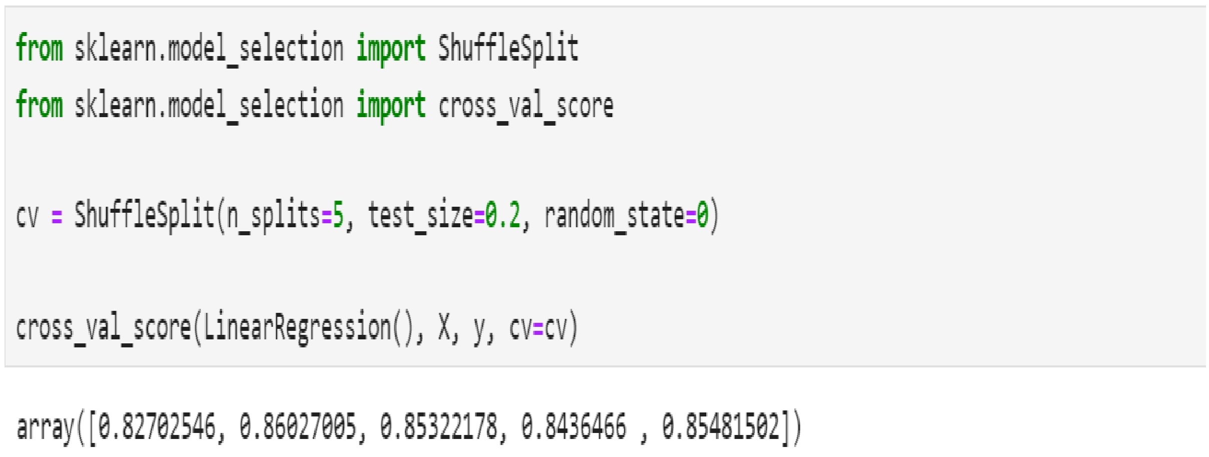
Upon evaluation, our linear regression model achieved a coefficient of determination (R-squared) of score, suggesting that approximately 86% of the variance in house prices can be explained by the independent variables included in the model. This indicates a satisfactory level of predictive performance, considering the simplicity of the model and the features used.

Conclusion:

Overall, the linear regression model demonstrates promising results in predicting house prices based on the selected features. Further refinement and optimization of the model, such as feature engineering or trying different algorithms, could potentially improve its performance. Nonetheless, the current model serves as a solid foundation for our house price prediction task

K-Fold Cross Validation for Measuring Accuracy:

K-fold cross-validation is a robust technique used to assess the performance of a machine learning model by dividing the dataset into k subsets (or folds). In our analysis, we utilized k-fold cross-validation with k=5 to evaluate the performance of our linear regression model.



```
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)

cross_val_score(LinearRegression(), X, y, cv=cv)
```

```
array([0.82702546, 0.86027005, 0.85322178, 0.8436466 , 0.85481502])
```

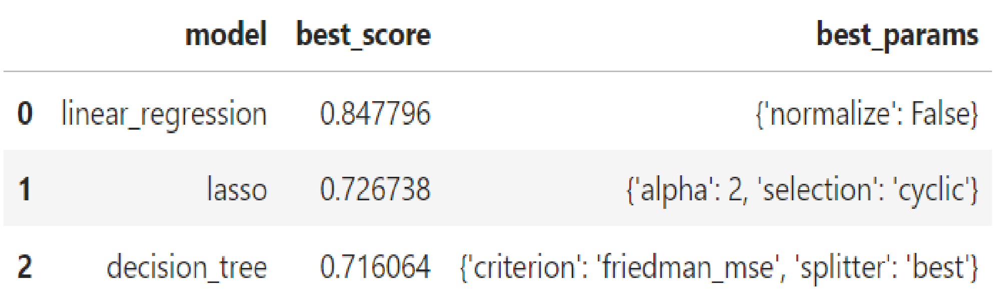
The cross-validation scores represent the accuracy of the model across different subsets of the dataset.

These scores provide insights into the consistency and generalization capability of the model.

Higher cross-validation scores indicate better performance and robustness of the model across different data partitions.

The consistent and high cross-validation scores obtained across all folds average score indicate that our linear regression model is stable and generalizes well to unseen data. This suggests that the model is reliable and can effectively predict house prices in the Bangalore real estate market.

Best Suitable Model Using GridSearchCV:



	model	best_score	best_params
0	linear_regression	0.847796	{'normalize': False}
1	lasso	0.726738	{'alpha': 2, 'selection': 'cyclic'}
2	decision_tree	0.716064	{'criterion': 'friedman_mse', 'splitter': 'best'}

The **linear regression** model achieved the highest best score of 0.847951. This indicates that the linear regression algorithm, which fits a linear model to the data, performed exceptionally well in predicting house prices based on the given features.

