

ALIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY, ALIGARH



(APPROVED BY A.I.C.T.E & AFFILIATED TO AKTU, LUCKNOW)
2020 – 2024

**A PROJECT REPORT ON
HOUSE PRICE PREDICTION**

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
AWARD OF THE DEGREE OF

**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING**



Under the Guidance of:

Mr. Rohit Yadav

SUBMITTED BY

Arpit Kumar Nauhwara

(2001090100006)

SUBMITTED TO

DR. ANAND SHARMA

HOD (CSE)

MR. ROHIT YADAV

(PROJECT INCHARGE)

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature

Name : Arpit Kumar Nauhwara

Roll No. : 2001090100006

CERTIFICATE

This is to certify that Project entitled “**House Price Prediction**”. Which is submitted by **Arpit Kumar Nauhwara (2001090100006)** in partial fulfillment of the requirement for the award of degree **Bachelor of Technology in Department of Computer Science and Engineering** from **Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh** is a record of the candidate’s own work carried out by him/her under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Signature

Project Incharge: Mr. Rohit Yadav

Signature

HOD CSE: Mr. Anand Sharma

ACKNOWLEDGEMENT

"OBLIGATION AND REGARD CAN NEVER BE REPAYED"

This preparation of this project owes its gratitude to many personnel, without their cooperation and efforts the project would not have seen the light of the day.

First of all, I would like to thanks **Mr. Anand Sharma**, Head of the Department of CSE, for his support and facility provided for completion of the course.

We express our deep sense of gratitude and thanks to Mr. Rohit Yadav for their kind permission to undergo this course in instructions and for providing the constant encouragement, guidance and valuable suggestions throughout the progress of this work.

We also pay our healthy regard and warm thanks for our project guide, **Mr. Rohit Yadav** and other faculty members.

At last, but not the least the course of writing this course we received cooperation, suggestions and encouragement for all our friends and each other. We convey our sincere thanks to them all along with God, our parents, our family and friends and would not forget to appreciate the spirit of our partners who tied up their belts in shipping it up with an equal impact who remained as constant source of inspiration in accomplishing this task all through. We pay also warm thanks to all of them.

THANKING YOU!!

Arpit Kumar Nauhwara (2001090100006)

ABSTRACT

The House Price Prediction Project aims to develop a sophisticated and accurate machine learning model for forecasting residential property prices in Bangalore. Leveraging a comprehensive dataset sourced from Kaggle.com, a prominent data source platform, this project employs state-of-the-art techniques in data preprocessing, feature engineering, and predictive modeling.

The primary goal is to empower homebuyers, sellers, and investors with a reliable tool that provides insights into the intricate dynamics of the Bangalore real estate market. By harnessing the power of machine learning algorithms, the project endeavors to predict house prices based on crucial factors such as location, size, amenities, and temporal trends.

The methodology involves a meticulous data exploration and cleaning process, followed by feature selection and engineering to capture the nuances of the market. Various machine learning algorithms, including regression models, are evaluated and fine-tuned to achieve the highest predictive accuracy.

The project also places a strong emphasis on transparency and interpretability, ensuring that users can understand the factors influencing the model's predictions. The developed model is expected to generalize well to unseen data, providing reliable estimates of house prices for different segments of the market.

Ultimately, the House Price Prediction Project strives to contribute to a more informed and transparent real estate ecosystem, offering users a tool that facilitates prudent decision-making, risk mitigation, and a deeper understanding of the factors shaping property values in Bangalore.

TABLE OF CONTENT

1. Project Description

- 1.1. Problem Statement
- 1.2. Objective
- 1.3. Introduction About Project
- 1.4. Tools and Libraries

2. Generic Flow of Project

3. Working of System

4. User Activity Diagram

5. Prediction Model

6. Data Collection

- 6.1. Dataset Source - Kaggle
- 6.1. Raw Dataset Preview

7. Exploratory Data Analysis

- 7.1. Data Cleaning
- 7.2. Data Analysis
- 7.3. Feature Engineering
- 7.4. Data Normalization
- 7.5 EDA Implementation

8. Machine Learning Model Development

- 8.1. Model Creation
- 8.2. Model Evaluation and Optimization
- 8.3. Model Deployment
- 8.4. Challenges in Machine Learning Model Deployment
- 8.5. Model Development Implementation

9. Web Development & Model Deployment Implementation

9.1. Frontend Development

9.1.1. Web Design Vs Web Development

9.1.2. HTML & CSS Implementation

9.2. Backend Development

9.2.1. Requirements & Project Structure setup

9.2.2. Flask Implementation

10. Website Preview & Tutorial

10.1. Commands to run Website

10.2. Website Preview

10.2. Website Tutorial

11. Github & Open Source

12. Contributions

13. Reference

1. Project Description:

1.1 Problem Statement:

The real estate market in Bangalore has experienced significant fluctuations over the years, driven by various economic, social, and environmental factors. Understanding and predicting house prices in such a dynamic market is crucial for both homebuyers and sellers, as well as for real estate investors. The dataset sourced from kaggle.com, a prominent data source platform. The dataset includes information on various attributes such as location, size, bathroom, and price.

1.2 Objective:

The objective of this project is to develop a robust and accurate predictive model that can estimate house prices in Bangalore based on historical data. By leveraging machine learning algorithms and statistical techniques, we aim to create a model that captures the underlying patterns and trends influencing house prices. The developed model will be valuable for individuals looking to make informed decisions regarding property transactions, as well as for real estate professionals seeking insights into market dynamics.

The specific objectives include:

- (i) Data Collection, Exploration and Preprocessing.
- (ii) Exploratory Data Analysis.
- (iii) Feature Selection and Engineering.
- (iv) Machine Learning Model Development.
- (v) Model Evaluation and Optimization.
- (vi) Model Deployment

1.3 Introduction About Project:

A House Price Prediction Website provides a range of benefits that can be instrumental for various stakeholders in the real estate market. Here's how such a website can be helpful:

Informed Decision-Making:

Buyers: Prospective homebuyers can use predicted house prices to make informed decisions about whether a property is within their budget. This helps streamline the house-hunting process and ensures that they are realistic about their purchasing power.

- Sellers: Property sellers can strategically price their homes based on market predictions, maximizing their chances of attracting potential buyers and optimizing their returns.
- Investors: Real estate investors can use the predictions to identify potential areas for investment, helping them allocate resources effectively and anticipate future returns.

Risk Assessment:

- The predictive modeling can highlight potential risks and uncertainties in the real estate market, enabling users to make risk-informed decisions. This is particularly valuable for investors and developers assessing the viability of new projects.

Time and Cost Savings:

- Homebuyers can save time by focusing on properties within their predicted budget range, reducing the need for extensive property viewings and negotiations.
- Sellers can avoid the inconvenience of overpricing or underpricing their properties by aligning their asking prices with the predicted values, potentially speeding up the selling process.

Market Insights:

- The website provides users with valuable insights into market trends, helping them understand how various factors such as location, size, and amenities impact property values. This knowledge can be used for strategic decision-making.

Transparency:

- By displaying the factors considered in the prediction model and the weightage assigned to each, the website promotes transparency. Users gain a clearer

understanding of how predictions are generated, enhancing trust in the platform.

Personalization:

- Users can customize predictions based on their specific criteria, tailoring the information to meet their unique preferences and requirements. This personalization ensures that the predictions are directly relevant to the user's individual circumstances.

Market Awareness:

- Users become more aware of market dynamics, allowing them to stay ahead of changes in the real estate landscape. This awareness is beneficial for making strategic decisions in response to evolving market conditions.

Educational Value:

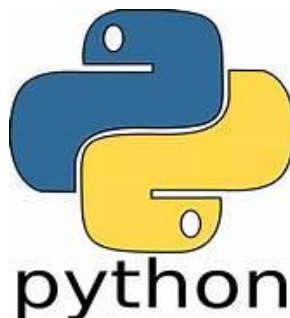
- The website can serve as an educational tool, helping users understand the various factors that influence house prices. This knowledge empowers individuals to navigate the real estate market more confidently.

In summary, a House Price Prediction Website serves as a valuable resource by providing accurate predictions, offering market insights, and promoting transparency. It enhances the decision-making process for buyers, sellers, and investors in the dynamic and complex real estate market.

1.4 Tools and Libraries

1.1 Tool's

1.1.1 Python: Python is a versatile, high-level programming language known for its simplicity and readability. It supports object-oriented, procedural, and functional programming paradigms. Python's extensive standard library provides modules for a wide range of tasks. It's widely used in web development, scientific computing, data analysis, artificial intelligence, and more.



1.1.2 Jupiter NoteBook: Jupyter Notebook is an interactive, open-source web application for creating and sharing documents that contain live code, visualizations, text, and equations. It supports various programming languages, including Python, R, and Julia. It's widely used for data analysis, machine learning, and scientific research due to its interactive and reproducible nature.



1.1.3 Flask: Flask is a lightweight, open-source web framework for Python. It simplifies the process of building web applications by providing tools and libraries to handle tasks like routing, request handling, and templating. Flask is known for its simplicity, flexibility, and ease of learning, making it popular for small to medium-sized projects.



1.1.4 HTML: HTML (Hypertext Markup Language) is the standard language for creating web pages. It uses tags to structure content, such as headings, paragraphs, and links. HTML documents are interpreted by web browsers to display text, images, and multimedia. It forms the backbone of virtually every webpage on the internet.



1.1.5 Github: GitHub is a web-based platform for version control using Git. It enables collaborative software development by allowing multiple contributors to work on projects simultaneously. Users can host repositories, track changes, manage issues, and deploy applications. It's widely used for open-source projects and facilitates code collaboration and project management.



1.1.6 CSS: Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS is among the core languages of the open web and is standardized across Web browsers according to W3C specifications. Previously, the development of various parts of CSS specification was done synchronously, which allowed the versioning of the latest recommendations.



1.2 Libraries,

1.2.1 Pandas: Pandas is an open-source data manipulation and analysis library for Python. It provides easy-to-use data structures, like DataFrames and Series, and functions for reading, cleaning, and transforming data. Pandas is widely used in data science for tasks like data exploration, cleaning, and preparation, making it a fundamental tool for data analysis.



1.2.2 Numpy: NumPy is a powerful open-source library for numerical computations in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays. NumPy is fundamental to scientific computing, used extensively in areas like linear algebra, calculus, and statistics.



1.2.3 Seaborn: Seaborn is a statistical data visualization library in Python. Built on top of Matplotlib, it provides a high-level interface for creating informative and attractive statistical graphics. Seaborn simplifies complex visualization tasks, offering functions for creating visually appealing plots for data exploration and presentation in areas like regression, distribution, and correlation analysis.



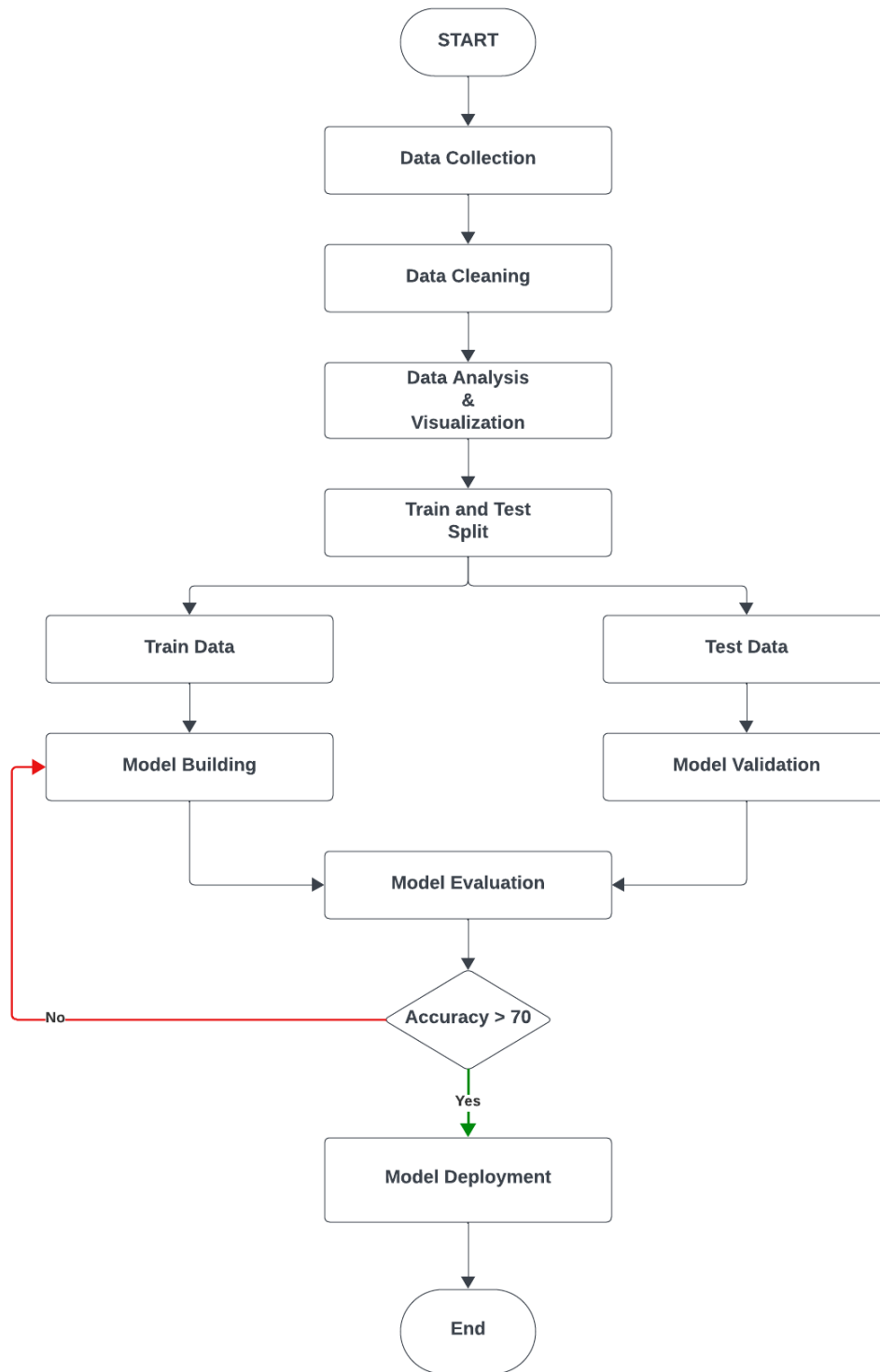
1.2.4 Matplotlib: Matplotlib is a comprehensive data visualization library in Python. It provides a wide range of high-quality plots, including line plots, bar charts, scatter plots, and more. Matplotlib is customizable and suitable for various data representation needs, making it a fundamental tool for visualizing and conveying insights from data in scientific and engineering fields.



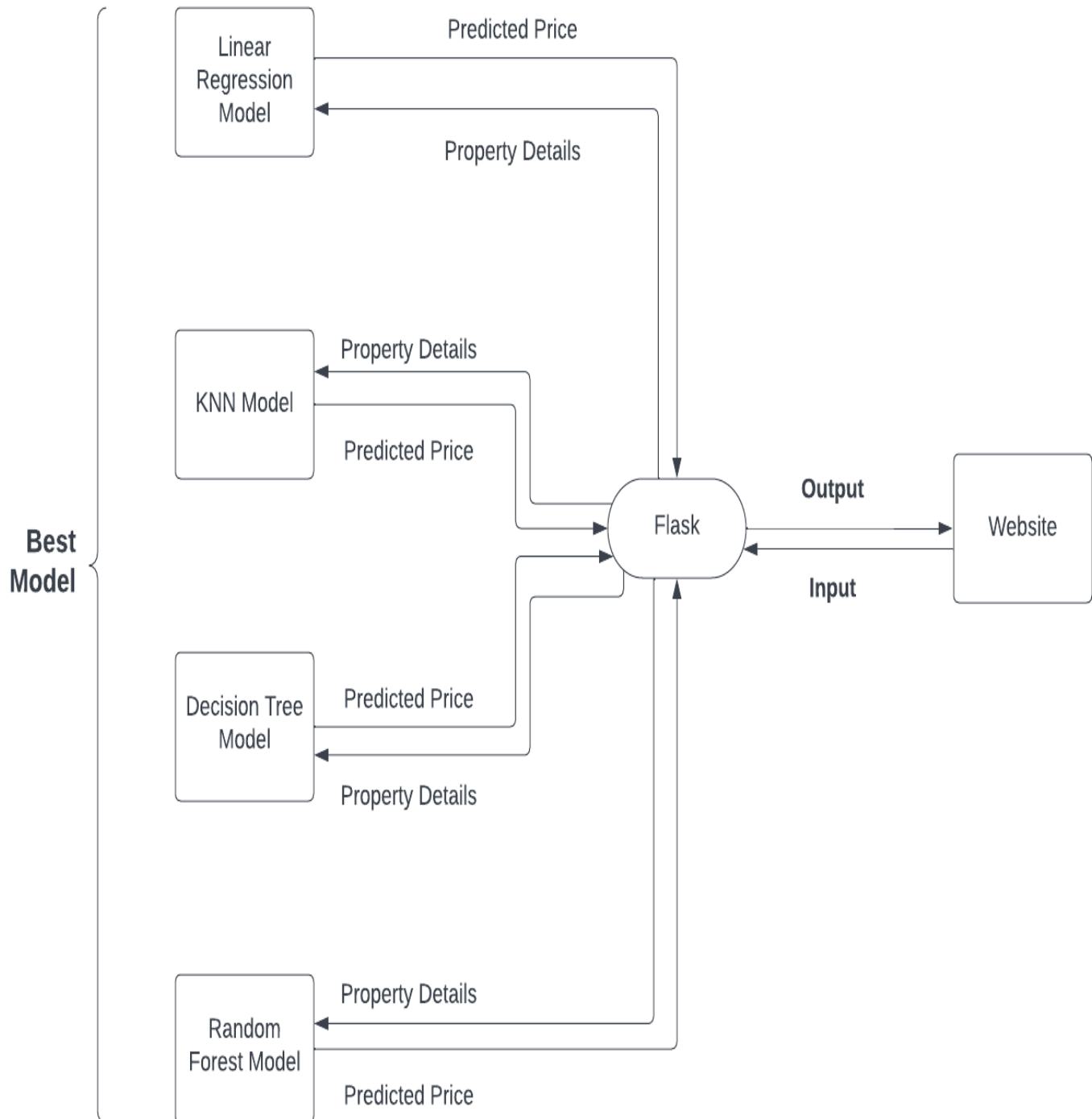
1.2.5 Scikit Learn: Scikit-learn is an open-source machine learning library for Python. It offers a wide array of tools for tasks like classification, regression, clustering, and more. With a user-friendly interface, it supports data preprocessing, model training, and evaluation. Scikit-learn is widely used for building and deploying machine learning models in various domains.



