MACHINE LEARNING

Using

Regressions

Real Estate and Housing Price Prediction

Winter Internship Report Submitted in partial fulfilment

of the requirement for undergraduate degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

By

Akash Varma Saripella

HU21CSEN0100672

Under the Guidance of

Dr.G.Soma Sekhar

Associate Professor

Computer Science and Engineering, GST , HYD

GITAM (Deemed to be University)

Hyderabad

December 2023

DECLARATION

I submit this industrial training work entitled “Real Estate and Housing Price Prediction” to GITAM (Deemed To Be University), Hyderabad in partial fulfilment of the requirements for the award of the degree of “Bachelor of Technology” in “Computer Science and Engineering”. I declare that it was carried out independently by me under the guidance of **Dr.G.Soma Sekhar**, Associate Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Regards,

Akash Varma Saripella,

HU21CSEN0100672.

Bengaluru House Price Prediction

Abstract:

The "Bengaluru House Price Prediction" project is a comprehensive exploration and analysis of real estate data in Bengaluru, India, with the goal of predicting house prices using machine learning techniques. The project encompasses various stages of the data science lifecycle, including data acquisition, cleaning, exploration, feature engineering, and model development.

The dataset used in the project contains information about different houses, including their locations, sizes, amenities, and prices. The initial steps involve data cleaning and pre-processing to handle missing values, eliminate irrelevant features, and convert data into a suitable format. Exploratory data analysis is then performed to gain insights into the distribution of house prices, the relationship between features, and the identification of potential outliers. To enhance the predictive capabilities of the model, feature engineering techniques are employed, such as converting total square footage to numeric values and calculating the price per square foot. The dataset's location dimensionality is reduced for efficiency by grouping locations with a small number of houses into a category labelled 'other.'

The machine learning component involves the training of a linear regression model to predict house prices based on relevant features. The model is evaluated using cross-validation to ensure robust performance and further refined through hyperparameter tuning using GridSearchCV.

The project also includes data visualization to illustrate the relationship between house prices and specific features in selected locations. Outliers in price per square foot are addressed to enhance the model's accuracy.

In conclusion, the "Bengaluru House Price Prediction" project serves as a valuable resource for real estate stakeholders and potential homebuyers by providing insights into factors influencing house prices. The machine learning model, trained on historical data, offers a tool for making informed predictions about future housing prices in Bengaluru.

Table Of Contents

Chapter 1: Data Science

Definition:
Data Science is an interdisciplinary field that harnesses scientific methods, processes, algorithms, and systems to extract meaningful patterns, insights, and knowledge from structured and unstructured data. It involves a fusion of expertise from statistics, mathematics, computer science, and domain-specific knowledge to transform raw data into valuable information, enabling data-driven decision-making.

Key Components:

Data Collection:

Definition: Gathering data from diverse sources, such as databases, APIs, sensors, and more.
Importance: The foundation of data science lies in collecting relevant and comprehensive data to derive meaningful insights.

Data Cleaning and Pre-processing:

Definition: Handling missing values, removing outliers, and transforming data into a suitable format for analysis.
Importance: Ensuring data quality and preparing it for analysis are critical for accurate and reliable results.
Exploratory Data Analysis (EDA):

Definition: Exploring and visualizing data to identify patterns, trends, and relationships.
Importance: EDA aids in understanding the structure of the data and guides further analysis.

Feature Engineering:

Definition: Creating new features from existing data to improve model performance.
Importance: Crafting relevant features enhances the predictive power of machine learning models.

Model Building:

Definition: Developing predictive models using machine learning algorithms.
Importance: The core of data science involves selecting and implementing models that can uncover patterns and make predictions.

Model Evaluation and Interpretation:

Definition: Assessing model performance and interpreting results to derive actionable insights.
Importance: Effective evaluation ensures the reliability of models, and interpretation provides actionable intelligence.

Deployment:

Definition: Implementing models into production environments for real-world applications.
Importance: Deploying models transforms insights into practical solutions, impacting decision-making processes.

Chapter 2: Machine Learning

Introduction to Python:

• Python is a high-level, interpreted, interactive and object-oriented scripting language.
• Python is a general-purpose programming language that is often applied in scripting
roles
• Python is Interpreted: Python is processed at runtime by the interpreter. You do not
need to compile your program before executing it. This is like PERL and PHP.
• Python is Interactive: You can sit at a Python prompt and interact with the interpreter
directly to write your programs.
• Python is Object-Oriented: Python supports the Object-Oriented style or technique of
programming that encapsulates code within objects.

Machine Learning:

Definition:

Machine Learning (ML) is a branch of artificial intelligence (AI) that focuses on the development of algorithms and models allowing computers to learn from data. ML systems aim to make predictions or decisions without being explicitly programmed, relying on patterns
and inferences derived from the data.

Types of Machine Learning

Supervised Learning:

Definition:

Supervised Learning is a type of machine learning where the model is trained on a labeled dataset, meaning that each input data point is paired with a corresponding output label. The goal is for the model to learn the mapping from inputs to outputs, enabling it to make predictions or classifications on new, unseen data.

Example:
Classification:

Definition: Assigning labels to input data points based on their characteristics.
Example: Email spam detection, where emails are classified as either spam or not spam based on features like content, sender, etc.

Regression:

Definition: Predicting numerical values for given input data.
Example: Predicting house prices based on features such as square footage, number of bedrooms, and location.

Unsupervised Learning:

Definition:
Unsupervised Learning involves training a model on unlabeled data, and the model must discover patterns, structures, or relationships within the data without explicit guidance.

Example:

Clustering:

Definition: Grouping similar data points together based on inherent patterns or similarities.
Example: Customer segmentation in marketing, where customers are grouped based on purchasing behavior.

Dimensionality Reduction:

Definition: Simplifying data while retaining its essential features.
Example: Principal Component Analysis (PCA) to reduce the number of features in a dataset while preserving its variance.

Reinforcement Learning:

Definition:
Reinforcement Learning involves training a model to make sequences of decisions by interacting with an environment. The model receives feedback in the form of rewards or penalties, allowing it to learn optimal behavior over time.

Example:

Autonomous Systems:

Definition: Training algorithms for systems that can operate autonomously in dynamic environments.
Example: Autonomous vehicles learning to navigate through traffic.