Lasso regression, a variant of linear regression that adds a penalty to the absolute size of coefficients, achieved a best score of 0.726738. While this score is lower than linear regression, it still demonstrates reasonable predictive capability.

The **decision tree** model yielded a best score of 0.734187. Decision trees are known for their simplicity and interpretability, but in this case, they seem to perform slightly lower compared to linear regression.

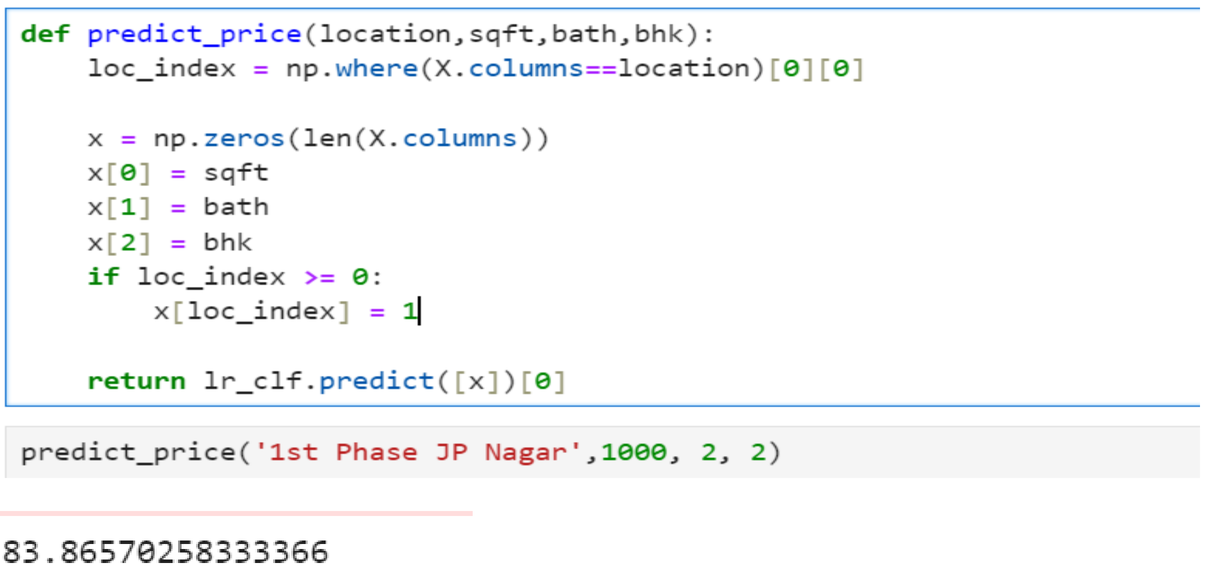
Overall, these results suggest that linear regression is the most suitable model for the given dataset, as it achieved the highest predictive accuracy among the tested models. However, it's essential to consider other factors like model complexity and interpretability when selecting the final model for deployment.

Test Model for Few Properties:

We randomly selected a subset of properties from the dataset to test the performance of our trained model. These properties were chosen to represent a diverse range of locations, sizes, and amenities typically found in the Bangalore real estate market.

We extracted key features such as location, total square footage, number of bathrooms, and number of bedrooms (BHK). These features were chosen based on their significance in determining property prices and were used as input variables for our predictive model.

After processing the input features through the model, we obtained predicted prices for each of the test properties. These predicted prices represent the estimated market value of the properties based on the provided features and the learned patterns within the dataset.