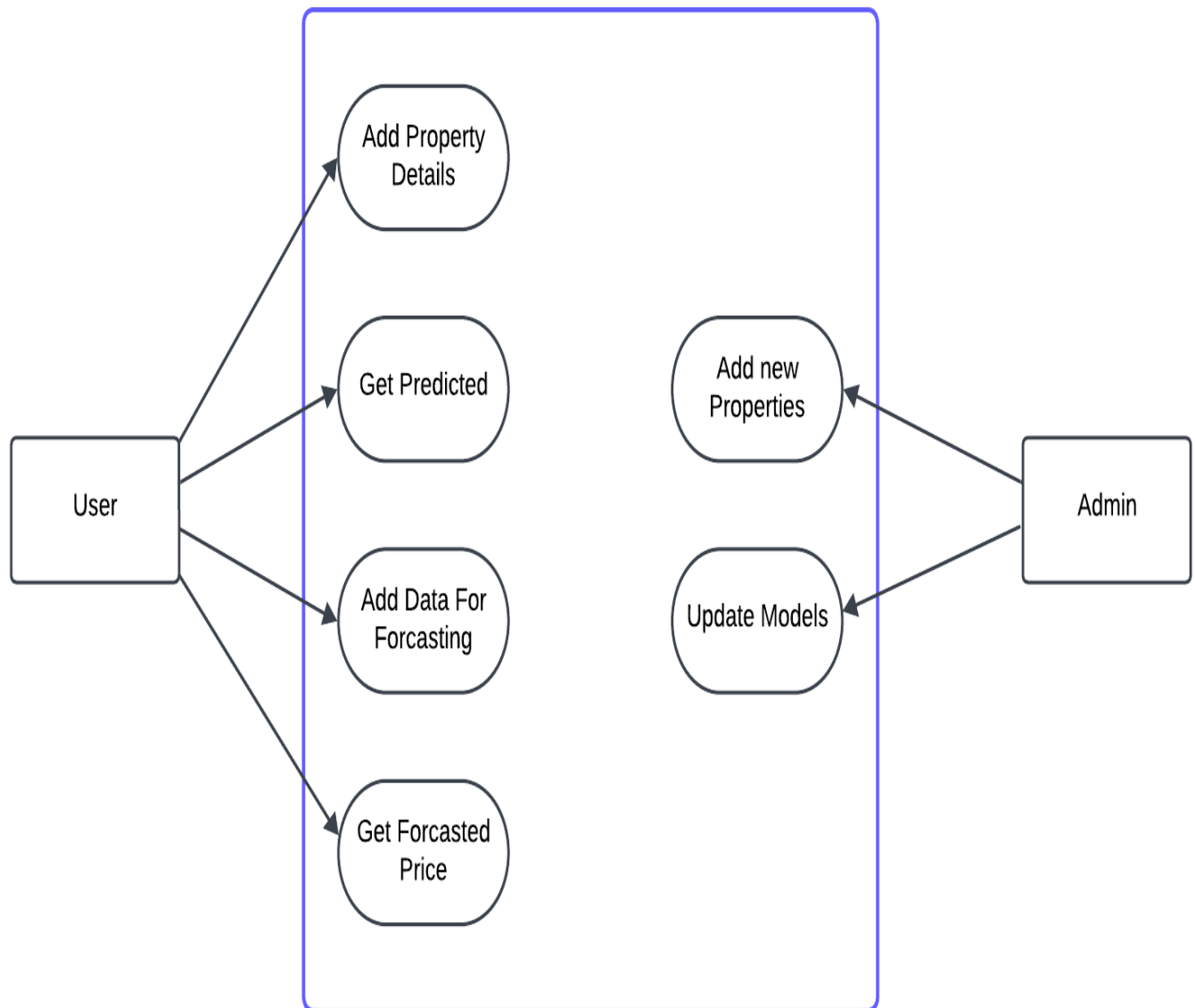
2. Genric Flow of Chart



3. Working Of System

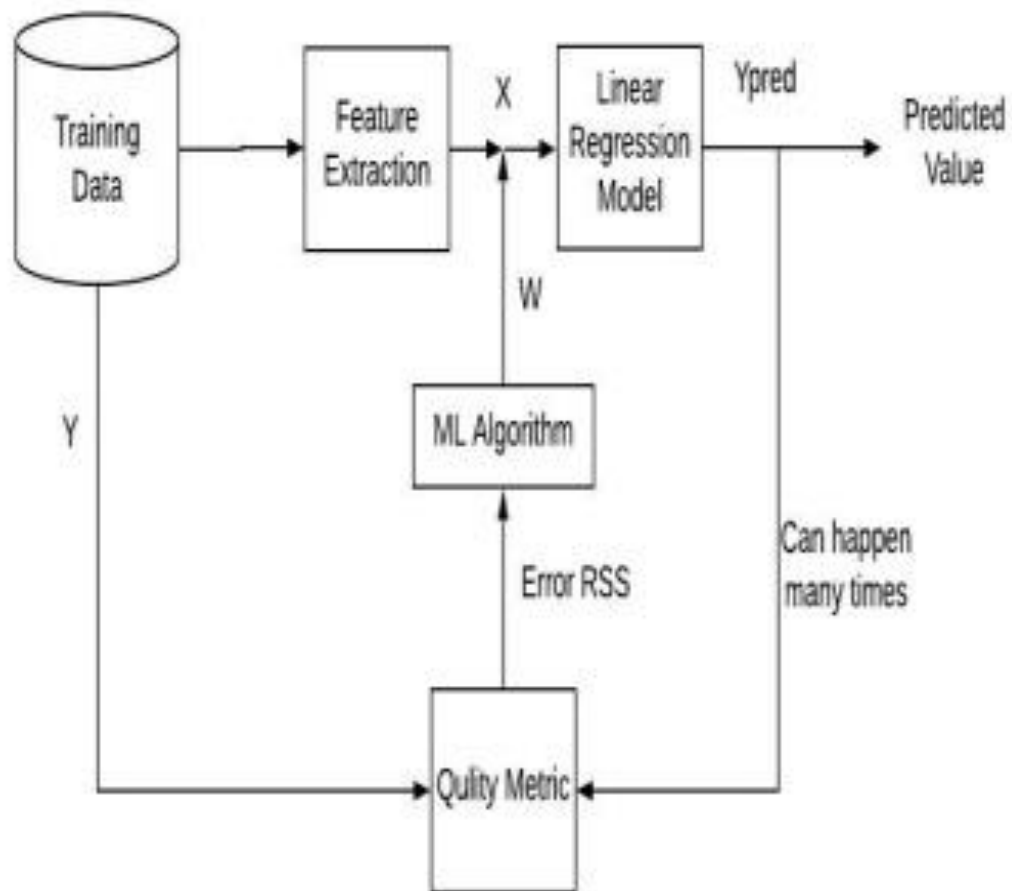


4. User Activity Diagram



User Activity Diagram

5. Prediction Model



6. Data Collection,

Data collection is the systematic process of gathering and recording information from various sources. It involves defining the objectives, selecting appropriate methods (surveys, observations, interviews, etc.), and collecting relevant data points. Quality control measures ensure accuracy and reliability. Ethical considerations, like consent and privacy, are crucial. Once collected, data is organized for analysis. Effective data collection is vital for informed decision-making and research in fields ranging from business and healthcare to academia and social sciences. It forms the foundation for generating meaningful insights and drawing conclusions.

For this project we used the data that is available on kaggle. There are 9 columns and 13321 Rows.

These are the major point about the data set.

Area type: defines the build up area of property

Availability: tells about when the property is available

Location: defines the various location in Bangalore

Size: defines the BHK

Society: defines various society in Bangalore

Square ft: defines the total sq/ft of property

Bathroom: defines number of bathroom available in a property

Balcony: defines number of balcony available in a property

Price: price of the property

6.1. Dataset Source – Kaggle

The screenshot shows the Kaggle website interface. The browser address bar displays the URL: kaggle.com/datasets/amanabhaioy/bengaluru-house-price-data. The Kaggle logo is in the top left, and navigation links for Home, Competitions, Datasets, Models, Code, Discussions, Learn, and More are on the left sidebar. The main header includes a search bar, 'Sign In', and 'Register' buttons. The dataset title 'Bengaluru House price data' is prominently displayed, along with a '377' badge, a 'New Notebook' button, and a 'Download (200 kB)' button. Below the title, tabs for 'Data Card', 'Code (241)', 'Discussion (1)', and 'Suggestions (0)' are visible. The 'Data Card' tab is active, showing the file 'Bengaluru_House_Data.csv' (938.02 kB) with download, share, and expand icons. The file is shown in 'Detail' view, indicating it has 9 of 9 columns. The 'About this file' section contains the following text: 'What are the things that a potential home buyer considers before purchasing a house? The location, the size of the property, vicinity to offices, schools, parks, restaurants, hospitals or the stereotypical white picket fence? What about the most important factor — the price? Now with the lingering impact of demonetization, the enforcement of the Real Estate (Regulation and Development) Act (RERA), and the lack of trust in property developers in the city, housing units sold across India in 2017 dropped by 7 percent. In fact, the property prices in Bengaluru fell by almost 5 percent in the second half of 2017, said a study published by property consultancy Knight Frank. For example, for a potential homeowner, over 9,000 apartment projects and flats for sale are available in the range of ₹42-52 lakh, followed by over 7,100 apartments that are in the ₹52-62 lakh budget segment, says a report by property website Makaan. According to the study, there are over 5,000 projects in the ₹15-25 lakh budget segment followed by those in the ₹34-43 lakh budget category. Buying a home, especially in a city like Bengaluru, is a tricky choice. While the major factors are usually the same for all metros,'. On the right, the 'Data Explorer' section shows 'Version 2 (938.02 kB)' and a file icon for 'Bengaluru_House_Data.csv'.

Dataset Link: <https://www.kaggle.com/datasets/amanabhaioy/bengaluru-house-price-data>

6.1. Raw Dataset Preview

AutoSave

off

Bengaluru_House_D... • Saved to this PC

Search

Anurag Gupta AG

FileHomeInsertPage LayoutFormulasDataReviewViewHelp

CutCopyPasteFormat PainterClipboard

Font

Alignment

Number

Styles

Cells

Editing

Add-ins

General

Conditional Formatting

Format as Table

Cell Styles

Insert

Delete

Format

AutoSum

Sort & Filter

Find & Select

Clear

Comments

Share

135

fx

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	area_type	availability	location	size	society	total_sqft	bath	balcony	price														
2	Super built	19-Dec	Electronic	2 BHK	Coomee	1056	2	1	39.07														
3	Plot Area	Ready To I	Chikka Tiru	4 Bedroom	Theanmp	2600	5	3	120														
4	Built-up	A Ready To I	Uttarahalli	3 BHK		1440	2	3	62														
5	Super built	Ready To I	Lingadhee	3 BHK	Soiewre	1521	3	1	95														
6	Super built	Ready To I	Kothanur	2 BHK		1200	2	1	51														
7	Super built	Ready To I	Whitefield	2 BHK	DuenaTa	1170	2	1	38														
8	Super built	18-May	Old Airport	4 BHK	Jaades	2732	4		204														
9	Super built	Ready To I	Rajaji Nagar	4 BHK	Brway G	3300	4		600														
10	Super built	Ready To I	Marathahalli	3 BHK		1310	3	1	63.25														
11	Plot Area	Ready To I	Gandhi Ba	6 Bedroom		1020	6		370														
12	Super built	18-Feb	Whitefield	3 BHK		1800	2	2	70														
13	Plot Area	Ready To I	Whitefield	4 Bedroom	Prrry M	2785	5	3	295														
14	Super built	Ready To I	7th Phase	2 BHK	Shncyys	1000	2	1	38														
15	Built-up	A Ready To I	Gottigere	2 BHK		1100	2	2	40														
16	Plot Area	Ready To I	Sarjapur	3 Bedroom	Skityer	2250	3	2	148														
17	Super built	Ready To I	Mysore Rc	2 BHK	PrntaEn	1175	2	2	73.5														
18	Super built	Ready To I	Bisuvanah	3 BHK	Prityel	1180	3	2	48														
19	Super built	Ready To I	Raja Rajes	3 BHK	GrrvaGr	1540	3	3	60														
20	Super built	Ready To I	Ramakrish	3 BHK	PeBayle	2770	4	2	290														
21	Super built	Ready To I	Manayata	2 BHK		1100	2	2	48														
22	Built-up	A Ready To I	Kengeri	1 BHK		600	1	1	15														
23	Super built	19-Dec	Binny Pete	3 BHK	She 2rk	1755	3	1	122														
24	Plot Area	Ready To I	Thanisand	4 Bedroom	Soitya	2800	5	2	380														
25	Super built	Ready To I	Bellandur	3 BHK		1767	3	1	103														
26	Super built	18-Nov	Thanisand	1 RK	Bhe 2ko	510	1	0	25.25														
27	Super built	18-May	Mangamm	3 BHK		1250	3	2	56														
28	Super built	Ready To I	Whitefield	3 BHK		660	1	1	23.1														

<>

Bengaluru_House_Data

+

Ready

Accessibility: Unavailable

100%

7. **Exploratory Data Analysis**

Exploratory Data Analysis (EDA) is a crucial phase in data analysis. It involves summarizing and visualizing data to understand its characteristics, patterns, and distributions. EDA helps identify outliers, missing values, and potential relationships between variables. Techniques like histograms, scatter plots, and summary statistics aid in uncovering insights. It guides preprocessing steps and informs the choice of analytical techniques. EDA is fundamental for gaining a comprehensive understanding of the data before applying more advanced modeling or statistical methods.

7.1 **Data Cleaning:**

Data cleaning is the process of identifying and correcting errors, inconsistencies, and inaccuracies in a dataset. It involves tasks like handling missing values, removing duplicates, and correcting anomalies. Techniques such as imputation, outlier detection, and standardization are applied to ensure data quality. Cleaning enhances the reliability and validity of analysis results. It is a critical step in preparing data for modeling, as accurate and consistent data is essential for generating meaningful insights and making informed decisions in various fields, from research to business analytics.

7.2 **Data Analysis:**

Data analysis is the systematic process of examining, cleaning, transforming, and interpreting data to extract meaningful insights and inform decision-making. It involves applying statistical, mathematical, and computational techniques to identify patterns, trends, and relationships within the dataset. Visualization tools and analytical models are often used to aid in the exploration of data. The results of data analysis can lead to informed conclusions, predictions, and recommendations, making it a crucial step in fields like business, science, and research.

7.3 **Feature Engineering:**

Feature engineering is a vital step in preparing data for machine learning. It involves creating new features or transforming existing ones to enhance model performance. Techniques include one-hot encoding, scaling, and creating interaction terms. Domain knowledge is crucial for selecting relevant features. Careful engineering can uncover hidden patterns and relationships, improving a model's ability to make accurate predictions or classifications. It plays a pivotal role in maximizing the effectiveness of machine learning algorithms in various applications.

7.4 **Data Normalization:**

Data normalization is a preprocessing technique used to scale and standardize numerical features in a dataset. It ensures that all features have similar scales, preventing certain variables from dominating others in a machine learning model. Common methods include Min-Max scaling and Z-score normalization. Normalization improves model performance and convergence, particularly for algorithms sensitive to feature scales. It is a crucial step in data preprocessing to ensure fair and accurate model predictions across different types of input variables.

7.5 EDA Implementation

Bangalore House Price Prediction.

Description.

What are the things that a potential home buyer considers before purchasing a house? The location, the size of the property, vicinity to offices, schools, parks, restaurants, hospitals or the stereotypical white picket fence? What about the most important factor — the price?

Now with the lingering impact of demonetization, the enforcement of the Real Estate (Regulation and Development) Act (RERA), and the lack of trust in property developers in the city, housing units sold across India in 2017 dropped by 7 percent. In fact, the property prices in Bengaluru fell by almost 5 percent in the second half of 2017, said a study published by property consultancy Knight Frank.

For example, for a potential homeowner, over 9,000 apartment projects and flats for sale are available in the range of ₹42-52 lakh, followed by over 7,100 apartments that are in the ₹52-62 lakh budget segment, says a report by property website Makaan. According to the study, there are over 5,000 projects in the ₹15-25 lakh budget segment followed by those in the ₹34-43 lakh budget category.

Buying a home, especially in a city like Bengaluru, is a tricky choice. While the major factors are usually the same for all metros, there are others to be considered for the Silicon Valley of India. With its help millennial crowd, vibrant culture, great climate and a slew of job opportunities, it is difficult to ascertain the price of a house in Bengaluru.

Problem Statement.

By analyzing these Bangalore house data we will determine the approximate price for the houses.

Data Description.

- **Area_type** : Description of the area.
- **Availability** : When it can be possessed or when it is ready.
- **Location** : Where it is located in Bengaluru.
- **Size** : BHK or Bedrooms.
- **Society** : To which society it belongs.
- **Total_sqft** : Size of the property in sq.ft.

- **Bath** : No. of Bathrooms.
- **Balcony** : No. of the Balcony.
- **Price** : Value of the property in lakhs (Indian Rupee - ₹).

Business Objectives and Constraints.

1. The cost of a mis-classification can be high.
2. There is strict latency concerns.
3. From this project we will able to understand how house prices depend on other factors.

Importing Libraries.

```
In [149]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
from IPython.display import Image
import warnings
warnings.filterwarnings('ignore')
```

Loading and Checking the Dataset.

```
In [150]: data=pd.read_csv("D:/acet project/dataset/Bengaluru_House_Data.csv")
df = data.copy()
```

```
In [151]: df.head()
```

```
Out[151]:
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	p
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	3
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	12
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	6
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	9
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	5

Exploratory Data Analysis (EDA).

Checking Shape of the Dataset.

```
In [152]: df.shape
```

```
Out[152]: (13320, 9)
```

How Many Columns are Present in the Dataset?

```
In [153]: df.columns
```

```
Out[153]: Index(['area_type', 'availability', 'location', 'size', 'society',  
                'total_sqft', 'bath', 'balcony', 'price'],  
                dtype='object')
```

```
In [154]: df.columns.nunique()
```

```
Out[154]: 9
```

Informations about the Dataset.

```
In [155]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 13320 entries, 0 to 13319  
Data columns (total 9 columns):  
#   Column          Non-Null Count  Dtype  
---  ---  
0   area_type       13320 non-null  object  
1   availability     13320 non-null  object  
2   location        13319 non-null  object  
3   size            13304 non-null  object  
4   society         7818 non-null   object  
5   total_sqft      13320 non-null  object  
6   bath            13247 non-null  float64  
7   balcony         12711 non-null  float64  
8   price           13320 non-null  float64  
dtypes: float64(3), object(6)  
memory usage: 936.7+ KB
```

To know the description about the dataset

```
In [156]: df.describe()
```

```
Out[156]:
```

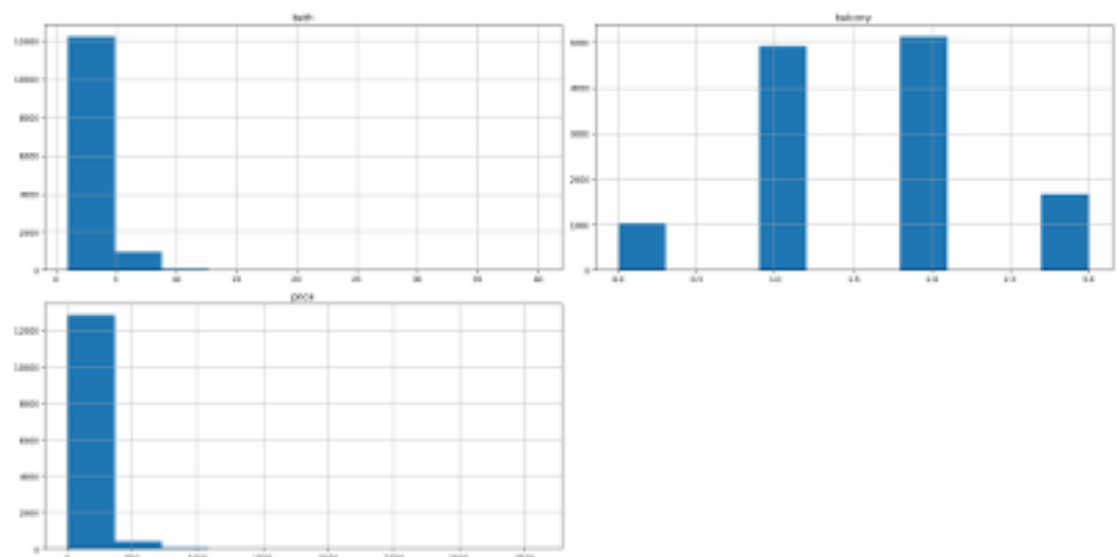
	bath	balcony	price
count	13247.000000	12711.000000	13320.000000
mean	2.692610	1.584376	112.565627
std	1.341458	0.817263	148.971674
min	1.000000	0.000000	8.000000
25%	2.000000	1.000000	50.000000
50%	2.000000	2.000000	72.000000
75%	3.000000	2.000000	120.000000
max	40.000000	3.000000	3600.000000

```
In [157]: df.describe().T
```

```
Out[157]:
```

	count	mean	std	min	25%	50%	75%	max
bath	13247.0	2.692610	1.341458	1.0	2.0	2.0	3.0	40.0
balcony	12711.0	1.584376	0.817263	0.0	1.0	2.0	2.0	3.0
price	13320.0	112.565627	148.971674	8.0	50.0	72.0	120.0	3600.0

```
In [158]: df.hist()  
plt.tight_layout()  
plt.show()
```



Checking if there is some null values or not.