Game-Playing Agents:

Definition: Training models to play games by making decisions to maximize rewards.
Example: Training a computer program to play chess or video games.

Semi-Supervised Learning:

Definition:
Semi-Supervised Learning is a hybrid approach that combines elements of both supervised and unsupervised learning. The model is trained on a dataset containing both labeled and unlabeled data.

Example:

Utilizing a Limited Labeled Dataset:
Definition: Training a model with a small subset of labeled data and a larger amount of unlabeled data.
Example: Training a speech recognition model with a limited dataset of transcribed speech and a vast amount of unlabeled audio data.

Transfer Learning:

Definition:
Transfer Learning involves leveraging pre-trained knowledge from a model on one task to improve its performance on a different but related task. The idea is to transfer the learned features or knowledge to a new domain.

Example:

Image Recognition Task:
Definition: Training a model on a vast dataset like ImageNet to learn general features from images.
Example: Using the pre-trained features for image recognition in a specific domain, such as identifying plant species in agriculture.

Understanding these different types of machine learning is crucial for selecting the appropriate approach based on the nature of the data and the goals of a particular task or project.

Chapter 3: Model Used

The type of machine learning model used in our code is a Supervised Learning Regression Model, and more specifically, it's a **Linear Regression Model**.

Here's why:

Supervised Learning:

The model is trained on a labelled dataset, where I have input features (independent variables) and corresponding output labels (dependent variable or target). In my case, the features include location, square footage, number of bathrooms, and bedrooms, and the target is the house price.

Regression:

The goal of the model is to predict a continuous output variable (house price) based on the input features. In contrast to classification, where the goal is to predict a categorical label, regression deals with predicting a quantity.

Linear Regression:

Specifically, my model is a Linear Regression model. Linear Regression assumes a linear relationship between the input features and the output variable. It models this relationship using a linear equation, where the predicted output is a linear combination of the input features.

In summary, our machine learning model is a Supervised Learning Regression model, and the algorithm chosen for this regression task is Linear Regression.

Regression And Types: -

Regression is a type of supervised learning where the goal is to predict a continuous output variable (dependent variable) based on one or more input features (independent variables). The relationship between the input features and the output variable is modelled using a mathematical function.

Types of Regression:

Linear Regression:

Definition: Assumes a linear relationship between the input features and the output variable.
Use Case: Predicting house prices based on square footage.

Multiple Linear Regression:

Definition: Extends linear regression to multiple input features.
Use Case: Predicting sales based on advertising spending and other factors.

Polynomial Regression:

Definition: Includes polynomial terms (e.g., squared or cubed features) to capture non-linear relationships.
Use Case: Modelling a curved relationship between variables.

Ridge Regression and Lasso Regression:

Definition: Introduces regularization terms to prevent overfitting.
Use Case: Handling multicollinearity in multiple linear regression.

Logistic Regression:

Definition: Despite its name, logistic regression is used for binary classification problems.
Use Case: Predicting whether an email is spam or not.

Support Vector Regression (SVR):

Definition: Uses support vector machines for regression tasks.
Use Case: Predicting stock prices.

Understanding these concepts is fundamental for anyone involved in data science and machine learning, as they form the basis for developing predictive models and deriving insights from data.

Chapter 4: Data modification

Introduction:

The CSV file containing Bangalore house prices with 13,000 entries and 9 columns representing basic requirements of people in the dataset with information about various houses in Bangalore.

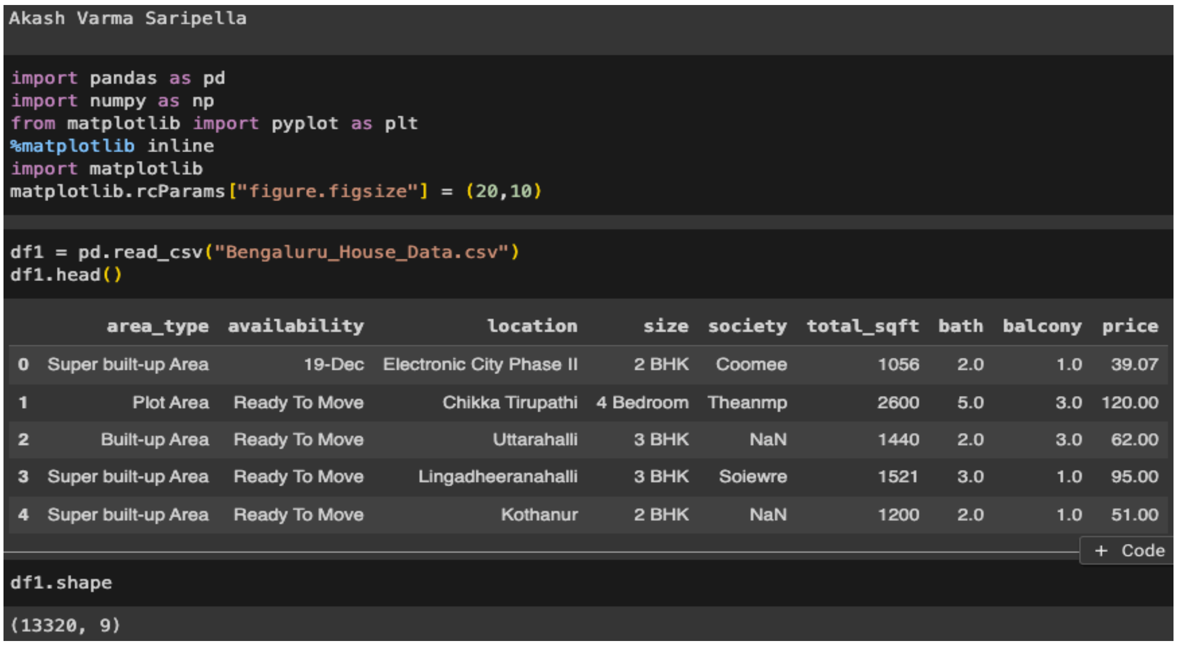
Data Loading and Exploration

Pandas and NumPy:

Used for data manipulation, analysis, and numerical operations.