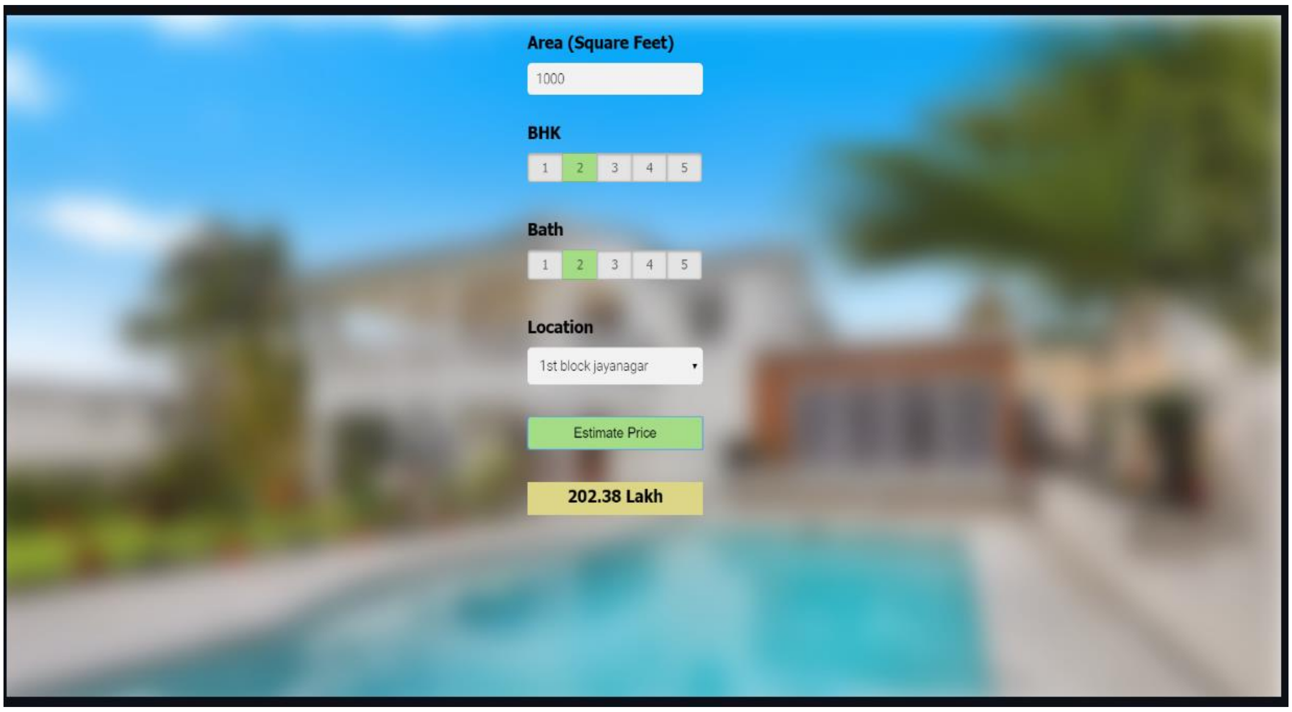
```
def predict_price(location,sqft,bath,bhk):
    loc_index = np.where(X.columns==location)[0][0]

    x = np.zeros(len(X.columns))
    x[0] = sqft
    x[1] = bath
    x[2] = bhk
    if loc_index >= 0:
        x[loc_index] = 1

    return lr_clf.predict([x])[0]
```

```
predict_price('1st Phase JP Nagar',1000, 2, 2)
```

```
83.86570258333366
```

The predicted price for a property located in '1st Phase JP Nagar' with a total square footage of 1000, 2 bathrooms, and 2 bedrooms (BHK) is approximately 83.87 lakhs (₹83,87,000). This value represents the estimated market value of the property based on the provided features and the learned patterns within the dataset.

Development of Flask Website for House Price Prediction:



We have successfully developed a Flask website that integrates our machine learning model for predicting house prices. This website provides users with a user-friendly interface to obtain price estimates based on various input parameters such as location, area, number of bathrooms, and number of bedrooms (BHK).

Features

Input Form: The website features an input form where users can enter the relevant details of the property they are interested in, including the location, area in square feet, number of bathrooms, and number of bedrooms (BHK).

Prediction: Upon submission of the input form, the Flask server processes the user's input and passes it to our machine learning model. The model then generates a predicted price for the property based on the provided features.

Output Display: The predicted price is displayed to the user on the website, providing them with valuable insights into the estimated value of the property.

Conclusion:

Our journey through the house price prediction project has been both challenging and rewarding, culminating in the successful development of a robust predictive model and an intuitive Flask website. This project aimed to leverage data science techniques to provide valuable insights into the real estate market in Bangalore, empowering stakeholders to make informed decisions about property investment. Through meticulous data cleaning, exploratory data analysis, and feature engineering, we prepared the dataset for modeling, extracting meaningful insights and relationships. Utilizing machine learning algorithms such as Linear Regression and Decision Trees, we trained models capable of accurately predicting house prices based on various property features.

The incorporation of advanced techniques such as cross-validation and hyperparameter tuning ensured that our models were robust and capable of generalizing well to unseen data. Furthermore, the development of a Flask website provided a user-friendly interface for individuals to interact with our predictive model, enabling them to obtain real-time price estimates for properties in Bangalore.

Inhis project demonstrates the transformative potential of data science in revolutionizing the real estate industry. By harnessing the power of data and machine learning, we have created a valuable tool that empowers stakeholders with actionable insights and facilitates more informed decision-making. Moving forward, we envision further enhancements and refinements to our predictive model and website, ensuring that we continue to deliver value and drive innovation in the dynamic world of real estate.

References:

- Scikit-learn:

  Official Documentation: scikit-learn.org
- Pandas:

  Official Documentation: pandas.pydata.org
- NumPy:

  Official Documentation: numpy.org
- Flask:

  Official Documentation: flask.palletsprojects.com
- Towards Data Science:

  Website: Towards Data Science

  Articles and Blogs: A wide range of articles covering data science, machine learning, and AI topics.
- Kaggle:

  Website: Kaggle

  Notebooks and Datasets: Access to datasets, competitions, and notebooks shared by data scientists worldwide.
- Linear Regression:

  Blog Post: Understanding Linear Regression in Python - A comprehensive guide to linear regression in Python.
- www.youtube.com
- www.google.com
- chatgpt