```
In [159]: df.isnull().sum()
```

```
Out[159]: area_type      0
availability    0
location        1
size            16
society         5502
total_sqft      0
bath            73
balcony         609
price           0
dtype: int64
```

Performing Group by operation on Area Type

```
In [160]: df.groupby("area_type")["area_type"].agg("count")
```

```
Out[160]: area_type
Built-up Area      2418
Carpet Area         87
Plot Area          2025
Super built-up Area 8790
Name: area_type, dtype: int64
```

Checking what different "Area Types" are present in the Dataset

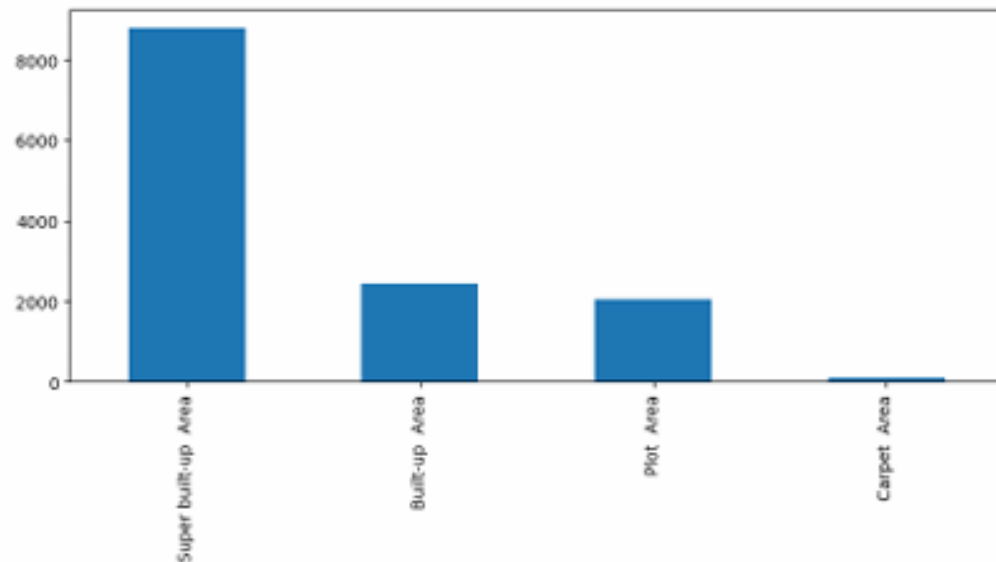
```
In [161]: df['area_type'].unique()
```

```
Out[161]: array(['Super built-up Area', 'Plot Area', 'Built-up Area',
                  'Carpet Area'], dtype=object)
```

Data Visualization.

```
In [162]: plt.figure(figsize=(10,4))
df.area_type.value_counts().plot(kind='bar')
```

Out[162]: <Axes: >

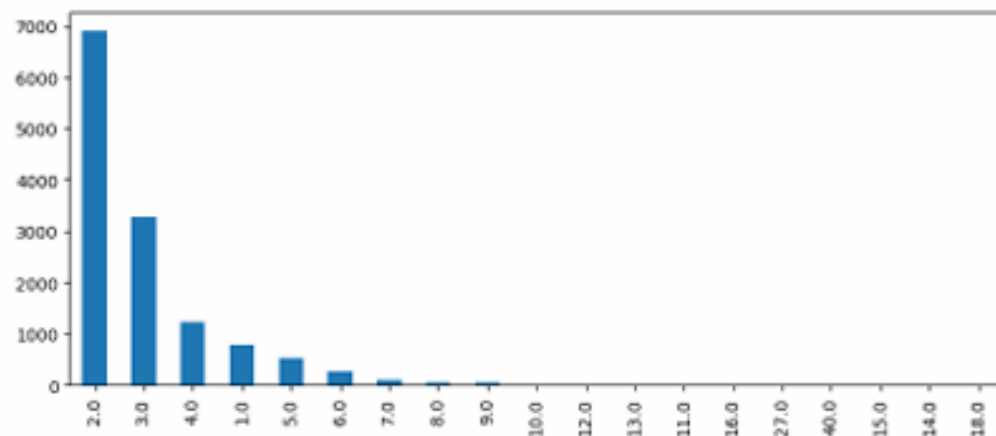


```
In [163]: df['bath'].unique()
```

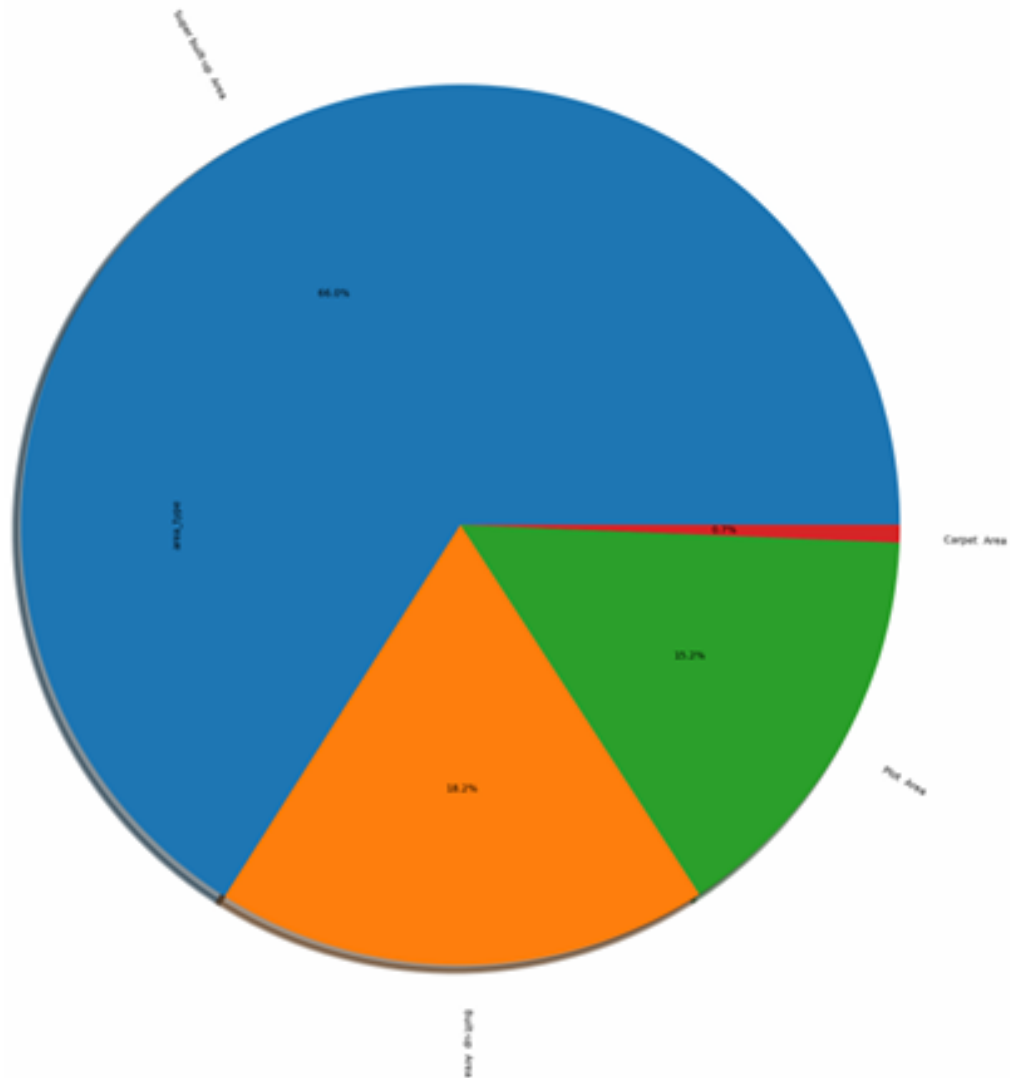
Out[163]: array([2., 5., 3., 4., 6., 1., 9., nan, 8., 7., 11., 10., 14.,
 27., 12., 16., 40., 15., 13., 18.])

```
In [164]: plt.figure(figsize=(10,4))
df.bath.value_counts().plot(kind='bar')
```

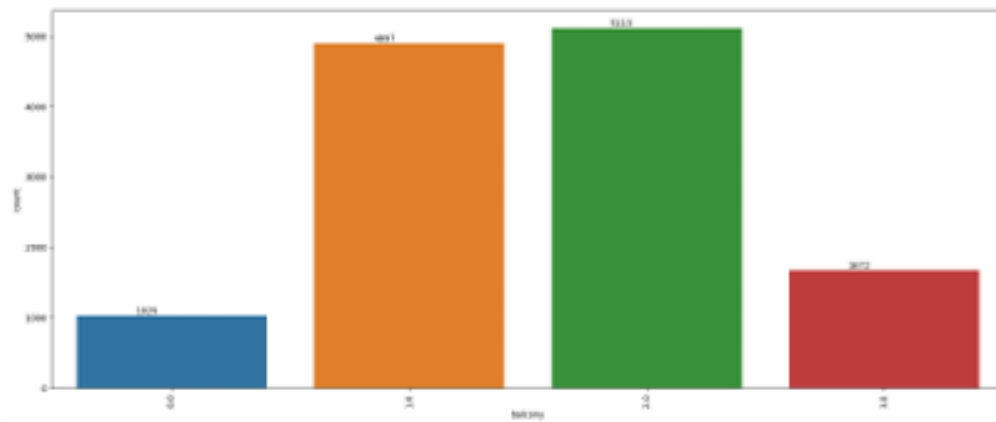
Out[164]: <Axes: >



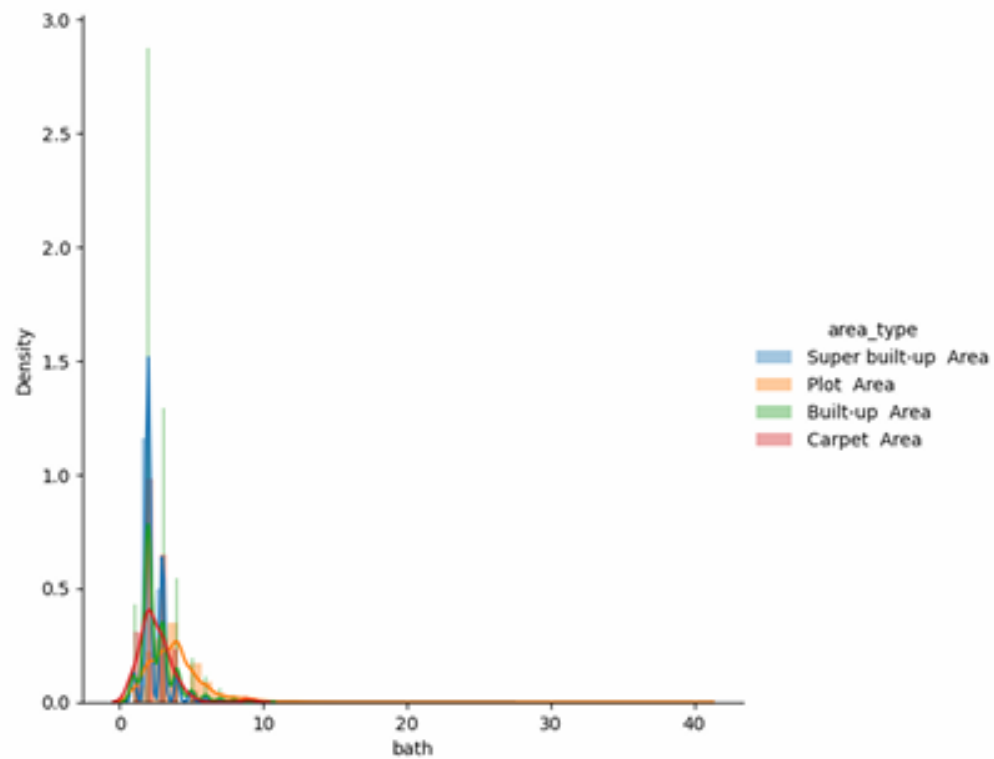
```
In [165]: (df["area_type"].value_counts()).plot.pie(autopct="%.1f%%", shadow=True, rot  
plt.show()
```



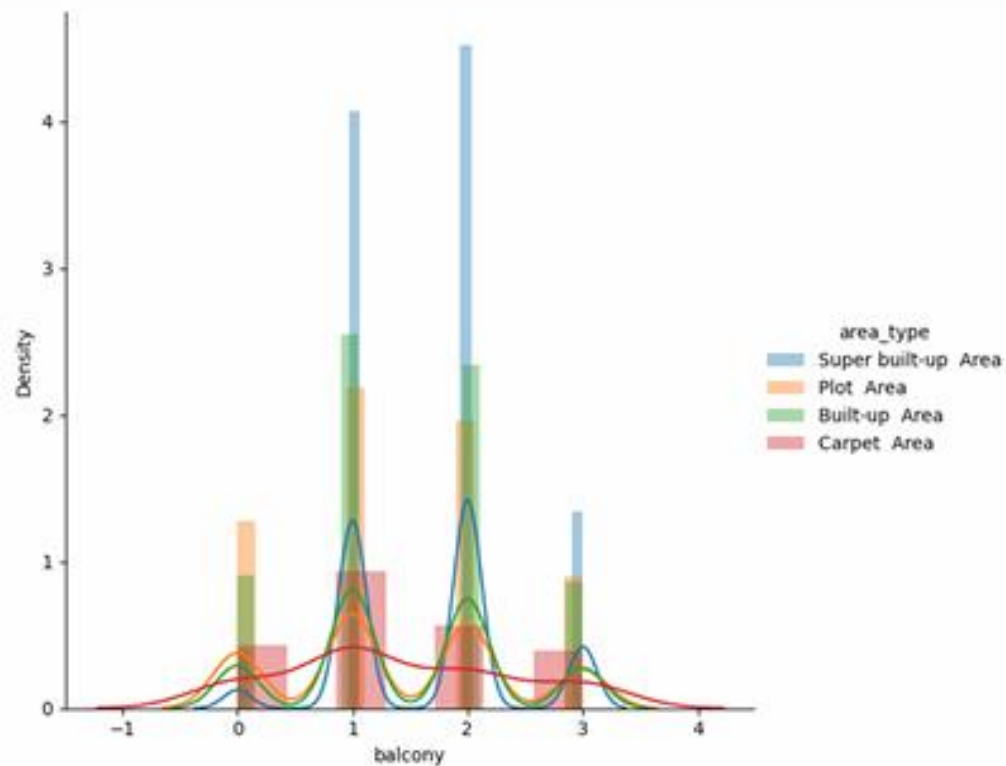
```
In [166]: plt.figure(figsize = (20,8))
ax=sns.countplot(x = 'balcony', data = df)
plt.xticks(rotation = 90)
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va
```



```
In [167]: sns.FacetGrid(df, hue='area_type', height=6).map(sns.distplot, 'bath').add_
plt.show()
```



```
In [168]: sns.FacetGrid(df, hue='area_type', height=6).map(sns.distplot, 'balcony').a
plt.show()
```



Dropping less important features.

```
In [169]: df = df.drop(["area_type", "society", "balcony", "availability"], axis = "co
```

```
In [170]: df.shape
```

```
Out[170]: (13320, 5)
```

Dropping Null Values.

```
In [171]: df=df.dropna()
```

```
In [172]: df.isnull().sum()
```

```
Out[172]: location      0
size      0
total_sqft  0
bath      0
price     0
dtype: int64
```



```
In [173]: df.shape
```

```
Out[173]: (13246, 5)
```

Feature Engineering.

Applying unique function on feature called Size.

```
In [174]: df["size"].unique()
```

```
Out[174]: array(['2 BHK', '4 Bedroom', '3 BHK', '4 BHK', '6 Bedroom', '3 Bedroom',  
                '1 BHK', '1 RK', '1 Bedroom', '8 Bedroom', '2 Bedroom',  
                '7 Bedroom', '5 BHK', '7 BHK', '6 BHK', '5 Bedroom', '11 BHK',  
                '9 BHK', '9 Bedroom', '27 BHK', '10 Bedroom', '11 Bedroom',  
                '10 BHK', '19 BHK', '16 BHK', '43 Bedroom', '14 BHK', '8 BHK',  
                '12 Bedroom', '13 BHK', '18 Bedroom'], dtype=object)
```

From the above we can clearly see that Bedroom is represented with 2 different methods. One is BHK and the other one is Bedroom. So we are making a new column called BHK and we are discarding all the units (like BHK, Bedroom).

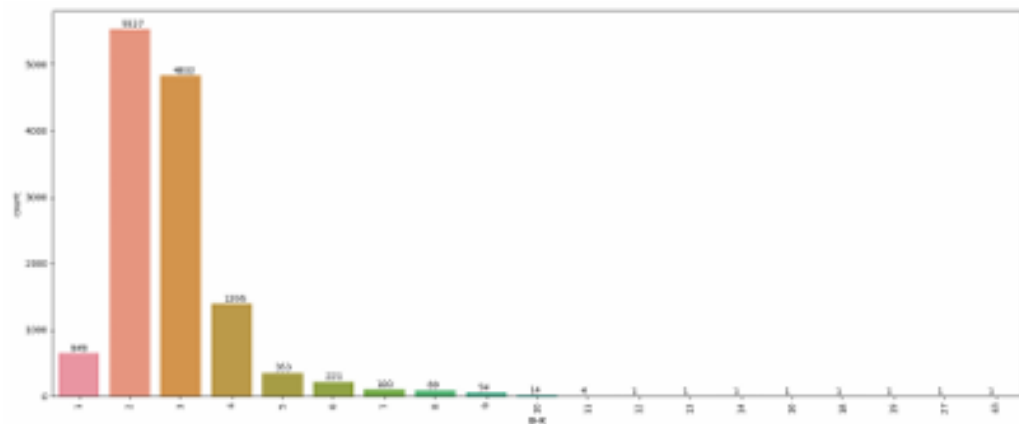
```
In [175]: df['BHK'] = df["size"].apply(lambda x: int(x.split(" ")[0]))
```