Loading Data:

Loaded the Bengaluru house data from a CSV file using Pandas.
Checked the shape and columns of the dataset.



Akash Varma Saripella

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
```

```
df1 = pd.read_csv("Bengaluru_House_Data.csv")
df1.head()
```

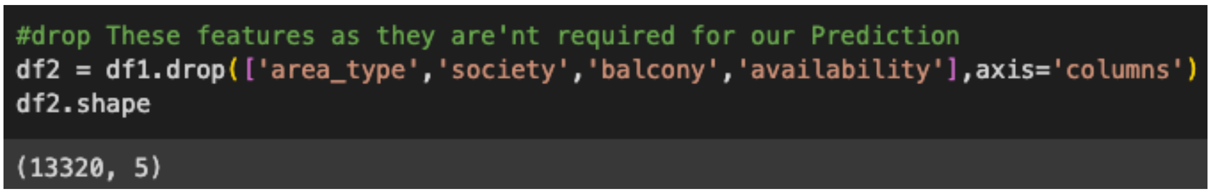
	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	51.00

```
df1.shape
```

```
(13320, 9)
```

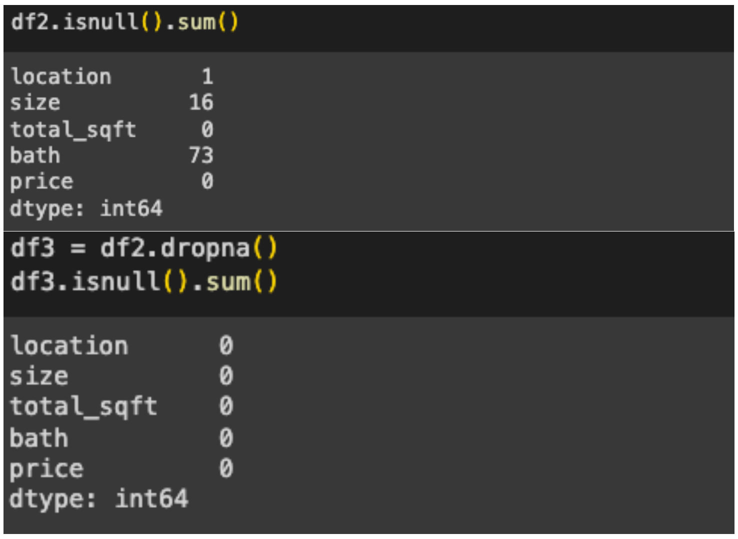
Data Cleaning:

- Explored unique values and value counts in the 'area_type' column.
- Dropped unnecessary features ('area_type', 'society', 'balcony', 'availability').
- Checked for null values and handled them by dropping rows with null values.



```
#drop These features as they are'nt required for our Prediction
df2 = df1.drop(['area_type','society','balcony','availability'],axis='columns')
df2.shape
```

```
(13320, 5)
```



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

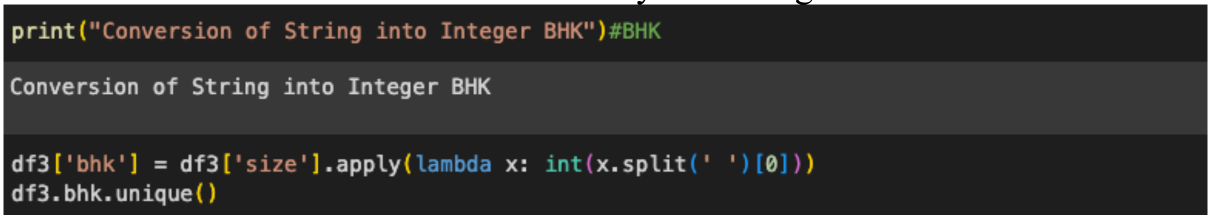
```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

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

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

Feature Engineering:

- Converted the 'size' column to 'bhk' by extracting the number of bedrooms.

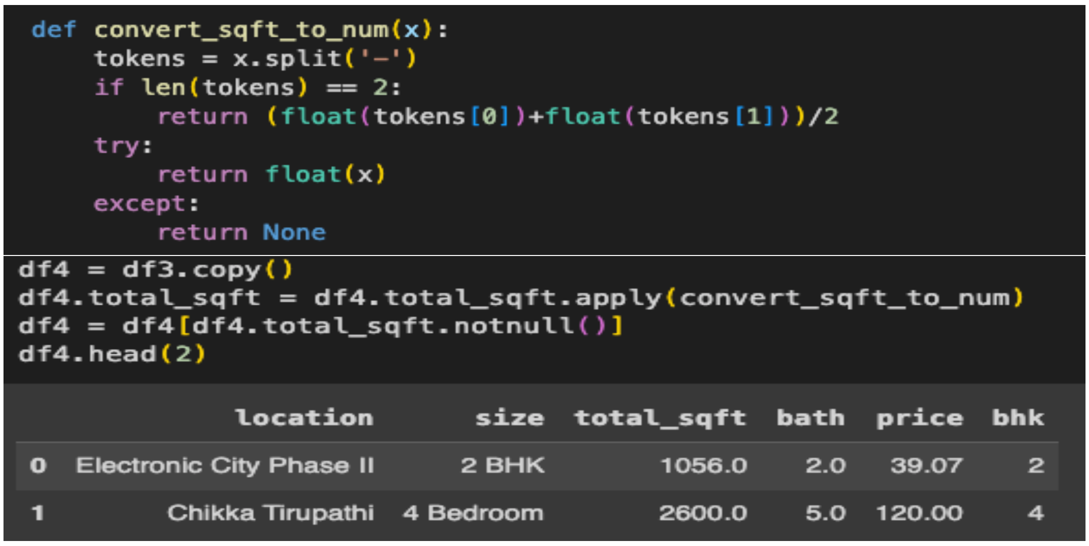


```
print("Conversion of String into Integer BHK")#BHK
```

```
Conversion of String into Integer BHK
```

```
df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
df3.bhk.unique()
```

- Handled non-numeric values in 'total_sqft' by converting them to numeric values.