```
In [176]: df.head()
```

```
Out[176]:
```

	location	size	total_sqft	bath	price	BHK
0	Electronic City Phase II	2 BHK	1056	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00	4
2	Uttarahalli	3 BHK	1440	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521	3.0	95.00	3
4	Kothanur	2 BHK	1200	2.0	51.00	2

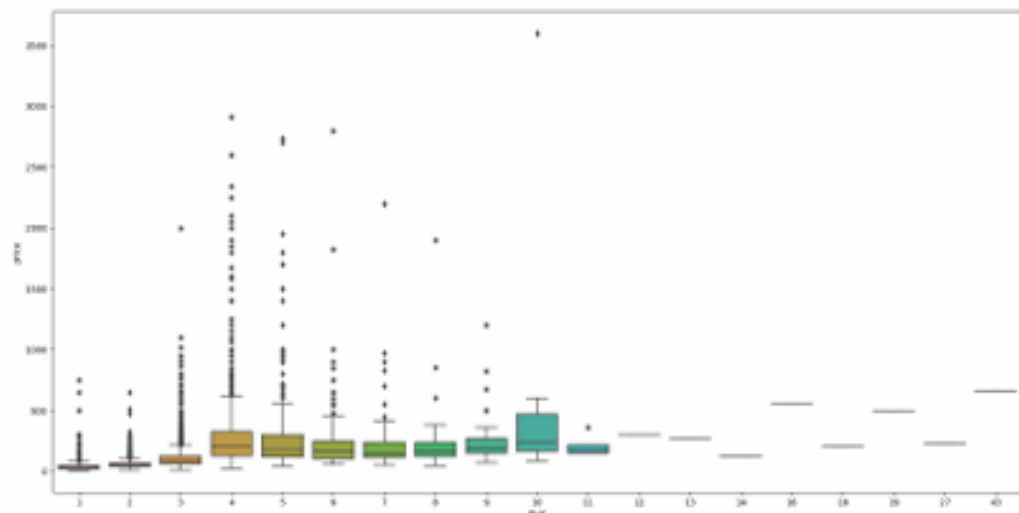
```
In [177]: plt.figure(figsize = (20,8))
ax=sns.countplot(x = 'BHK', data = df)
plt.xticks(rotation = 90)
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va
```



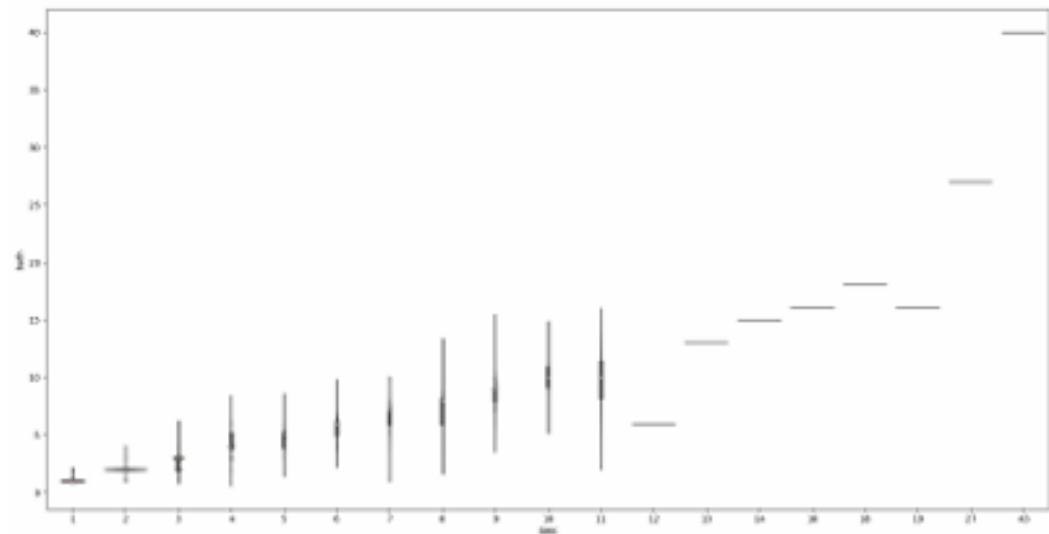
```
In [178]: df.total_sqft.unique()
```

```
Out[178]: array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

```
In [179]: sns.boxplot(x = 'BHK', y = 'price', data = df)
plt.show()
```



```
In [180]: sns.violinplot(x='BHK', y = 'bath', data = df)
plt.show()
```



Exploring total_sqft feature.

```
In [181]: def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

```
In [182]: df[~df["total_sqft"].apply(is_float)].head(10)
```

```
Out[182]:
```

	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
165	Sarjapur	2 BHK	1145 - 1340	2.0	43.490	2
188	KR Puram	2 BHK	1015 - 1540	2.0	56.800	2
410	Kengeri	1 BHK	34.46Sq. Meter	1.0	18.500	1
549	Hennur Road	2 BHK	1195 - 1440	2.0	63.770	2
648	Arekere	9 Bedroom	4125Perch	9.0	265.000	9
661	Yelahanka	2 BHK	1120 - 1145	2.0	48.130	2
672	Bettahalsoor	4 Bedroom	3090 - 5002	4.0	445.000	4

From the above we can see that total_sqft can be a range (say, 3090-5002). For such cases we can just take average of the minimum and maximum value in the range. There are other cases such as 34.46Sq. Meter which one can convert to square ft using unit conversion. So, we are going to just drop such corner cases to keep things simple.

Converting Sq.Ft to Number.

```
In [183]: def convert_sqft_to_number(x):
          tokens = x.split("-")
          if len(tokens) == 2:
              return (float(tokens[0])+float(tokens[1]))/2
          try:
              return float(x)
          except:
              return None
```

```
In [184]: df = df.copy()
          df["total_sqft"] = df["total_sqft"].apply(convert_sqft_to_number)
          df.head(10)
```

```
Out[184]:
```

	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
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3
4	Kothanur	2 BHK	1200.0	2.0	51.00	2
5	Whitefield	2 BHK	1170.0	2.0	38.00	2
6	Old Airport Road	4 BHK	2732.0	4.0	204.00	4
7	Rajaji Nagar	4 BHK	3300.0	4.0	600.00	4
8	Marathahalli	3 BHK	1310.0	3.0	63.25	3
9	Gandhi Bazar	6 Bedroom	1020.0	6.0	370.00	6

Here, adding a new feature called Price per Square Feet.

```
In [185]: df = df.copy()
          df["price_per_sqft"] = df["price"]*100000/df["total_sqft"]
          df.head()
```

```
Out[185]:
```

	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

Here, we are going to use Dimensionality Reduction for the data which are categorical variable. We need to apply Dimensionality Reduction here to reduce number of locations.

```
In [186]: df.location = df.location.apply(lambda x: x.strip())
location_stats = df['location'].value_counts(ascending=False)
location_stats
```

```
Out[186]: Whitefield          535
Sarjapur Road          392
Electronic City        384
Kanakpura Road        266
Thanisandra           236
...
Vasantapura main road    1
Bapuji Layout           1
1st Stage Radha Krishna Layout  1
BEML Layout 5th stage    1
Abshot Layout           1
Name: location, Length: 1293, dtype: int64
```

```
In [187]: len(location_stats[location_stats<=10])
```

```
Out[187]: 1052
```

```
In [188]: location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10
```

```
Out[188]: Naganathapura          10
Sadashiva Nagar          10
Nagappa Reddy Layout     10
BTM 1st Stage            10
Sector 1 HSR Layout      10
..
Vasantapura main road    1
Bapuji Layout           1
1st Stage Radha Krishna Layout  1
BEML Layout 5th stage    1
Abshot Layout           1
Name: location, Length: 1052, dtype: int64
```

```
In [189]: df.location = df.location.apply(lambda x: 'other' if x in location_stats_le
len(df.location.unique())
```

```
Out[189]: 242
```

```
In [190]: df.head()
```

```
Out[190]:
```

	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

Here we will discard some more data. Because, normally if a 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 threshold per bhk to be 300 sqft.

```
In [191]: df[df.total_sqft/df.BHK<300].head()
```

```
Out[191]:
```

	location	size	total_sqft	bath	price	BHK	price_per_sqft
9	other	6 Bedroom	1020.0	6.0	370.0	6	36274.509804
45	HSR Layout	8 Bedroom	600.0	9.0	200.0	8	33333.333333
58	Murugeshpalya	6 Bedroom	1407.0	4.0	150.0	6	10660.980810
68	Devarachikkanahalli	8 Bedroom	1350.0	7.0	85.0	8	6296.296296
70	other	3 Bedroom	500.0	3.0	100.0	3	20000.000000

```
In [192]: df=df[~(df.total_sqft/df.BHK<300)]  
df.shape
```

```
Out[192]: (12502, 7)
```

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.

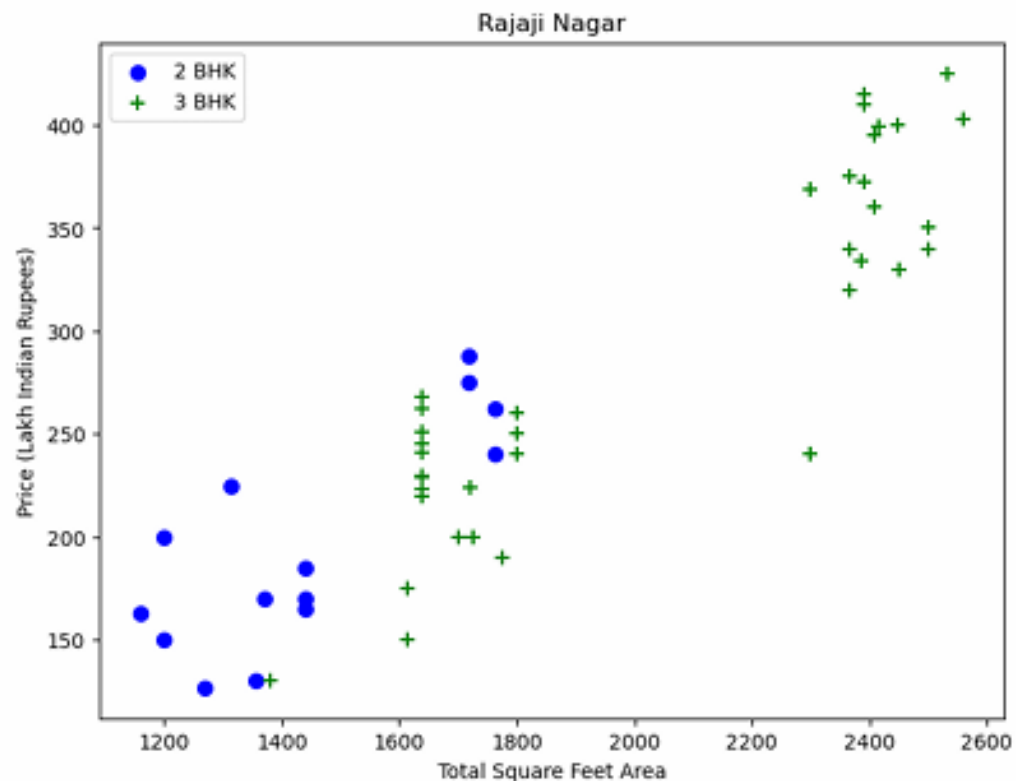
```
In [193]: def remove_pps_outliers(df):  
    df_out = pd.DataFrame()  
    for key, subdf in df.groupby('location'):  
        m = np.mean(subdf.price_per_sqft)  
        st = np.std(subdf.price_per_sqft)  
        reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<(m+st))]  
        df_out = pd.concat([df_out,reduced_df],ignore_index=True)  
    return df_out  
df = remove_pps_outliers(df)  
df.shape
```

```
Out[193]: (10241, 7)
```

Plotting the Scatter Chart for 2 BHK and 3 BHK properties.

```
In [194]: 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'] = (8,6)
    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')
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price (Lakh Indian Rupees)")
    plt.title(location)
    plt.legend()

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



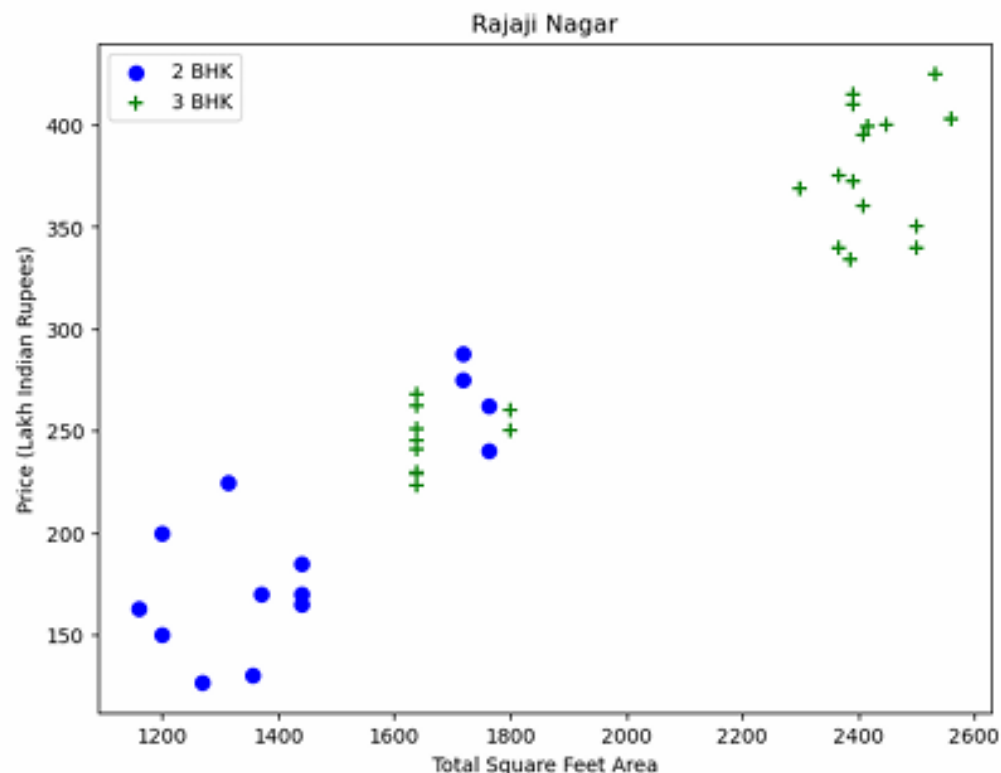
remove those 2 BHK apartments whose price_per_sqft is less than mean price_per_sqft of 1 BHK apartment

```
In [195]: def remove_bhk_outliers(df):
           exclude_indices = np.array([])
           for location, location_df in df.groupby('location'):
               bhk_stats = {}
               for bhk, bhk_df in location_df.groupby('BHK'):
                   bhk_stats[bhk] = {
                       'mean': np.mean(bhk_df.price_per_sqft),
                       'std': np.std(bhk_df.price_per_sqft),
                       'count': bhk_df.shape[0]
                   }
               for bhk, bhk_df in location_df.groupby('BHK'):
                   stats = bhk_stats.get(bhk-1)
                   if stats and stats['count']>5:
                       exclude_indices = np.append(exclude_indices, bhk_df[bhk_df.
               return df.drop(exclude_indices,axis='index')
df = remove_bhk_outliers(df)
df.shape
```

Out[195]: (7329, 7)

Plot same scatter chart again to visualize price_per_sqft for 2 BHK and 3 BHK properties

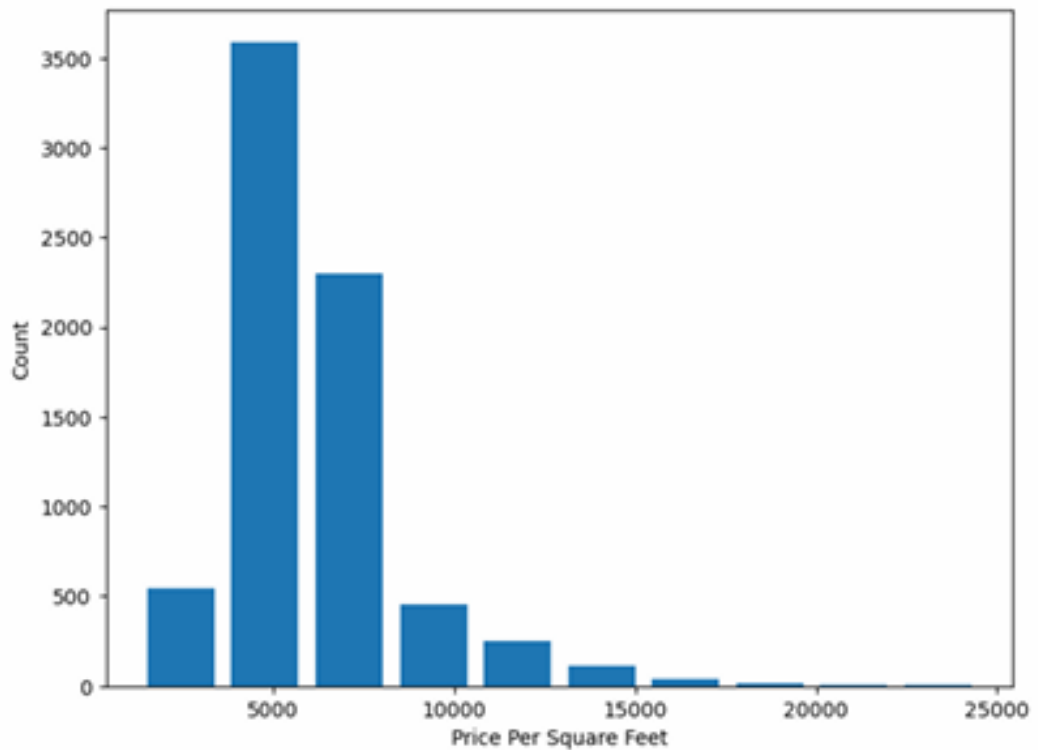
```
In [196]: plot_scatter_chart(df, "Rajaji Nagar")
```



Ploting the histogram for Price Per Square Feet vs Count.

```
In [197]: plt.hist(df.price_per_sqft,rwidth=0.8)
plt.xlabel("Price Per Square Feet")
plt.ylabel("Count")
```

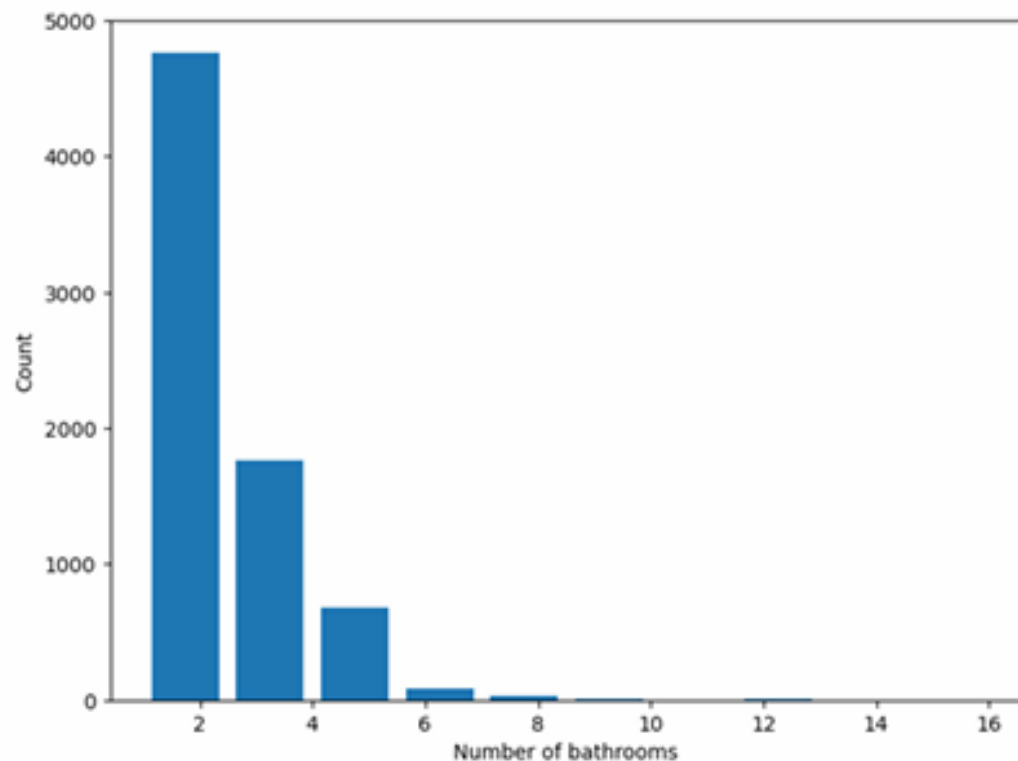
```
Out[197]: Text(0, 0.5, 'Count')
```



Plotting the histogram for Number of bathrooms vs Count.

```
In [198]: plt.hist(df.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
```

```
Out[198]: Text(0, 0.5, 'Count')
```



```
In [199]: df[df.bath>10]
```

```
Out[199]:
```

	location	size	total_sqft	bath	price	BHK	price_per_sqft
5277	Neeladri Nagar	10 BHK	4000.0	12.0	160.0	10	4000.000000
8486	other	10 BHK	12000.0	12.0	525.0	10	4375.000000
8575	other	16 BHK	10000.0	16.0	550.0	16	5500.000000
9308	other	11 BHK	6000.0	12.0	150.0	11	2500.000000
9639	other	13 BHK	5425.0	13.0	275.0	13	5069.124424

It is unusual to have 2 more bathrooms than number of bedrooms in a home. So we are discarding that also.

```
In [200]: df = df[df.bath<df.BHK+2]
```

```
In [201]: df.head()
```

```
Out[201]:
```

	location	size	total_sqft	bath	price	BHK	price_per_sqft
0	1st Block Jayanagar	4 BHK	2850.0	4.0	428.0	4	15017.543860
1	1st Block Jayanagar	3 BHK	1630.0	3.0	194.0	3	11901.840491
2	1st Block Jayanagar	3 BHK	1875.0	2.0	235.0	3	12533.333333
3	1st Block Jayanagar	3 BHK	1200.0	2.0	130.0	3	10833.333333
4	1st Block Jayanagar	2 BHK	1235.0	2.0	148.0	2	11983.805668

Using Label Encoding for Location.

```
In [202]: from sklearn.preprocessing import LabelEncoder  
encoder = LabelEncoder()  
encoded_col = encoder.fit_transform(df[["location"]])  
df["encoded_loc"] = encoded_col
```

In [203]: `encoder.classes_`

```
Out[203]: array(['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',
                'AECS Layout', 'Abbigere', 'Akshaya Nagar', 'Ambalipura',
                'Ambedkar Nagar', 'Amruthahalli', 'Anandapura', 'Ananth Nagar',
                'Anekal', 'Anjanapura', 'Ardendale', 'Arekere', 'Attibele',
                'BEML Layout', 'BTM 2nd Stage', 'BTM Layout', 'Babusapalaya',
                'Badavala Nagar', 'Balagere', 'Banashankari',
                'Banashankari Stage II', 'Banashankari Stage III',
                'Banashankari Stage V', 'Banashankari Stage VI', 'Banaswadi',
                'Banjara Layout', 'Bannerghatta', 'Bannerghatta Road',
                'Basavangudi', 'Basaveshwara Nagar', 'Battarahalli', 'Begur',
                'Begur Road', 'Bellandur', 'Benson Town', 'Bharathi Nagar',
                'Bhoganhalli', 'Billekahalli', 'Binny Pete', 'Bisuvanahalli',
                'Bommanahalli', 'Bommasandra', 'Bommasandra Industrial Area',
                'Bommenahalli', 'Brookefield', 'Budigere', 'CV Raman Nagar',
                'Chamrajpet', 'Chandapura', 'Channasandra', 'Chikka Tirupathi',
                'Chikkabanavar', 'Chikkalasandra', 'Choodasandra', 'Cooke Town',
                'Cox Town', 'Cunningham Road', 'Dasanapura', 'Dasarahalli',
                'Devanahalli', 'Devarachikkanahalli', 'Dodda Nekkundi',
                'Doddaballapur', 'Doddakallasandra', 'Doddathoguru', 'Domlur',
                'Dommasandra', 'EPIP Zone', 'Electronic City',
                'Electronic City Phase II', 'Electronics City Phase 1',
                'Frazer Town', 'GM Palaya', 'Garudachar Palya', 'Giri Nagar',
                'Gollarapalya Hosahalli', 'Gottigere', 'Green Glen Layout',
                'Gubbalala', 'Gunjur', 'HAL 2nd Stage', 'HBR Layout',
                'HRBR Layout', 'HSR Layout', 'Haralur Road', 'Harlur', 'Hebbal',
                'Hebbal Kempapura', 'Hegde Nagar', 'Hennur', 'Hennur Road',
                'Hoodi', 'Horamavu Agara', 'Horamavu Banaswadi', 'Hormavu',
                'Hosa Road', 'Hosakerehalli', 'Hoskote', 'Hosur Road', 'Hulimavu',
                'ISRO Layout', 'ITPL', 'Iblur Village', 'Indira Nagar', 'JP Nagar',
                'Jakkur', 'Jalahalli', 'Jalahalli East', 'Jigani',
                'Judicial Layout', 'KR Puram', 'Kadubeesanahalli', 'Kadugodi',
                'Kaggadasapura', 'Kaggalipura', 'Kaikondrahalli',
                'Kalena Agrahara', 'Kalyan nagar', 'Kambipura', 'Kammanahalli',
                'Kammasandra', 'Kanakapura', 'Kanakpura Road', 'Kannamangala',
                'Karuna Nagar', 'Kasavanahalli', 'Kasturi Nagar', 'Kathriguppe',
                'Kaval Byrasandra', 'Kenchenahalli', 'Kengeri',
                'Kengeri Satellite Town', 'Kereguddadahalli', 'Kodichikkanahalli',
                'Kodigehaali', 'Kodigehalli', 'Kodihalli', 'Kogilu', 'Konanakunte',
                'Koramangala', 'Kothannur', 'Kothanur', 'Kudlu', 'Kudlu Gate',
                'Kumaraswami Layout', 'Kundalahalli', 'LB Shastri Nagar',
                'Laggere', 'Lakshminarayana Pura', 'Lingadheeranahalli',
                'Magadi Road', 'Mahadevpura', 'Mahalakshmi Layout', 'Mallasandra',
                'Malleshpalya', 'Malleshwaram', 'Marathahalli', 'Margondanahalli',
                'Marsur', 'Mico Layout', 'Munnekollal', 'Murugeshpalya',
                'Mysore Road', 'NGR Layout', 'NRI Layout', 'Nagarbhavi',
                'Nagasandra', 'Nagavara', 'Nagavarapalya', 'Narayanapura',
                'Neeladri Nagar', 'Nehru Nagar', 'OMBR Layout', 'Old Airport Road',
                'Old Madras Road', 'Padmanabhanagar', 'Pai Layout', 'Panathur',
                'Parappana Agrahara', 'Pattandur Agrahara', 'Poorna Pragna Layout',
                'Prithvi Layout', 'R.T. Nagar', 'Rachenahalli',
                'Raja Rajeshwari Nagar', 'Rajaji Nagar', 'Rajiv Nagar',
                'Ramagondanahalli', 'Ramamurthy Nagar', 'Rayasandra',
                'Sahakara Nagar', 'Sanjay nagar', 'Sarakki Nagar', 'Sarjapur',
                'Sarjapur Road', 'Sarjapura - Attibele Road',
                'Sector 2 HSR Layout', 'Sector 7 HSR Layout', 'Seegehalli',
                'Shampura', 'Shivaji Nagar', 'Singasandra', 'Somasundara Palya',
                'Sompura', 'Sonnenahalli', 'Subramanyapura', 'Sultan Palaya',
                'TC Palaya', 'Talaghattapura', 'Thanisandra', 'Thigalarapalya',
```

```
In [204]: df.head()
```

```
Out[204]:
```

	location	size	total_sqft	bath	price	BHK	price_per_sqft	encoded_loc
0	1st Block Jayanagar	4 BHK	2850.0	4.0	428.0	4	15017.543860	0
1	1st Block Jayanagar	3 BHK	1630.0	3.0	194.0	3	11901.840491	0
2	1st Block Jayanagar	3 BHK	1875.0	2.0	235.0	3	12533.333333	0
3	1st Block Jayanagar	3 BHK	1200.0	2.0	130.0	3	10833.333333	0
4	1st Block Jayanagar	2 BHK	1235.0	2.0	148.0	2	11983.805668	0

Making dictionary for encoded location for cross checking

```
In [205]: location_dictionary = []
for i in encoder.classes_:
    location_dictionary.append([i,encoder.transform([[i]])[0]])
location_dictionary = pd.DataFrame(location_dictionary,columns=["location",
location_dictionary
```

```
Out[205]:
```

	location	encoded
0	1st Block Jayanagar	0
1	1st Phase JP Nagar	1
2	2nd Phase Judicial Layout	2
3	2nd Stage Nagarbhavi	3
4	5th Block Hbr Layout	4
...
237	Yelahanka	237
238	Yelahanka New Town	238
239	Yelenahalli	239
240	Yeshwanthpur	240
241	other	241

242 rows × 2 columns

8. Machine Learning Model Development,

When data scientists start designing machine learning (ML) models, their end goal is the deployment of the model. Deploying a model refers to placing an ML model into an environment where it can do the job it was created to do.

Deploying a model into production requires a proper machine learning deployment architecture involving various teams of ML professionals, software, and machinery. In this blog, we will delve into the action of model deployment and identify the challenges it currently faces.

8.1 Model Creation,

Once the data is in usable shape and you know the problem you're trying to solve, it's time to train the model to learn from the quality data by applying a range of techniques and algorithms. This phase requires selecting and applying model techniques and algorithms; setting and adjusting hyperparameters; training and validating the model; developing and testing ensemble models, if needed; and optimizing the model.

To accomplish all that, this stage often includes the following actions:

- Select the right algorithm for your learning objective and data requirements. For example, linear regression is a popular option for mapping correlations between two variables in a data set.
- Configure and tune hyperparameters for optimal performance and determine a method of iteration such as learning rate to attain the best hyperparameters.
- Identify features that provide the best results.
- Determine whether model explainability or interpretability is required.
- Develop ensemble models for improved performance.
- Compare the performance of different model versions.
- Identify requirements for the model's operation and deployment.

8.2 Model Evaluation and Optimization,

Evaluating a model's performance encompasses confusion matrix calculations, business KPIs, machine learning metrics, model quality measurements and a final determination of whether the model can meet the established business goals.

During the model evaluation process, perform the following assessments:

- Evaluate the model using a validation data set.
- Determine confusion matrix values for classification problems.
- Identify methods for K-fold cross-validation, if using that approach.
- Further tune hyperparameters for optimal performance.
- Compare the machine learning model to the baseline model or heuristic.

8.3 Model Deployment,

Model deployment is the process of implementing a fully functioning machine learning model into production where it can make predictions based on data. Users, developers, and systems then use these predictions to make practical business decisions. Models can be deployed in multiple ways, but they're usually integrated with apps through APIs so that all users can gain access to them. Model deployment is the fourth stage in the model development life cycle after planning, data preparation, and model development. However, this stage is usually the most cumbersome for data scientists as it takes time and resources. Any given model typically undergoes countless modifications until it is ready to be released into a production environment. Also, some models don't even make it to the deployment stage if they don't meet the desired objectives.

8.4 Challenges in Machine Learning Model Deployment

If a machine learning model isn't properly deployed into production, it cannot provide accurate predictions. In turn, professionals cannot utilize the information to implement efficient business strategies.

It is estimated that only 10% of all models eventually make it into production. This is because many companies aren't yet adequately equipped with machinery and software to support many models. At times, there can be a discrepancy between the model's programming language and the company's production system, making it impossible or too costly to find a solution.

Sometimes models don't see the light of day due to a lack of access to data and a disconnect between IT, data science, and engineering teams. Software developers, data scientists, as well as business professionals need to coordinate more transparently to address problems before they escalate and exchange valuable information for better model development and deployment.

8.5 Model Development Implementation

Splitting Data and Dropping some columns

```
In [206]: X = df.drop(['location', 'size', 'price_per_sqft', 'price'], axis='columns')
X.head()
```

```
Out[206]:
```

	total_sqft	bath	BHK	encoded_loc
0	2850.0	4.0	4	0
1	1630.0	3.0	3	0
2	1875.0	2.0	3	0
3	1200.0	2.0	3	0
4	1235.0	2.0	2	0

```
In [207]: y = df.price
y.head()
```

```
Out[207]:
```

0	428.0
1	194.0
2	235.0
3	130.0
4	148.0

Name: price, dtype: float64

```
In [208]: X.shape
```

```
Out[208]: (7251, 4)
```

```
In [209]: y.shape
```

```
Out[209]: (7251,)
```

Machine Learning Model Development

```
In [210]: #importing neccessary Libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```



```
In [211]: # train and test data splitting
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.1,random_s
print('Shape of x_train is {}'.format(x_train.shape))
print('Shape of x_test is {}'.format(x_test.shape))
print('Shape of y_train is {}'.format(y_train.shape))
print('Shape of y_test is {}'.format(y_test.shape))
```

```
Shape of x_train is (6525, 4)
Shape of x_test is (726, 4)
Shape of y_train is (6525,)
Shape of y_test is (726,)
```

```
In [212]: #model training
def modelevaluation(x_train,y_train,x_test,y_test):
    models = {"LR":LinearRegression(),"DT":DecisionTreeRegressor(),
              "EN":ElasticNet()}
    for i in models.keys():
        regressor_model = models[i]
        regressor_model.fit(x_train,y_train)
        y_pred = regressor_model.predict(x_test)
        print('_'*50)
        print(models[i])
        print("Accuracy Score is:",r2_score(y_test,y_pred))
        print("MSE:",mean_squared_error(y_test,y_pred))
        print("MAE:",mean_absolute_error(y_test,y_pred))
```

```
In [213]: modelevaluation(x_train,y_train,x_test,y_test)
```

```
LinearRegression()
Accuracy Score is: 0.7276985403562471
MSE: 1733.9815095901956
MAE: 23.306535767308954
```

```
DecisionTreeRegressor()
Accuracy Score is: 0.582729602106449
MSE: 2657.125508593875
MAE: 22.586367445163994
```

```
ElasticNet()
Accuracy Score is: 0.7281269733891832
MSE: 1731.2533018230372
MAE: 23.281732852506536
```

Linear Regression is performing the best out of all, So we are developing model with it

```
In [214]: lr_model = LinearRegression()
lr_model.fit(x_train,y_train)
y_pred = lr_model.predict(x_test)
print("Accuracy Score is:",r2_score(y_test,y_pred)*100)
```

```
Accuracy Score is: 72.76985403562472
```

Exporting models using Pickle module

In [215]: lr_model

Out[215]: LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [216]: encoder

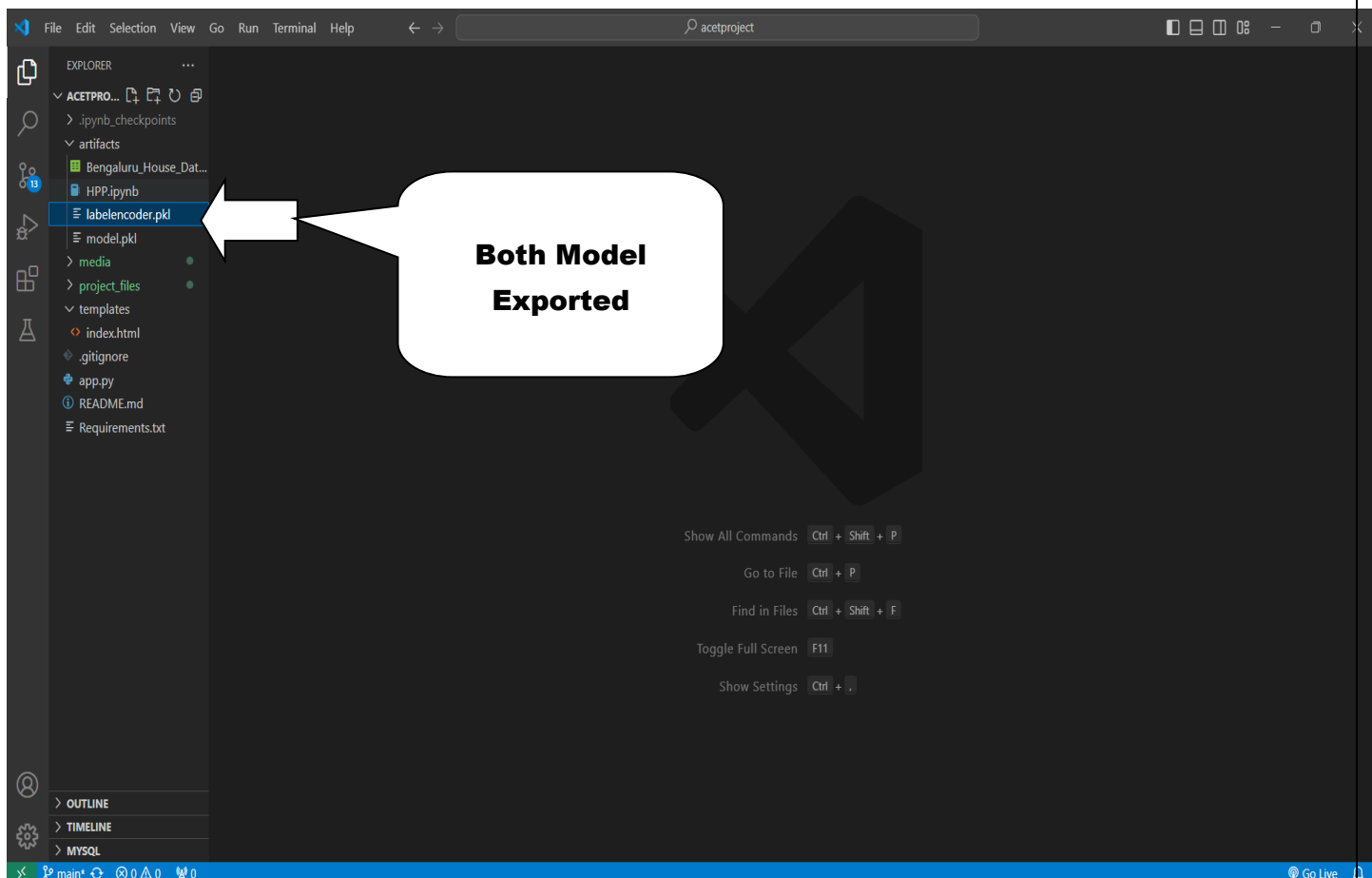
Out[216]: LabelEncoder()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [217]: import pickle
pickle.dump(encoder, open("labelencoder.pkl", "wb"))
pickle.dump(lr_model, open("model.pkl", "wb"))
```

In []:



9. Web Development & Model Deployment Implementation

9.1 Frontend Development

Everything you see on a website, like buttons, links, animations, and more, were created by a front end web developer. It is the front end developer's job to take the vision and design concept from the client and implement it through code.

Everything on the page from the logo to the search bar, buttons, overall layout and how the user interacts with the page was created by a front end developer. Front end developers are in charge of the look and feel of the website.

Front end developers also have to make sure the website looks good on all devices (phones, tablets, and computer screens).

What Skills Do You Need to Become a Front End Developer?

The three main languages you need to know well are HTML, CSS, and JavaScript. From there you can focus on frameworks, libraries, and other useful tools.

HTML

HTML stands for HyperText Markup Language. HTML displays the content on the page like buttons, links, headings, paragraphs, and lists.

You should not use HTML for styling. That is what CSS is for.

CSS

CSS stands for Cascading Style Sheets. CSS is responsible for the style of your web page including colors, layouts, and animations.

Cascading Style Sheets (CSS) controls the presentation aspect of the site and allows your site to have its own unique look. It does this by maintaining style sheets that sit on top of other style rules and are triggered based on other inputs, such as device screen size and resolution. The CSS can be added externally, internally, or embedded in the HTML tags.