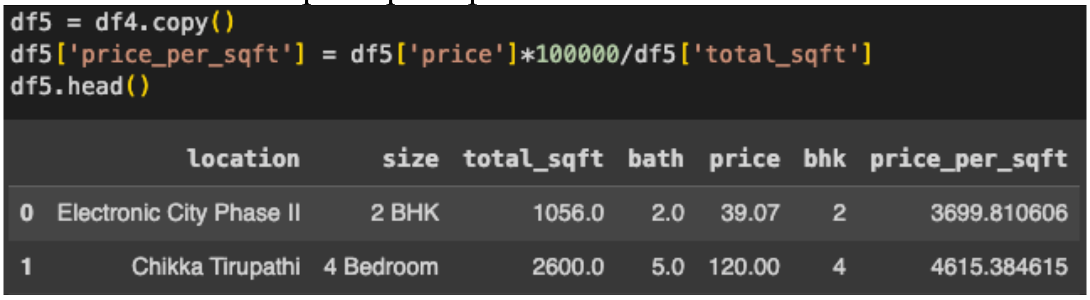


```
def convert_sqft_to_num(x):
    tokens = x.split('-')
    if len(tokens) == 2:
        return (float(tokens[0])+float(tokens[1]))/2
    try:
        return float(x)
    except:
        return None
```

```
df4 = df3.copy()
df4.total_sqft = df4.total_sqft.apply(convert_sqft_to_num)
df4 = df4[df4.total_sqft.notnull()]
df4.head(2)
```

	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4

- Calculated the price per square foot.



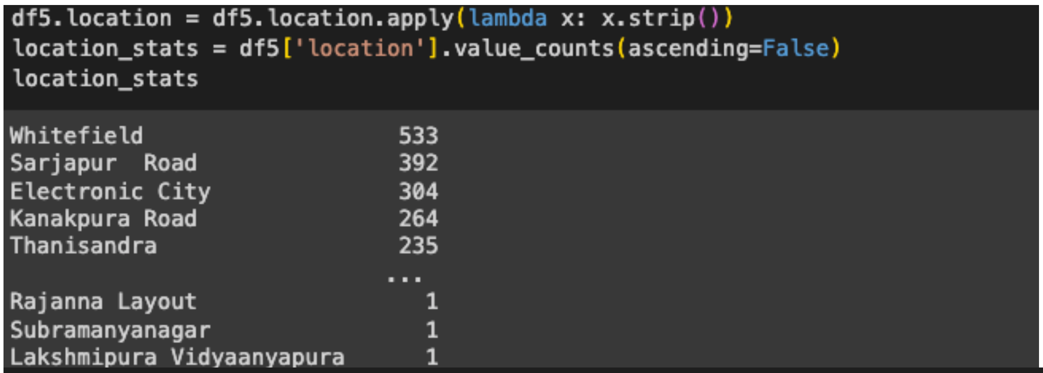
```
df5 = df4.copy()
df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
df5.head()
```

	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

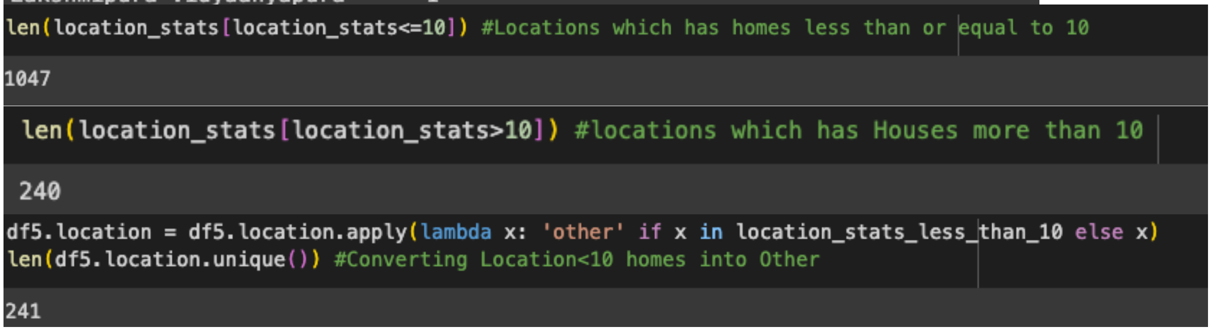
Chapter 5: Analysis and Visualisation

Dimensionality Reduction:

Reduced the number of local visualize efficiency by grouping those with fewer than 10 homes into 'other'.



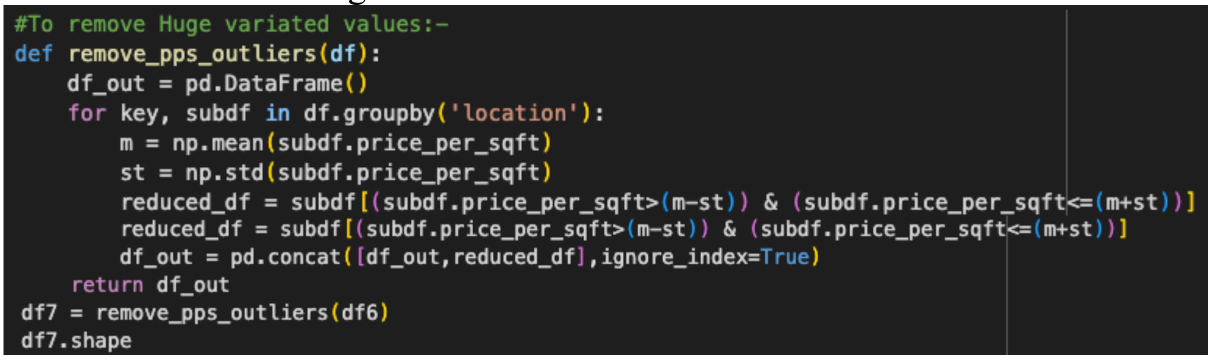
```
df5.location = df5.location.apply(lambda x: x.strip())
location_stats = df5['location'].value_counts(ascending=False)
location_stats

Whitefield                    533
Sarjapur  Road                392
Electronic City               304
Kanakpura Road                264
Thanisandra                   235
                             ...
Rajanna Layout                  1
Subramanyanagar                 1
Lakshmipura Vidyaanyapura       1
```



```
len(location_stats[location_stats<=10]) #Locations which has homes less than or equal to 10

1047

len(location_stats[location_stats>10]) #locations which has Houses more than 10

240

df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
len(df5.location.unique()) #Converting Location<10 homes into Other

241
```