JavaScript

JavaScript allows users to interact with the web page. Examples of JavaScript can be found in virtually any web page. JavaScript is an event-based imperative programming language (as opposed to HTML's declarative language model) that is used to transform a static HTML page into a dynamic interface. JavaScript code can use the Document Object Model (DOM), provided by the HTML standard, to manipulate a web page in response to events, like user input.

Using a technique called AJAX, JavaScript code can also actively retrieve content from the web (independent of the original HTML page retrieval), and also react to server-side events as well, adding a truly dynamic nature to the web page experience.

9.1.1. Web Design Vs Web Development

Web design is the art of planning and arranging content on a website so that it can be shared and accessed online with the world. A combination of aesthetic and functional elements, web design is a type of digital design that determines the look of a website—such as its colors, fonts, graphics and user interface (see our guide on website design best practices).

Today, creating a website is one of the pillars of having an online presence. Because of this, the world of web design is as dynamic as ever. It is constantly evolving, including mobile apps and user interface design, to meet the growing needs of website owners and visitors alike.

Web design is often a collaborative process that combines knowledge and tools from related industries, ranging from web design statistics to SEO optimization and UX. Web designers will often bring together professionals from these areas who can optimize performance and focus on the larger process and outcome.

The first step in our web design journey is to clarify the difference between web design and website development, since the two are closely related and often (mistakenly) used interchangeably:

- **Web design** refers to the visual design and experiential aspects of a particular website. We're going to dive into more detail about web design throughout the rest of this article.
- **Website development** refers to the building and maintenance of a website's structure, and involves intricate coding systems that ensure the website functions properly.

9.1.2. HTML & CSS Implementation

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>HPP | Home Page</title>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css"
integrity="sha512-
SnH5WK+bZxgPHs44uWIX+LLJAJ9/2PkPKZ5QiAj6Ta86w+fsb2TkcmfRyVX3pBnMFCv7oQPJkl9
QevSCWr3W6A==" crossorigin="anonymous" referrerpolicy="no-referrer" />
<style>
body {
font-family: Arial, sans-serif;
background: linear-gradient(135deg,#9AC8CD,#7b2ed8);
margin: 0;
padding: 0;}
.property-listing {
max-width: 600px;
margin: 100px auto;
padding: 30px;
background-color: #5d7ceb;
border-radius: 10px;
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
.property-listing:hover{
background-color: #6685f3;
}
.property-listing h1{
text-align: center;
}
.property-info {
margin-bottom: 15px;
}
label {
display: block;
font-weight: bold;
}
input[type="text"],
input[type="number"],
```

```
select {
width: 100%;
padding: 10px;
border: 1px solid #ccc;
border-radius: 3px;
font-size: 14px;
}

.submit-button {
background-color: #305af5;
color: #fff;
border: none;
border-radius: 3px;
padding: 10px 20px;
cursor: pointer;
font-size: 20px;
height: 100%;
width: 100%;
}

.submit-button:hover {
background-color: #023f80;
}

header{
display: flex;
position: fixed;
top:0;
left: 0;
width: 100%;
padding: 20px 12%;
background-color:#2e53d8 ;
display: flex;
}

.logo{
font-size: 40px;
font-weight: bold;
color: azure;
}

.logo:hover{
color:#1C1678;
}

.project_heading{
font-size: 30px;
font-family:'Times New Roman';
```

```

font-weight: bold;
left: 10px;
padding: 10px;
text-align: center;
padding-left: 300px;
}
.footer{
text-align: center;
font-size: 15px;
font-weight: bold;
display: block;
background: linear-gradient(100deg,#9AC8CD,#7b2ed8);
}
input{
width: 100%;
}
.prediction{
text-align: center;
background-color: yellow;
padding: 5px;
margin-top: 10px;
}
</style>
</head>
<body>
<header>
<div class="logo">
HPP
</div>
<div class=" project_heading">
HOUSE PRICE PREDICTION
</div>
</header>
<div class="property-listing">
<h1>Property Details</h1>
<form action="{ { url_for('index') } }" method="POST">
<div class="property-info">
<label for="bhk">BHK:</label>
<select id="bhk" name="bhk">
<option value="1">1 BHK</option>
<option value="2">2 BHK</option>
<option value="3">3 BHK</option>

```

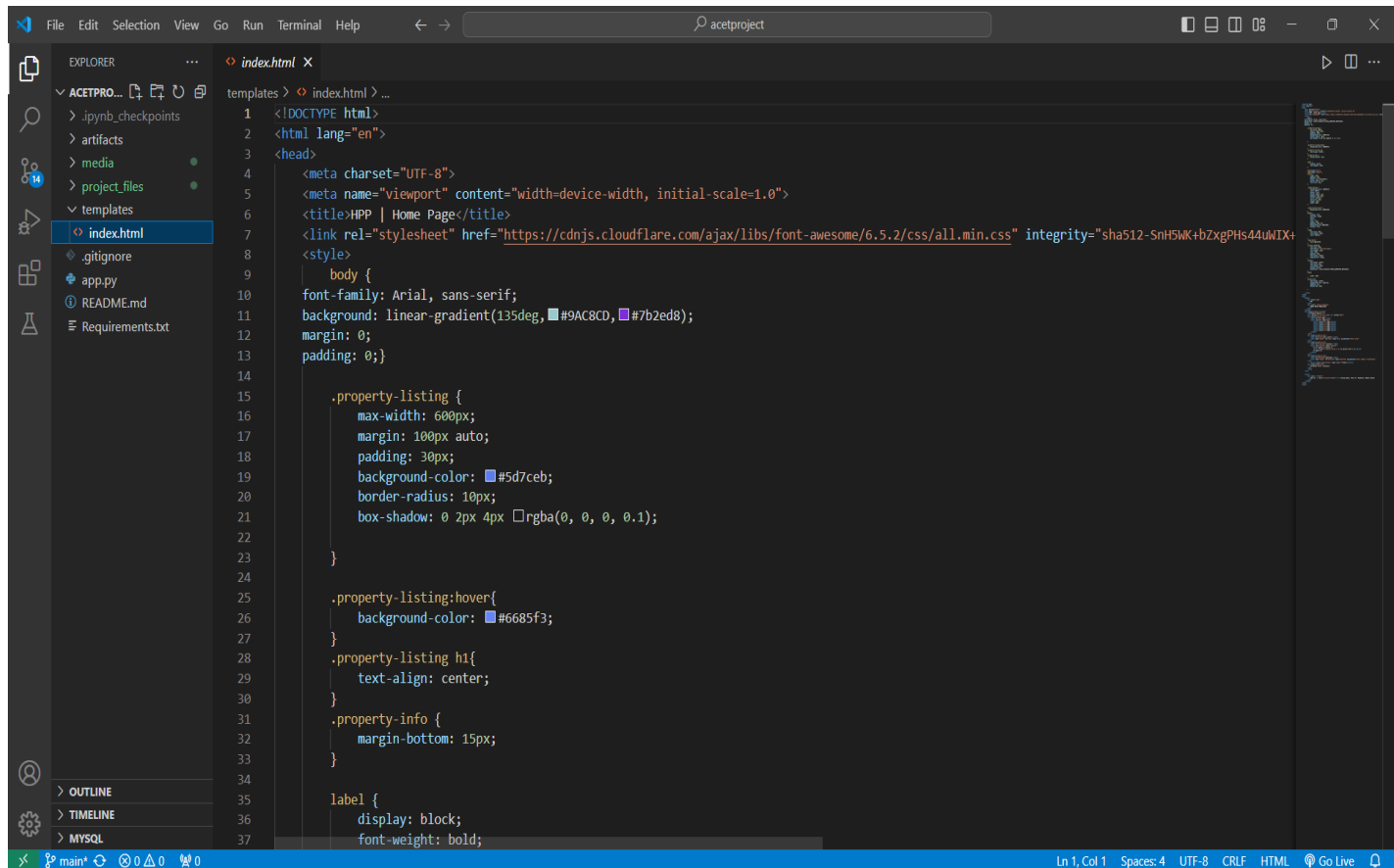


```

<option value="4">4 BHK</option>
<option value="5">5 BHK</option>
<option value="6">6 BHK</option>
</select>
</div>
<div class="property-info">
<label for="area">Area (sq/ft):</label>
<input type="number" id="area" name="area" placeholder="Enter area">
</div>
<div class="property-info">
<label for="location">Location:</label>
<select id="location" name="location">
{ % for option in options % }
<option value="{ { option['value'] } }">{ { option['text'] } }</option>
{ % endfor % }
</select>
</div>
<div class="property-info">
<label for="bathroom">Bathroom:</label>
<input type="number" id="bathroom" name="bathroom" placeholder="Enter number of bathrooms">
</div>
<button class="submit-button" type="submit">Predict</button>
<div class="prediction">
Estimated Price: { { output } }
</div>
</form>
</div>
<footer>
<div class = "footer">
made by <i class="fa-solid fa-heart"></i> Anurag Gupta, Arpit Kr. Nauhwara, Hemant Sharma
</div>
</footer>
</body>
</html>

```

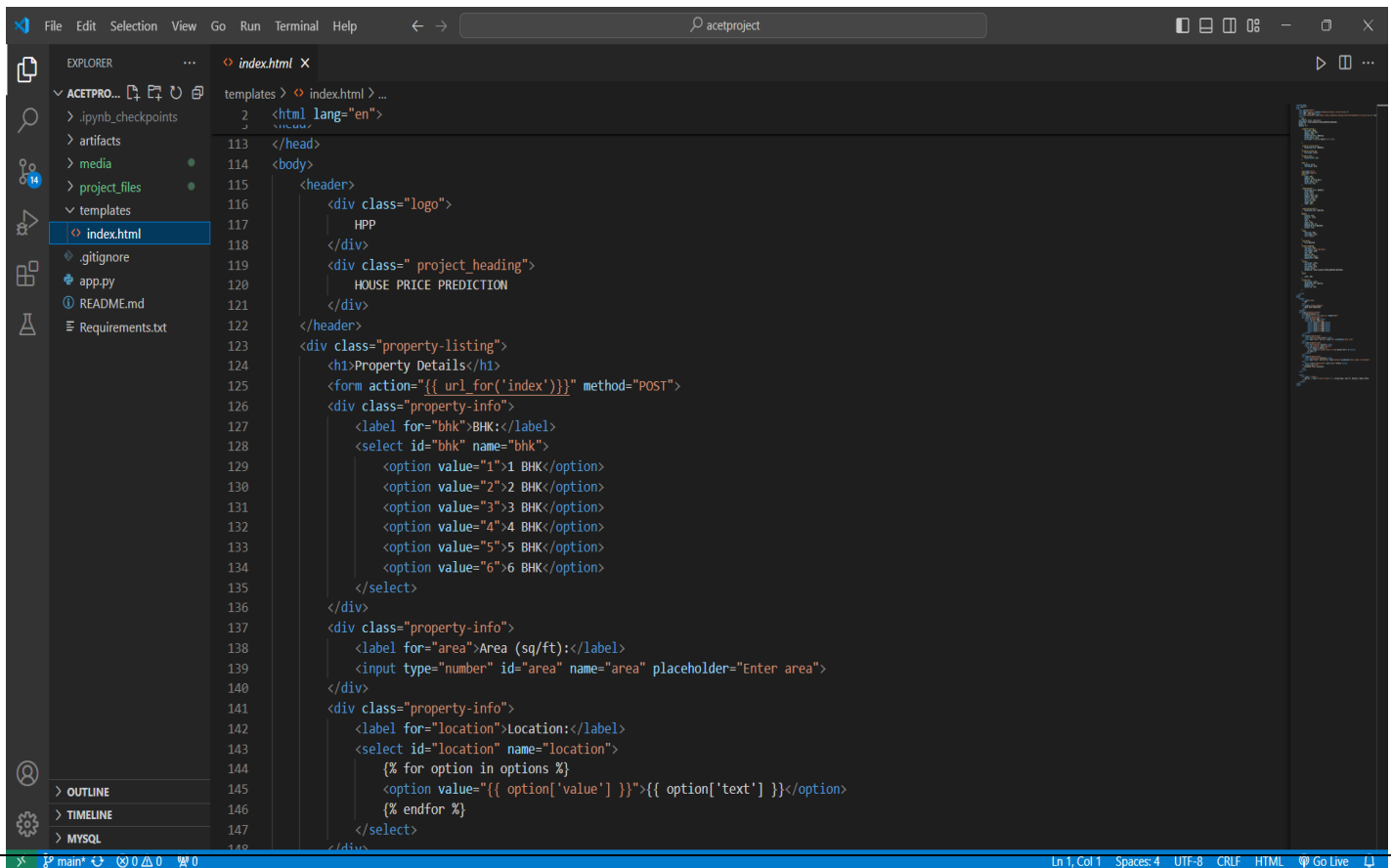
HTML & CSS Snapshots



This screenshot shows the CSS portion of the `index.html` file in VS Code. The Explorer sidebar on the left shows the project structure with `index.html` selected. The main editor displays the following CSS code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>HPP | Home Page</title>
7   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css" integrity="sha512-SnH5MK+bZxgPHS44uWIX+">
8   <style>
9     body {
10      font-family: Arial, sans-serif;
11      background: linear-gradient(135deg, #9AC8CD, #7b2ed8);
12      margin: 0;
13      padding: 0;
14    }
15    .property-listing {
16      max-width: 600px;
17      margin: 100px auto;
18      padding: 30px;
19      background-color: #5d7ceb;
20      border-radius: 10px;
21      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
22    }
23    .property-listing:hover {
24      background-color: #6685f3;
25    }
26    .property-listing h1 {
27      text-align: center;
28    }
29    .property-info {
30      margin-bottom: 15px;
31    }
32    label {
33      display: block;
34      font-weight: bold;
35    }
```

The status bar at the bottom indicates the cursor is at line 1, column 1, with 4 spaces, in UTF-8 encoding, CRLF line endings, and HTML mode.



This screenshot shows the HTML portion of the `index.html` file in VS Code. The Explorer sidebar on the left shows the project structure with `index.html` selected. The main editor displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>HPP | Home Page</title>
7   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css" integrity="sha512-SnH5MK+bZxgPHS44uWIX+">
8   <style>
9     body {
10      font-family: Arial, sans-serif;
11      background: linear-gradient(135deg, #9AC8CD, #7b2ed8);
12      margin: 0;
13      padding: 0;
14    }
15    .property-listing {
16      max-width: 600px;
17      margin: 100px auto;
18      padding: 30px;
19      background-color: #5d7ceb;
20      border-radius: 10px;
21      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
22    }
23    .property-listing:hover {
24      background-color: #6685f3;
25    }
26    .property-listing h1 {
27      text-align: center;
28    }
29    .property-info {
30      margin-bottom: 15px;
31    }
32    label {
33      display: block;
34      font-weight: bold;
35    }
36  </style>
37 </head>
38 <body>
39   <header>
40     <div class="logo">
41       HPP
42     </div>
43     <div class="project_heading">
44       HOUSE PRICE PREDICTION
45     </div>
46   </header>
47   <div class="property-listing">
48     <h1>Property Details</h1>
49     <form action="{ url_for('index') }}" method="POST">
50       <div class="property-info">
51         <label for="bhk">BHK</label>
52         <select id="bhk" name="bhk">
53           <option value="1">1 BHK</option>
54           <option value="2">2 BHK</option>
55           <option value="3">3 BHK</option>
56           <option value="4">4 BHK</option>
57           <option value="5">5 BHK</option>
58           <option value="6">6 BHK</option>
59         </select>
60       </div>
61       <div class="property-info">
62         <label for="area">Area (sq/ft)</label>
63         <input type="number" id="area" name="area" placeholder="Enter area">
64       </div>
65       <div class="property-info">
66         <label for="location">Location</label>
67         <select id="location" name="location">
68           {% for option in options %}
69             <option value="{ option['value'] }">{{ option['text'] }}</option>
70           {% endfor %}
71         </select>
72       </div>
73     </form>
74   </div>
```

The status bar at the bottom indicates the cursor is at line 1, column 1, with 4 spaces, in UTF-8 encoding, CRLF line endings, and HTML mode.

9.2. Backend Development

Generally, a website consists of two parts — the front end and the back end. The front end, also known as the client-side, is what you see in the browser. The back end, or server-side, is everything that happens “under the hood,” and its components aren’t immediately obvious.

Think of a website as a restaurant. When you first sit down, you’re presented with a menu, which may include pictures and descriptions of the items you can order. When you place your order, you might request something specific, like a salad with dressing on the side. This represents the front end.

Then, the kitchen staff takes your order, gets the ingredients from the refrigerator and pantry, cooks them together, and brings you your food. This represents the back end.

Below, we’ll take a closer look at the back end, what Back-End Developers do, the tools they use, and how to become one. (Or if you’d rather jump into the back-end yourself, check out our courses on web development.)

What goes into the back end?

The back end is a combination of servers and databases. Servers control how users access files. Databases are organized and structured collections of data.

Here’s an example: When you log into a website and enter your username or email and password. This information is sent to the server-side software which validates the structure of your email and password. If everything looks good, it checks the data with the database to ensure someone with that username and password exists. If it does, the database will log you in and sends information back to you in the form of your user page.

What do Back-End Developers do?

In another post, Doug, one of our Senior Back-End Engineers, gives us an overview of what a Back-End Developer does. While touching on the role’s capacity for problem-solving, he provides a list of common responsibilities, including:

- Creating, integrating, and managing databases.
- Using back-end frameworks to build server-side software.

- Validating data to make sure it's formatted correctly before being sent to the database.
- Integrating user-facing elements with server-side elements to make sure that information is being sent to the right place so the server can retrieve it.

What tools do Back-End Developers use?

Back-End Developers use a range of technology and software, many of which fall into three categories: databases, programming languages, and frameworks.

Databases

As we explained above, databases are used to store user information and other important data. Popular database management systems include:

- MySQL
- MongoDB
- Oracle
- PostgreSQL

Programming languages

Back-End Developers use query languages like SQL to manipulate the data stored in databases. They also use various programming languages to build applications that facilitate communication between servers and databases. Some of these languages include:

- Ruby
- Python
- PHP
- Node.js
- Java

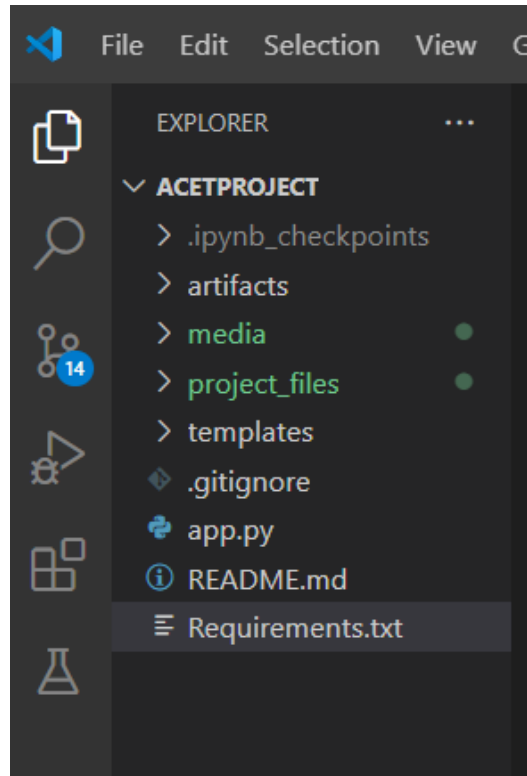
Python is a great choice for beginners. It's concise and easy to read. It's also extremely popular and has a large programming community behind it. Ruby is another beginner-friendly language that has an enthusiastic programming community behind it.

Frameworks

Frameworks make software development faster and easier, saving time that developers would otherwise spend writing code. Popular frameworks include:

- Ruby on Rails
- Django and Flask
- Express.js

9.2.1. Requirements & Project Structure Setup



Project Structure:

a. Artifacts Folder: In the realm of data science projects, the "artifacts" folder serves as a crucial repository for key components essential to the project's lifecycle. It acts as a centralized location housing datasets, models, and Jupyter notebooks, among other relevant artifacts. Let's delve into the significance of each component within this folder:

Datasets: At the heart of any data science endeavor lies the data itself. Datasets within the artifacts folder encompass the raw or pre-processed data used for analysis, modeling, and evaluation. These datasets may include structured data in formats like CSV, Excel, or SQL databases, or unstructured data such as text, images, or videos. Organizing datasets within the artifacts folder ensures easy accessibility and reproducibility of analyses.

Models: The models directory contains trained machine learning or statistical models generated during the project's lifecycle. These models encapsulate the learned patterns and relationships within the data, enabling predictive or descriptive analytics. Examples of model formats include serialized objects (e.g., Pickle files), TensorFlow SavedModels, or serialized scikit-learn models. Storing

models within the artifacts folder facilitates version control, model sharing, and deployment across various environments.

Jupyter Notebooks: Jupyter notebooks are interactive computing documents that blend code, visualizations, and explanatory text within a single interface. They play a pivotal role in data exploration, experimentation, and documentation. Placing Jupyter notebooks in the artifacts folder ensures transparency and reproducibility of analyses, allowing team members or stakeholders to follow the project's workflow step-by-step. Moreover, Jupyter notebooks serve as valuable documentation, providing insights into data preprocessing, model training, evaluation metrics, and insights derived from the analysis.

Beyond these primary components, the artifacts folder may also include supplementary files such as configuration files, scripts for data preprocessing or model evaluation, visualization outputs, and documentation detailing project methodologies, assumptions, and findings. Adopting a standardized structure for the artifacts folder fosters collaboration, streamlines project management, and enhances the project's overall reproducibility and transparency.

In summary, the artifacts folder serves as a centralized hub housing critical components of a data science project, including datasets, models, and Jupyter notebooks. By organizing these artifacts within a unified directory, teams can streamline collaboration, ensure reproducibility, and facilitate knowledge sharing across the project's lifecycle.

b. Media Folder: In the context of a website project, the "media" folder serves as a repository for images and videos utilized in various aspects of the site's design, functionality, and content. Here's a breakdown of what you might find in such a folder:

Images:

Content Visuals: Images used within the website's pages, such as product photos, blog post headers, or background images.

Icons and Logos: Graphic elements like icons, logos, and buttons used for navigation, branding, or calls to action.

User Interface Elements: Visual elements for user interface components like buttons, menus, sliders, or forms.

Responsive Design: Images optimized for different screen sizes and resolutions to ensure a consistent user experience across devices.

Favicons: Small icons displayed in the browser tab or bookmark bar to represent the website.

Videos:

Background Videos: Full-screen or section-specific videos used as background elements to enhance visual appeal or convey messages.

Product Demos: Video demonstrations showcasing product features, benefits, or usage instructions.

Testimonials: Video testimonials from customers or clients, providing social proof and credibility.

Tutorials or How-tos: Instructional videos guiding users through specific tasks or processes related to the website's content or functionality.

By organizing images and videos within the media folder, web developers and designers can efficiently manage assets, streamline website development, and ensure consistency in visual elements throughout the site. Additionally, this structured approach facilitates version control and collaboration among team members working on different aspects of the website project.

c. Templates Folder: In Flask, the "templates" folder holds HTML templates that define the structure and layout of web pages rendered by the application. Here's a brief overview of the significance of the templates folder in Flask:

HTML Templates: Flask follows the Model-View-Controller (MVC) pattern, where templates serve as the "View" component. HTML templates contain the presentation logic of the web application, defining how data is displayed to users in their browsers.

Dynamic Content: Templates in Flask support the insertion of dynamic content using Jinja2, a powerful templating engine. With Jinja2, developers can embed Python code within HTML templates to generate dynamic content, such as displaying database query results, user inputs, or session data.

Template Inheritance: Flask templates support inheritance, allowing developers to create base templates with common elements like headers, footers, and navigation bars, which can be extended by other templates. This promotes code reusability and maintainability by reducing redundancy in template code.

Overall, the templates folder in Flask plays a pivotal role in defining the user interface of web applications, enabling the creation of dynamic and interactive web pages.

d. .gitignore File: The .gitignore file serves as a crucial tool in Git repositories, instructing Git on which files and directories to ignore during version control operations. Its primary function is to maintain a clean and focused repository by excluding files that are irrelevant to the project or should not be tracked.

Common entries in a .gitignore file include temporary files generated by the operating system or text editors (e.g., .DS_Store on macOS, Thumbs.db on Windows), compiled binaries and executables (e.g., *.exe, *.o, *.class), dependency directories (e.g., node_modules/, venv/), and sensitive configuration files (e.g., .env, config.ini).

The structure of a .gitignore file is based on file patterns and directory names, supporting wildcards and special characters to match multiple files or directories. Each entry represents a file pattern or directory name that Git should ignore.

For instance, a .gitignore file contain entries like:

```
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
pip-wheel-metadata/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST
```

```
# PyInstaller
# Usually these files are written by a python script from a template
# before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.nox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
*.py,cover
.hypothesis/
.pytest_cache/

# Translations
*.mo
*.pot

# Django stuff:
*.log
local_settings.py
db.sqlite3
db.sqlite3-journal

# Flask stuff:
instance/
.webassets-cache

# Scrapy stuff:
.scrapy

# Sphinx documentation
docs/_build/

# PyBuilder
target/

# Jupyter Notebook
.ipynb_checkpoints
```

```
# IPython
profile_default/
ipython_config.py

# pyenv
.python-version

# pipenv
# According to pypa/pipenv#598, it is recommended to include Pipfile.lock in version
control.
# However, in case of collaboration, if having platform-specific dependencies or
dependencies
# having no cross-platform support, pipenv may install dependencies that don't work,
or not
# install all needed dependencies.
#Pipfile.lock

# PEP 582; used by e.g. github.com/David-OConnor/pyflow
__pypackages__

# Celery stuff
celerybeat-schedule
celerybeat.pid

# SageMath parsed files
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json
```

```
# Pyre type checker
.pyre/
```

The .gitignore file typically resides at the root directory of the Git repository. Git applies the rules specified in this file recursively to all files and directories within the repository, except those explicitly added to the staging area.

By utilizing .gitignore effectively, developers ensure that the repository remains focused on versioning source code and essential project files. This practice improves collaboration, maintains a clean version control history, and prevents the inclusion of irrelevant or sensitive files in the repository. Overall, the .gitignore file plays a critical role in streamlining the management of Git repositories and enhancing the development workflow.

e. app.py File: The app.py file is a fundamental component in Flask web applications, serving as the primary entry point for defining the application's behavior and routes. Here's a breakdown of its key elements:

Imports: The file begins with import statements to bring in necessary modules and libraries. This typically includes Flask itself (from flask import Flask) and may include additional modules for database interaction, form handling, or authentication.

App Initialization: Within app.py, the Flask application object is created and configured. This involves instantiating the Flask class (app = Flask(__name__)) and setting up any necessary configurations, such as secret keys, database connections, or debug mode (app.config['DEBUG'] = True).

Route Definitions: The heart of app.py lies in defining routes, which map URL paths to Python functions. Routes are created using decorators (@app.route('/')) followed by a function definition that executes when the corresponding URL is accessed (def home():). These functions typically return responses to the client, which can include HTML content, JSON data, or redirects.

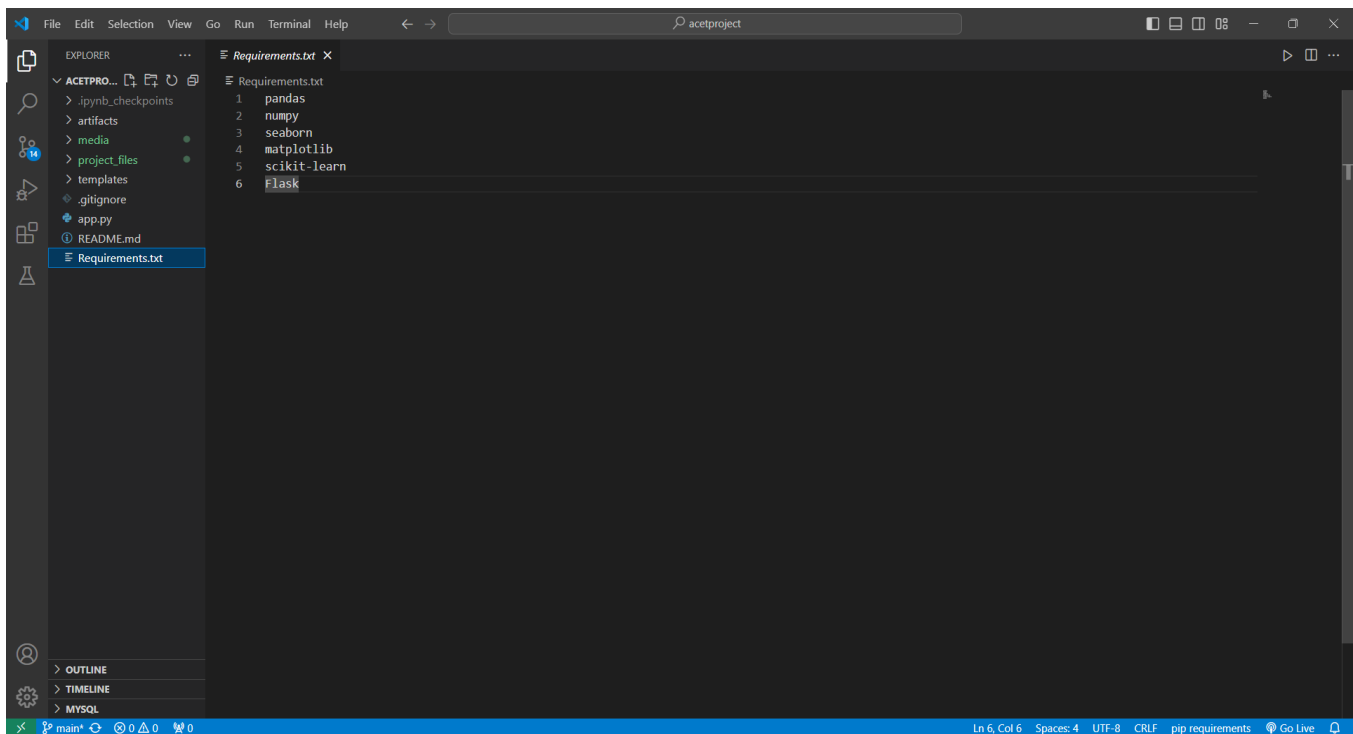
View Functions: Inside route definitions, view functions are defined to handle incoming requests and generate responses. These functions have access to request parameters (request.args, request.form) and can perform various operations such as data processing, database interaction, or rendering templates (render_template('index.html', data=data)).

Error Handling: app.py may include error handling logic to gracefully handle exceptions or HTTP errors. Flask provides mechanisms for registering error handlers (@app.errorhandler(404)) to customize error responses and improve the user experience.

Run the Application: Finally, app.py typically includes a block of code that runs the Flask application when the script is executed directly (if __name__ == '__main__': app.run()). This starts the development server and listens for incoming HTTP requests, allowing the application to be accessed via a web browser.

In summary, app.py is the backbone of a Flask application, housing route definitions, view functions, and configuration settings. It orchestrates the application's behavior, enabling developers to create dynamic and interactive web experiences.

Requirements.txt File:

A screenshot of a code editor window titled 'acetproject'. The left sidebar shows the 'EXPLORER' view with a file tree containing folders like '.ipynb_checkpoints', 'artifacts', 'media', 'project_files', 'templates', and files like '.gitignore', 'app.py', 'README.md', and 'Requirements.txt'. The 'Requirements.txt' file is selected and its contents are displayed in the main editor area. The file lists six Python packages with their versions: pandas, numpy, seaborn, matplotlib, scikit-learn, and Flask. The status bar at the bottom indicates 'Ln 6, Col 6', 'Spaces: 4', 'UTF-8', 'CRLF', and 'pip requirements'.

The requirements.txt file lists Python package dependencies and their versions for a project. It simplifies environment setup by allowing developers to install all dependencies with a single command (pip install -r requirements.txt). This file aids in version control, ensuring consistency across development environments. Developers specify exact versions or version ranges to manage compatibility and updates. It's crucial for maintaining a clean, reproducible environment and streamlining the setup process for Python projects.

9.2.2. Flask Implementation

```
9. from flask import Flask, render_template,request,url_for,redirect
10. import pickle
11. import numpy as np
12. import pandas as pd
13. from sklearn.preprocessing import LabelEncoder
14.
15. app = Flask(__name__)
16.
17. @app.route('/',methods = ["GET","POST"])
18. def index():
19.
20.     if request.method == "GET":
21.         encoder = pickle.load(open("artifacts/labelencoder.pkl","rb"))
22.         options = [{"value": i, "text": f"{i}"} for i in encoder.classes_]
23.         return render_template('index.html', options=options)
24.
25.     elif request.method == "POST":
26.         def encoding(loc):
27.             encoder = pickle.load(open("artifacts/labelencoder.pkl","rb"))
28.             return encoder.transform([loc])[0]
29.
30.         def get_dataframe(area,bath,bhk,encoded_location):
31.             df =
pd.DataFrame([{"total_sqft":area,"bath":bath,"BHK":bhk,"encoded_loc":encoded_location}
])
32.             return df
33.
34.         def predictor(area,bath,bhk,encoded_location):
35.             model = pickle.load(open("artifacts/model.pkl","rb"))
36.             df = get_dataframe(area,bath,bhk,encoded_location)
37.             return model.predict(df)[0]
38.
39.
40.
41.         areaip = request.form['area']
42.         bathip = float(request.form["bathroom"])
43.         bhkip = int(request.form["bhk"])
44.         location = request.form["location"]
45.         encoded_locationip = encoding(location)
46.
47.         prediction = str(int(predictor(areaip,bathip,bhkip,encoded_locationip)))
48.
49.         def beautify(prediction):
50.             if len(prediction)>=3:
51.                 return prediction[0]+ " Crores " + prediction[1]+prediction[2]+" Lakhs"
52.             else:
53.                 return prediction + " Lakhs"
54.
```

```

55.         return render_template("index.html",output=beautify(prediction))
56.
57.
58. if __name__ == '__main__':
59.     app.run(debug=True)

```

Flask App Snapshot

```

File Edit Selection View Go Run Terminal Help
acetproject

EXPLORER
ACETPROJECT
> .ipynb_checkpoints
> artifacts
> media
> project_files
> templates
> index.html
> .gitignore
app.py
README.md
Requirements.txt

app.py
1 from flask import Flask, render_template, request, url_for, redirect
2 import pickle
3 import numpy as np
4 import pandas as pd
5 from sklearn.preprocessing import LabelEncoder
6
7 app = Flask(__name__)
8
9 @app.route('/', methods = ["GET", "POST"])
10 def index():
11
12     if request.method == "GET":
13         encoder = pickle.load(open("artifacts/labelencoder.pkl", "rb"))
14         options = [{"value": i, "text": f"{i}"} for i in encoder.classes_]
15         return render_template('index.html', options=options)
16
17     elif request.method == "POST":
18         def encoding(loc):
19             encoder = pickle.load(open("artifacts/labelencoder.pkl", "rb"))
20             return encoder.transform([loc])[0]
21
22         def get_dataframe(area, bath, bhk, encoded_location):
23             df = pd.DataFrame([{"total_sqft": area, "bath": bath, "BHK": bhk, "encoded_loc": encoded_location}])
24             return df
25
26         def predictor(area, bath, bhk, encoded_location):
27             model = pickle.load(open("artifacts/model.pkl", "rb"))
28             df = get_dataframe(area, bath, bhk, encoded_location)
29             return model.predict(df)[0]
30
31
32
33         areaip = request.form["area"]
34         bathip = float(request.form["bathroom"])
35         bhkip = int(request.form["bhk"])
36         location = request.form["location"]
37         encoded_locationip = encoding(location)

```

Ln 22, Col 59 Spaces: 4 UTF-8 CRLF Python 3.10.4 64-bit Go Live

10. Website Preview & Website Tutorial

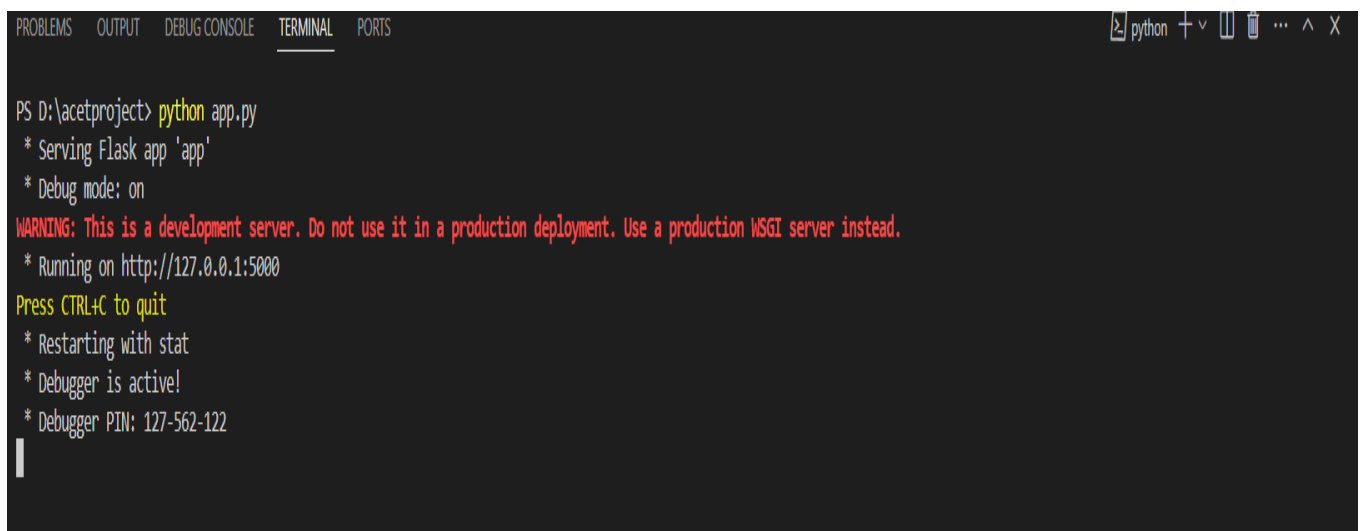
10.1. Commands to run Website:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\acetproject> python app.py
```