Outlier Removal:

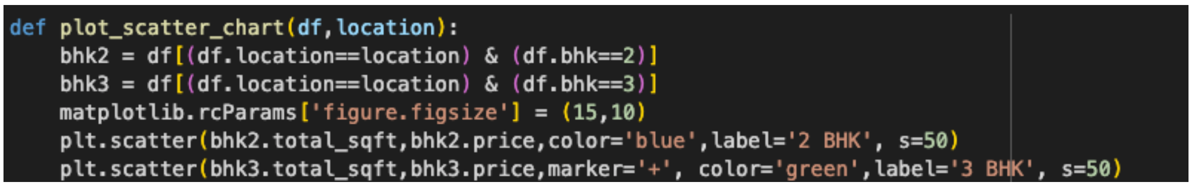
Removed outliers in the dataset based on the price per square foot for each location. Which has huge difference.



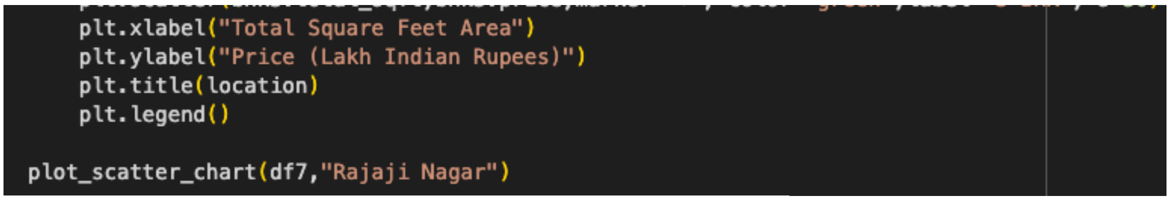
```
#To remove Huge variated values:-
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key, subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)
        st = np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m+st))]
        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
```

Data Visualization:

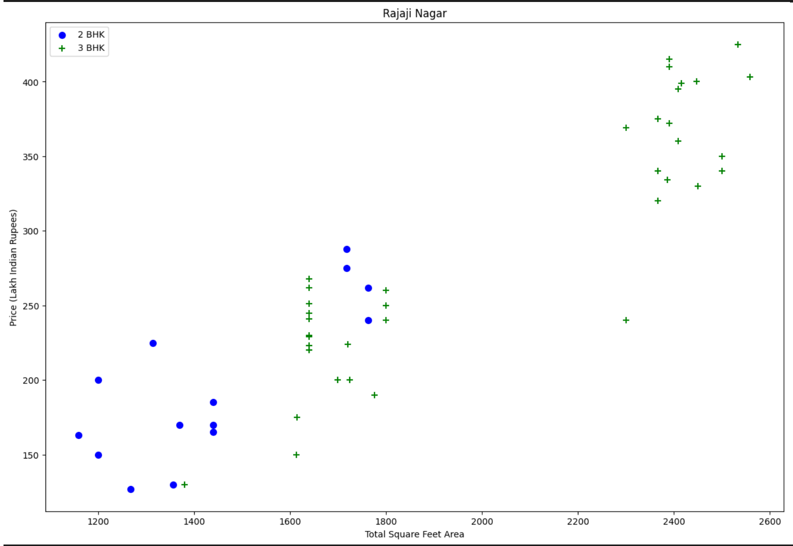
Plotted scatter charts to visualize the distribution of prices based on total square feet for different locations.

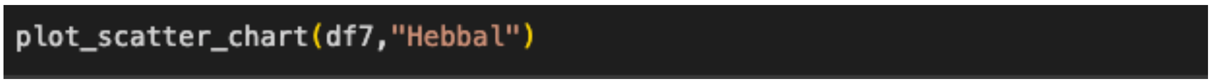


```
def plot_scatter_chart(df,location):
    bhk2 = df[(df.location==location) & (df.bhk==2)]
    bhk3 = df[(df.location==location) & (df.bhk==3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s=50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker='+', color='green',label='3 BHK', s=50)
```

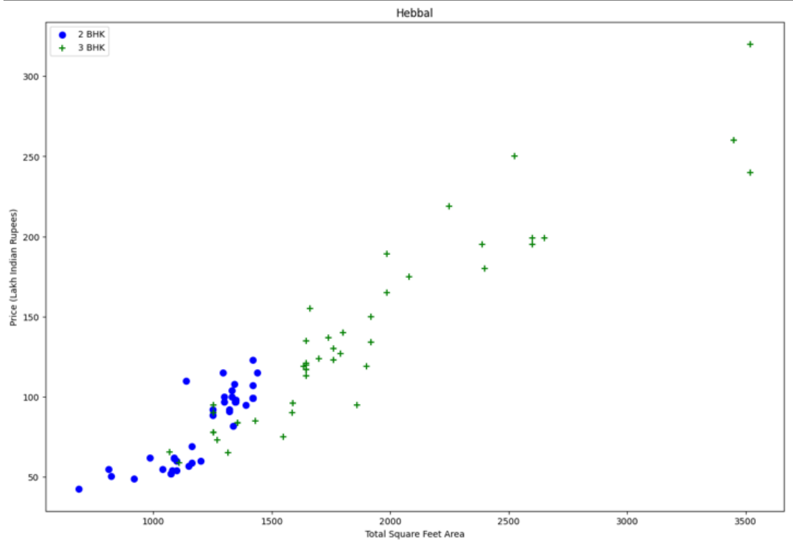


```
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price (Lakh Indian Rupees)")
    plt.title(location)
    plt.legend()

plot_scatter_chart(df7,"Rajaji Nagar")
```

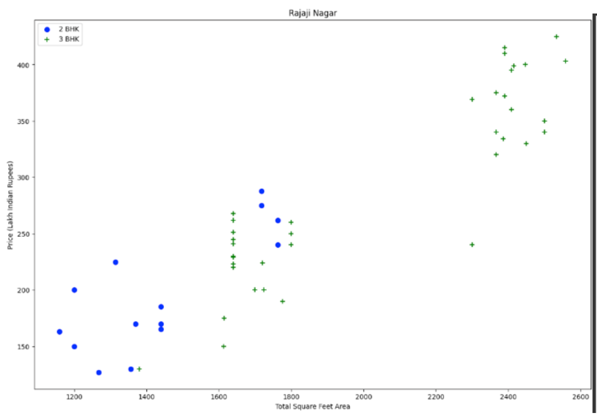


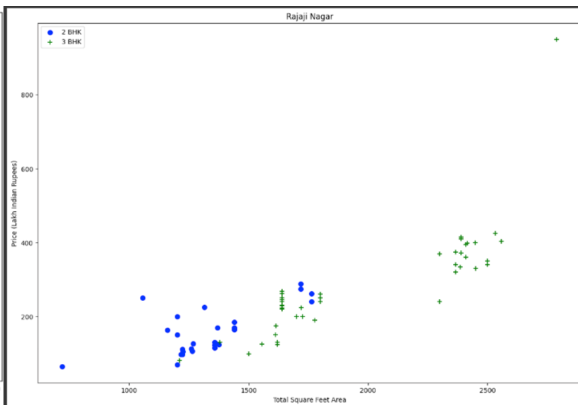


```
plot_scatter_chart(df7,"Hebbal")
```



With And without Outlier: -



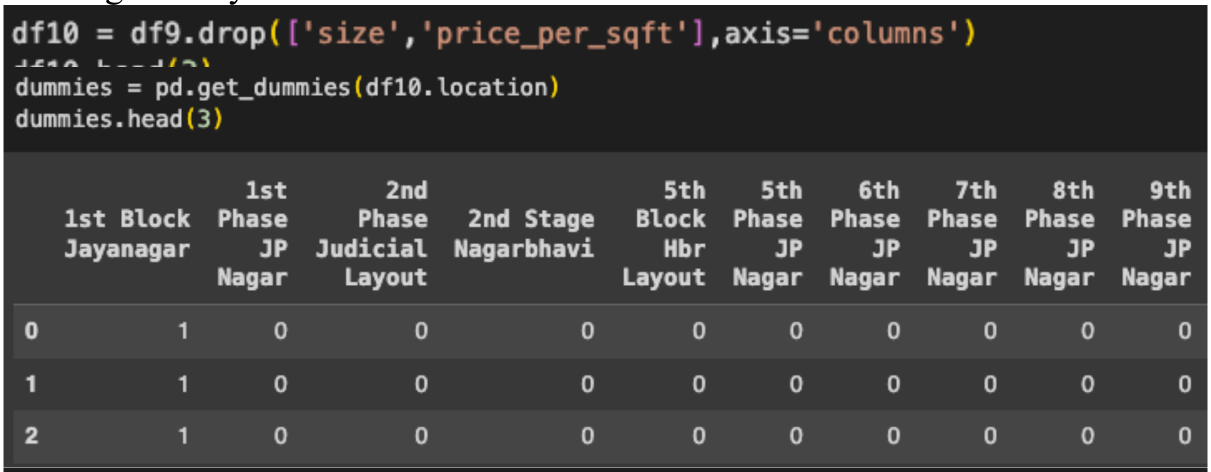


Chapter 6: Model Building

Model Building

Data Preparation:

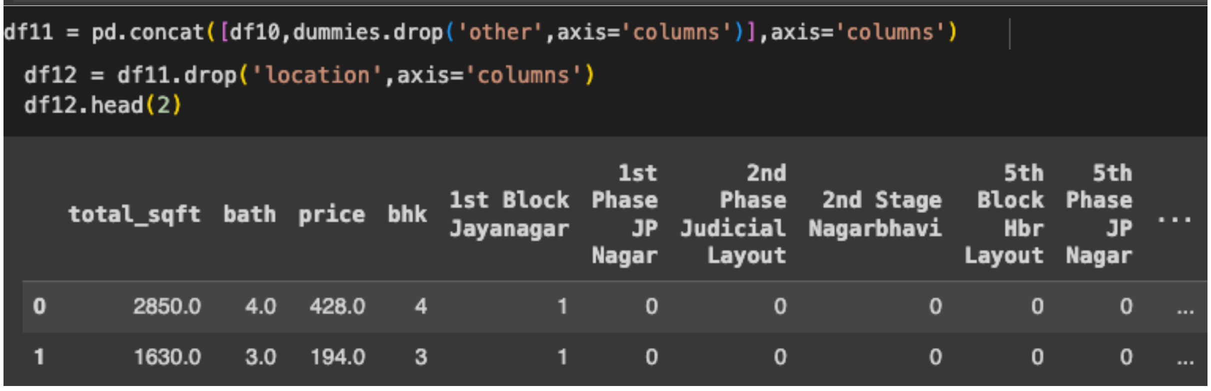
Prepared the data for building the model by dropping irrelevant columns and creating dummy variables for the 'location' column.



```
df10 = df9.drop(['size','price_per_sqft'],axis='columns')
```

```
dummies = pd.get_dummies(df10.location)
dummies.head(3)
```

	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar	9th Phase JP Nagar
0	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0



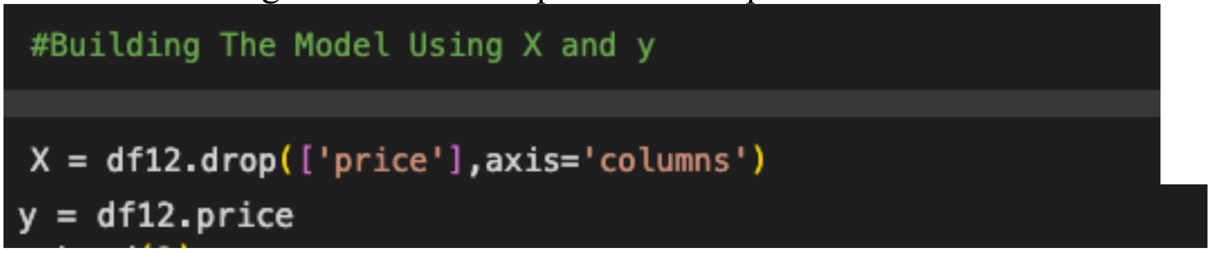
```
df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
```

```
df12 = df11.drop('location',axis='columns')
df12.head(2)
```

	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	...
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...

Linear Regression Model:

Built a Linear Regression model to predict house prices.

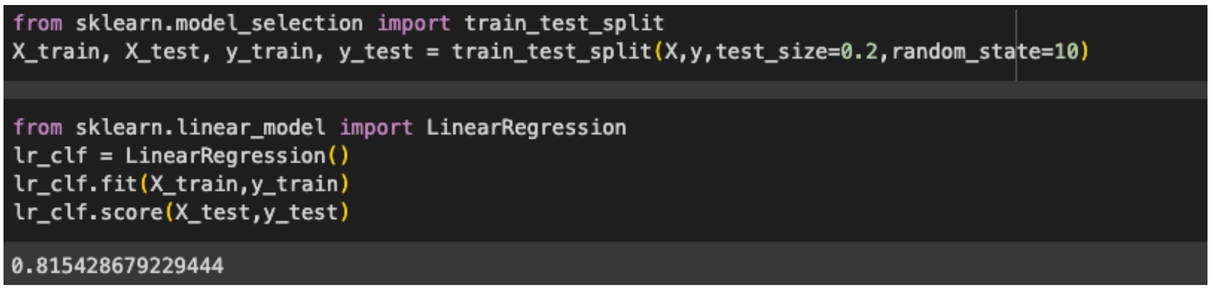


```
#Building The Model Using X and y
```

```
X = df12.drop(['price'],axis='columns')
y = df12.price
```

Model Evaluation:

Split the data into training and testing sets and evaluated the model's performance. The code snippet you provided is using the train_test_split function from the sklearn.model_selection module. This function is commonly used in machine learning to split a dataset into training and testing sets. It also shows accuracy for the code given.

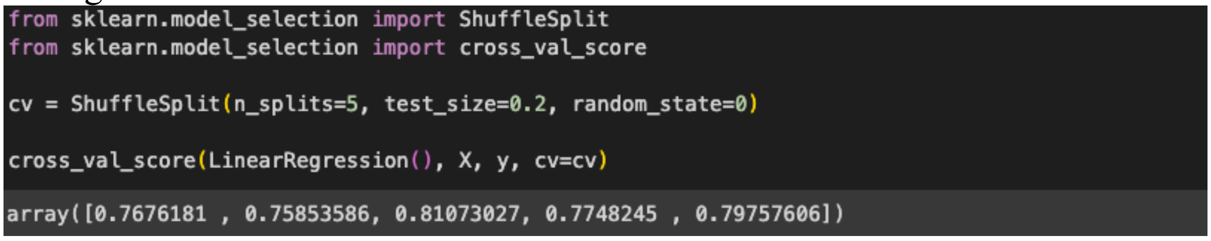


```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=10)
```

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

```
0.815428679229444
```

K Fold Cross-Validation:
K Fold Cross-Validation is a technique used to assess the performance and generalization of a machine learning model. It involves splitting the dataset into K folds (subsets), training the model K times, each time using K-1 folds for training and the remaining one for testing. The performance metric is then averaged over all K iterations.

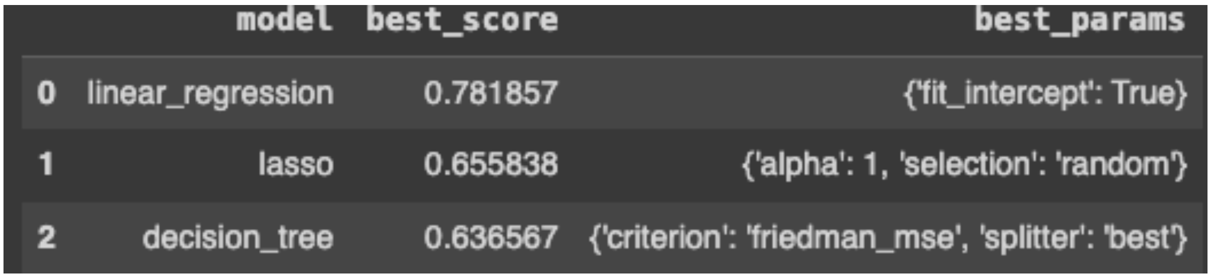


```
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

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

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

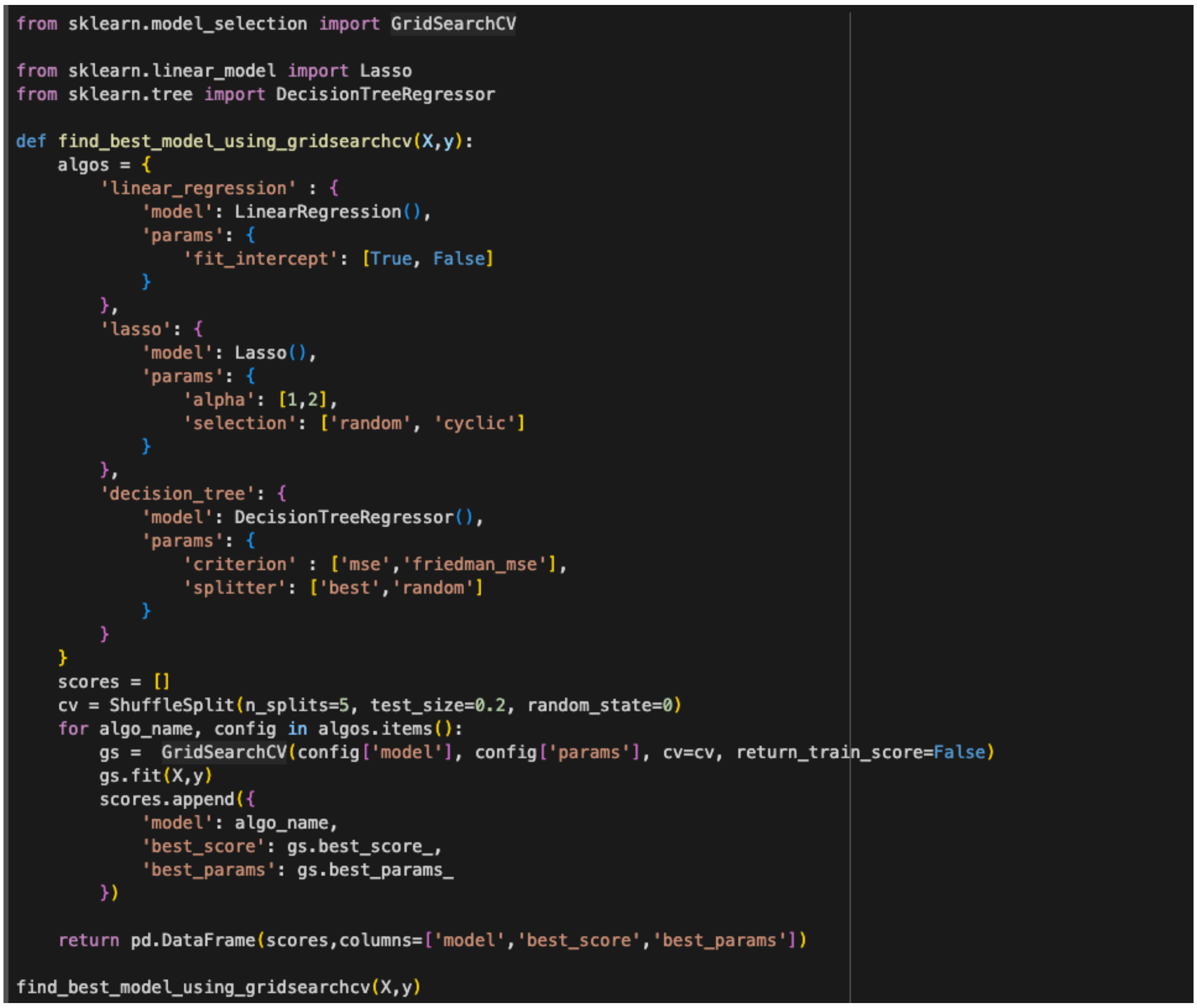
```
array([0.7676181 , 0.75853586, 0.81073027, 0.7748245 , 0.79757606])
```

Finding Best Model

GridSearchCV:
GridSearchCV is a technique used for hyperparameter tuning in machine learning models. It systematically searches for the best hyperparameters for a given model by evaluating the model's performance across a range of hyperparameter values.
Utilized GridSearchCV to find the best hyperparameters for different regression models (Linear Regression, Lasso, Decision Tree).
As Linear Regression has the highest accuracy, we choose it as our model.

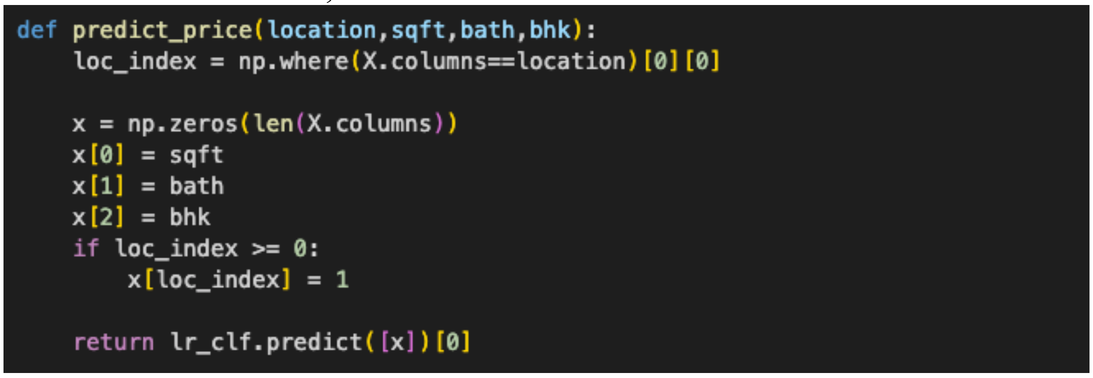


	model	best_score	best_params
0	linear_regression	0.781857	{'fit_intercept': True}
1	lasso	0.655838	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.636567	{'criterion': 'friedman_mse', 'splitter': 'best'}



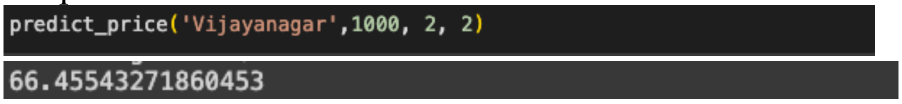
```python
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor

def find_best_model_using_gridsearchcv(X,y):
    algos = {
        'linear_regression' : {
            'model': LinearRegression(),
            'params': {
                'fit_intercept': [True, False]
            }
        },
        'lasso': {
            'model': Lasso(),
            'params': {
                'alpha': [1,2],
                'selection': ['random', 'cyclic']
            }
        },
        'decision_tree': {
            'model': DecisionTreeRegressor(),
            'params': {
                'criterion' : ['mse','friedman_mse'],
                'splitter': ['best','random']
            }
        }
    }
    scores = []
    cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
    for algo_name, config in algos.items():
        gs =  GridSearchCV(config['model'], config['params'], cv=cv, return_train_score=False)
        gs.fit(X,y)
        scores.append({
            'model': algo_name,
            'best_score': gs.best_score_,
            'best_params': gs.best_params_
        })

    return pd.DataFrame(scores,columns=['model','best_score','best_params'])

find_best_model_using_gridsearchcv(X,y)
```

Function for Predictions:
Created a function to predict house prices based on location, square footage, number of bathrooms, and bedrooms.



```python
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]
```

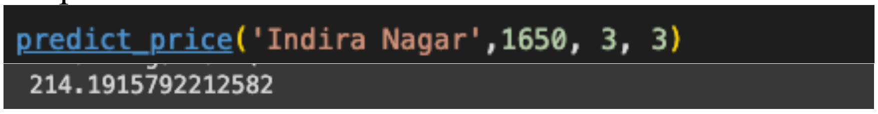
Output 1: -



```
predict_price('Vijayanagar',1000, 2, 2)
```

```
66.45543271860453
```

Output 2: -



```
predict_price('Indira Nagar',1650, 3, 3)
```

```
 214.1915792212582
```

Conclusion: -

The house price prediction project involved the analysis of Bangalore house data, aiming to build a model that predicts house prices based on various features. The project followed a structured approach, encompassing data loading, exploration, cleaning, visualization, and model building using machine learning techniques.