Ln 18, Col 7 Spaces: 4 UTF-8 CRLF Ignore Go Live

To execute a Python script named `app.py` via command line, use `python app.py` [arguments]. Replace [arguments] with any specific command-line inputs expected by the script. This standard command invokes the Python interpreter to run the script. For example, if your script requires a specific action, say "command", it would be like `python app.py command`. Ensure Python is installed and properly configured in your environment. This approach facilitates seamless execution of Python scripts, allowing for a wide range of functionalities from simple calculations to complex applications, all manageable through the command line interface.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\acetproject> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 127-562-122
```


10.2. Website Preview:

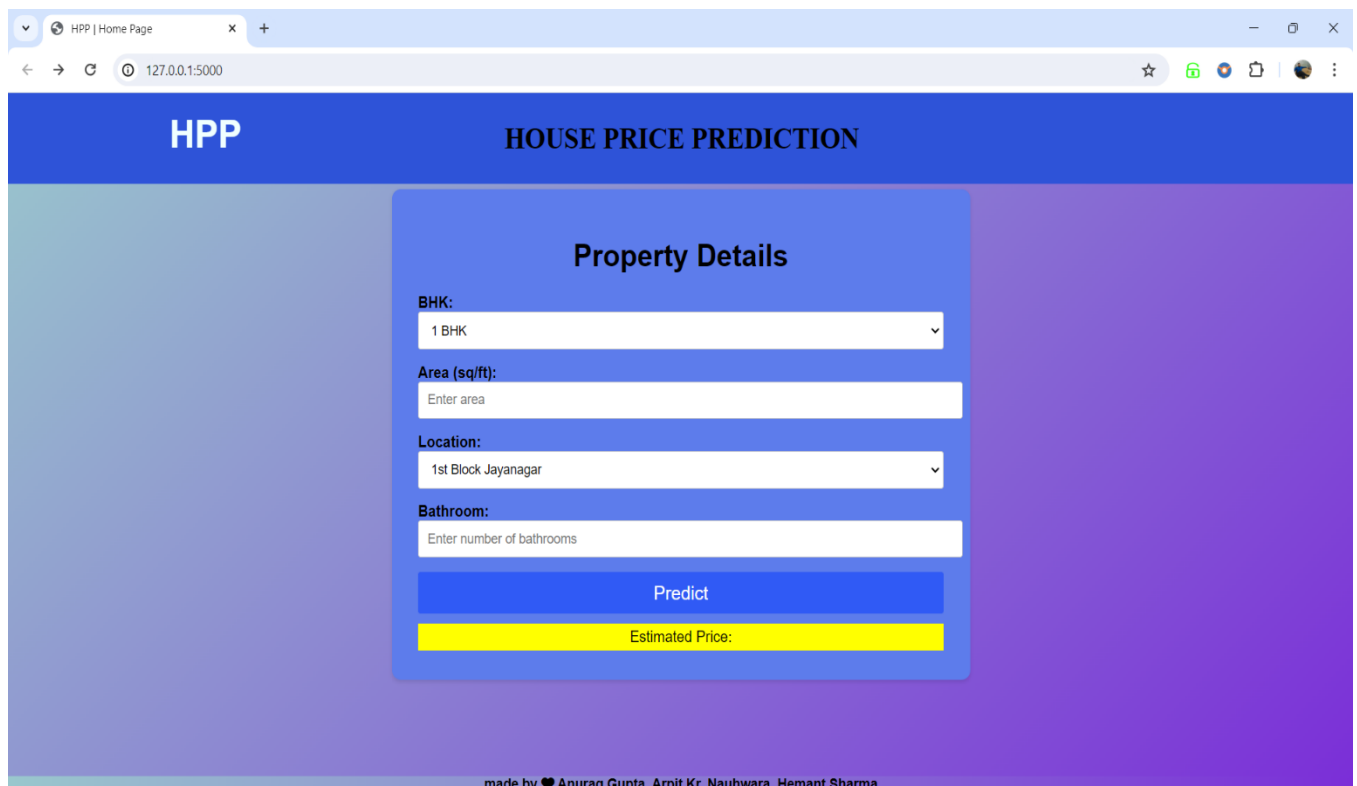
The initial impression of a website, often referred to as the "first look," is paramount in shaping users' perceptions and engagement. Upon landing on a website, users are greeted by its landing page, which serves as the digital storefront. A well-designed landing page captivates visitors with its visual appeal, utilizing vibrant colors, high-quality images, and clear branding elements to establish identity and evoke interest.

Content presentation plays a pivotal role in guiding users through the page. Concise headings and subheadings organize information effectively, while whitespace enhances readability and reduces clutter.

Responsiveness across various devices ensures a consistent experience for users, regardless of their preferred platform. Optimized loading speed minimizes wait times, enhancing user satisfaction and reducing bounce rates. Brand consistency, reflected in design elements, typography, and color schemes, reinforces brand identity and fosters recognition.

Overall, the first look at a website leaves a lasting impression, instilling confidence and curiosity in users. A well-executed landing page effectively communicates the brand's value proposition, setting the stage for a rewarding browsing experience and encouraging further exploration of the site's offerings.

Below is the first Look of our project:



The screenshot displays a web browser window with the title "HPP | Home Page" and the URL "127.0.0.1:5000". The website has a blue header with the text "HPP" and "HOUSE PRICE PREDICTION". The main content area features a "Property Details" form with the following fields:

- BHK:** A dropdown menu showing "1 BHK".
- Area (sq/ft):** A text input field with the placeholder "Enter area".
- Location:** A dropdown menu showing "1st Block Jayanagar".
- Bathroom:** A text input field with the placeholder "Enter number of bathrooms".

Below the form fields is a blue "Predict" button. Underneath the button is a yellow box labeled "Estimated Price:". At the bottom of the page, a footer reads "made by ❤️ Anurag Gupta, Arpit Kr. Nauhwara, Hemant Sharma".

10.3. Website Tutorial:

A website tutorial serves as a comprehensive guide for users, offering step-by-step instructions on navigating and utilizing the features of a website. Typically found in the form of written content, videos, or interactive demonstrations, website tutorials aim to enhance user understanding and proficiency in interacting with the site.

The tutorial typically begins with an overview of the website's purpose and main features, providing users with a clear understanding of what they can expect to find and accomplish. It then proceeds to walk users through the various sections of the website, highlighting key functionalities and explaining how to access and utilize them effectively.

Each step of the tutorial is accompanied by clear instructions, visuals, and examples to aid comprehension. Users are often encouraged to follow along interactively, practicing the actions demonstrated in real-time to reinforce learning.

Additionally, website tutorials may include troubleshooting tips and FAQs to address common issues or questions that users may encounter during their interaction with the site.

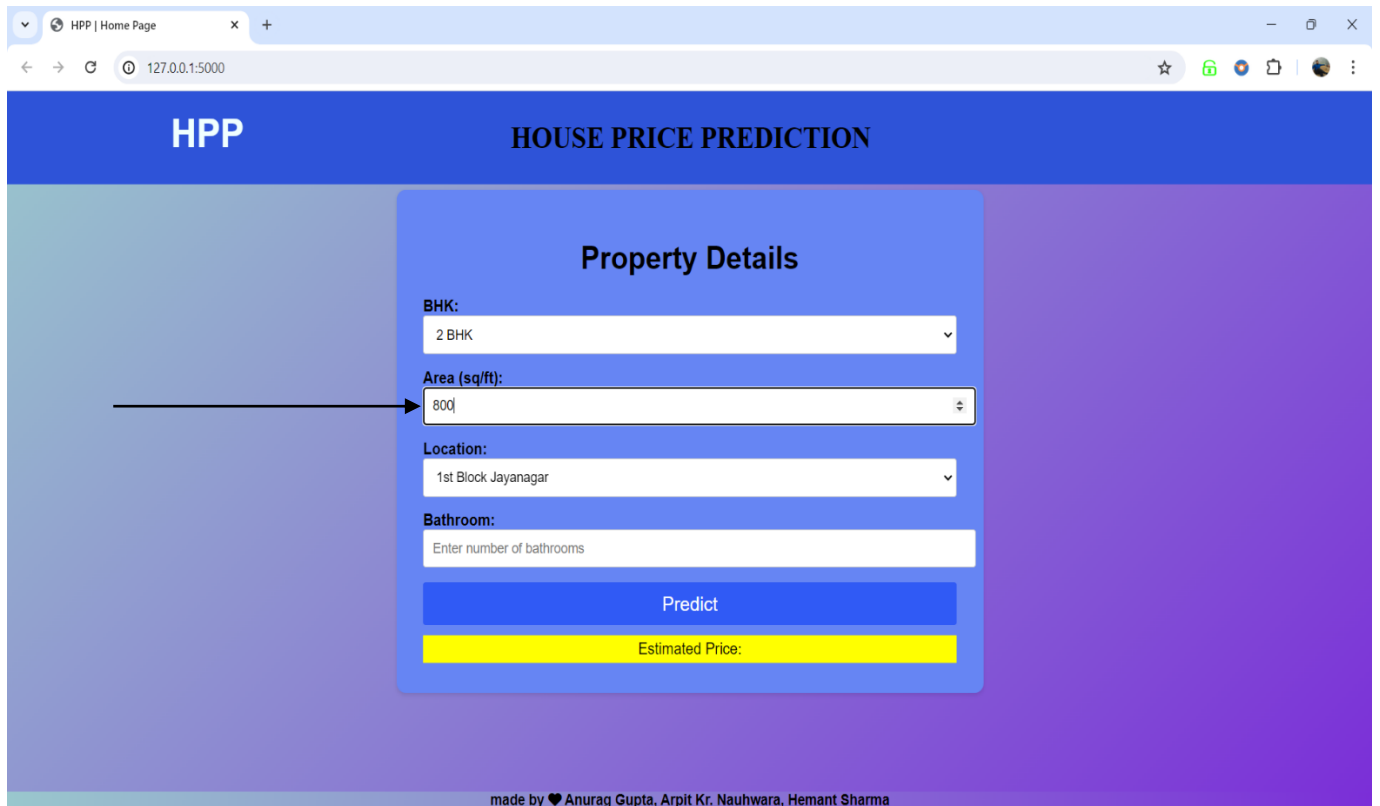
Overall, a well-designed website tutorial empowers users to maximize their experience on the website, reducing frustration and increasing engagement by providing the necessary guidance and support they need to navigate and utilize its features effectively.

Below are the steps to be followed:

Step 1: Enter the number of BHK required

The screenshot shows a web browser window with the address bar displaying 'HPP | Home Page' and the URL '127.0.0.1:5000'. The website has a blue header with 'HPP' and 'HOUSE PRICE PREDICTION'. The main content area is purple. A central white box titled 'Property Details' contains a dropdown menu for 'BHK:' with options 1 BHK through 6 BHK. An arrow points to the '1 BHK' option. Below the dropdown is a text input field for 'Bathroom:' with the placeholder 'Enter number of bathrooms'. A blue 'Predict' button is below the input field. At the bottom of the white box is a yellow box labeled 'Estimated Price:'. The footer of the website says 'made by ♥ Anurag Gupta, Arpit Kr. Nauhwara, Hemant Sharma'.

Step 2: Enter the area of the property looking for

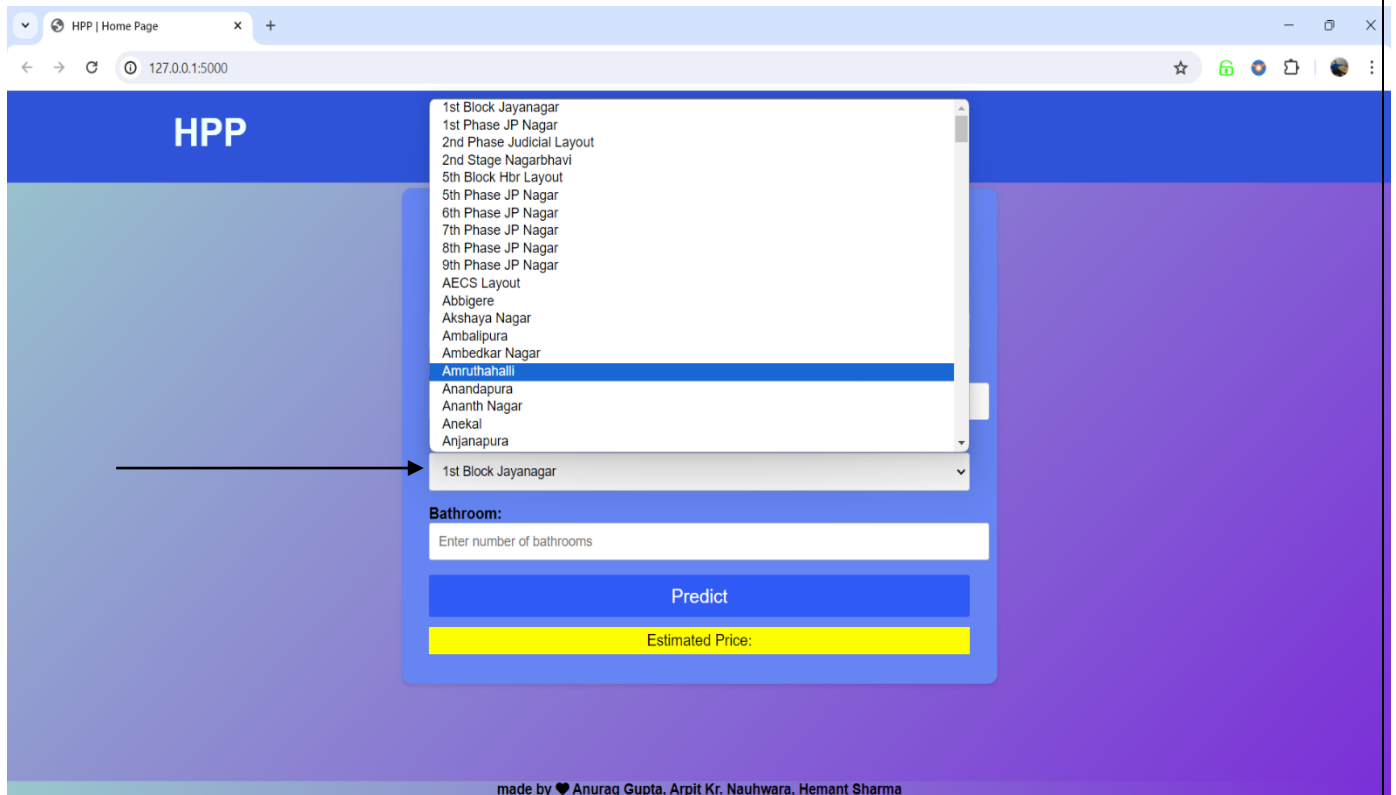


The screenshot shows a web browser window with the URL 127.0.0.1:5000. The page has a blue header with 'HPP' and 'HOUSE PRICE PREDICTION'. The main content area has a light blue background. A white box titled 'Property Details' contains the following fields:

- BHK:** A dropdown menu with '2 BHK' selected.
- Area (sq/ft):** A text input field with '800' entered. An arrow points to this field from the left.
- Location:** A dropdown menu with '1st Block Jayanagar' selected.
- Bathroom:** A text input field with the placeholder 'Enter number of bathrooms'.

Below the input fields is a blue 'Predict' button and a yellow 'Estimated Price:' field. At the bottom of the page, it says 'made by ♥ Anurag Gupta, Arpit Kr. Nauhwara, Hemant Sharma'.

Step 3: Enter the location in which property you are looking



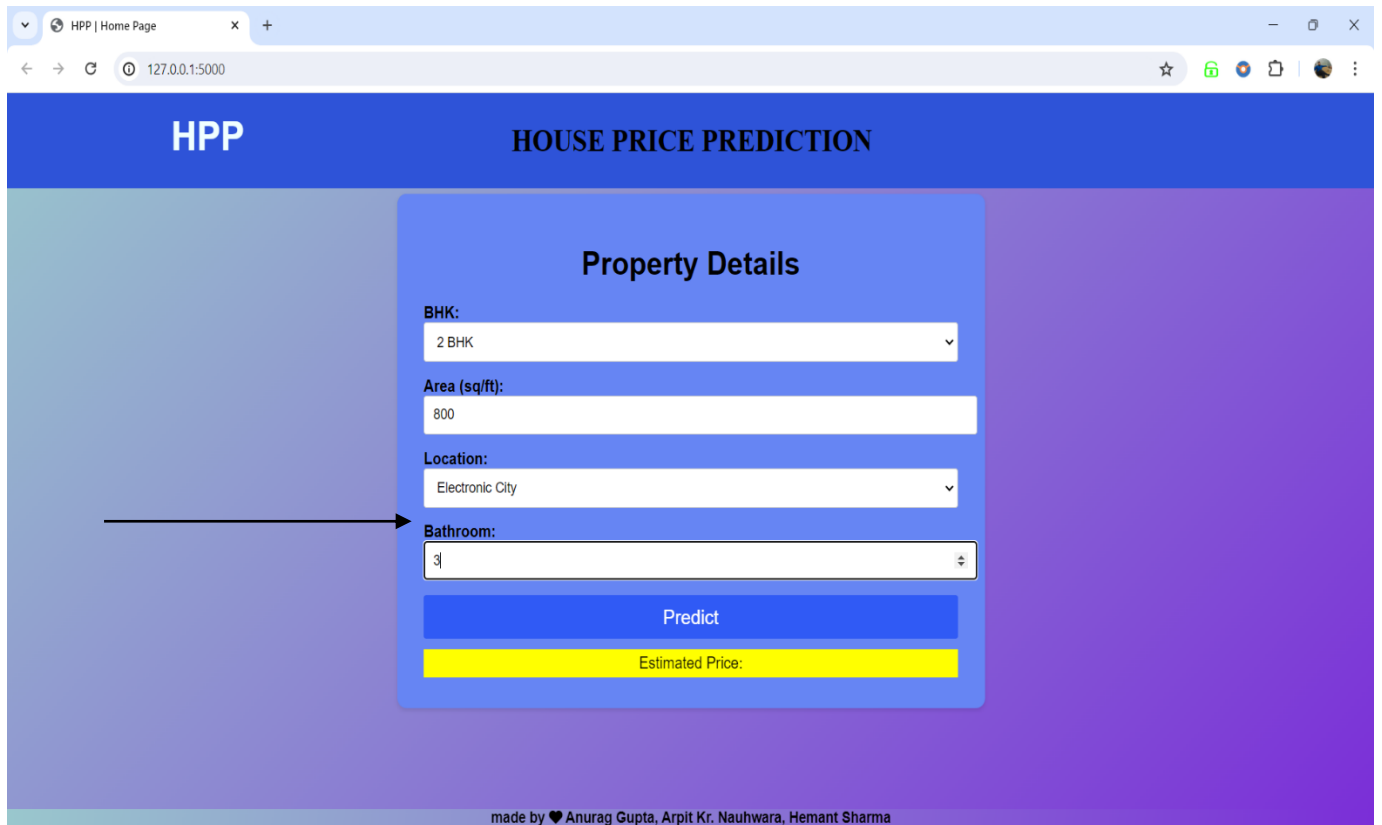
The screenshot shows the same web browser window as Step 2. The 'Location' dropdown menu is open, displaying a list of locations. An arrow points to the dropdown menu from the left.

Location Options:

- 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
- AECS Layout
- Abbigere
- Akshaya Nagar
- Ambalipura
- Ambedkar Nagar
- Amruthahalli** (highlighted)
- Anandapura
- Ananth Nagar
- Anekal
- Anjanapura

The 'Bathroom' field and the 'Predict' button are visible below the dropdown menu. At the bottom of the page, it says 'made by ♥ Anurag Gupta, Arpit Kr. Nauhwara, Hemant Sharma'.

Step 4: Enter the number of bathrooms in property



The screenshot shows a web browser window with the URL 127.0.0.1:5000. The page has a blue header with 'HPP' and 'HOUSE PRICE PREDICTION'. The main content area has a light blue background with a white 'Property Details' form. The form contains four input fields: 'BHK:' with a dropdown menu showing '2 BHK', 'Area (sq/ft):' with a text input containing '800', 'Location:' with a dropdown menu showing 'Electronic City', and 'Bathroom:' with a text input containing '3'. A blue 'Predict' button is below the form, and a yellow bar at the bottom of the form says 'Estimated Price:'. A black arrow points from the left towards the 'Bathroom:' input field. At the bottom of the page, it says 'made by ♥ Anurag Gupta, Arpit Kr. Nauhara, Hemant Sharma'.

HPP **HOUSE PRICE PREDICTION**

Property Details

BHK:
2 BHK

Area (sq/ft):
800

Location:
Electronic City

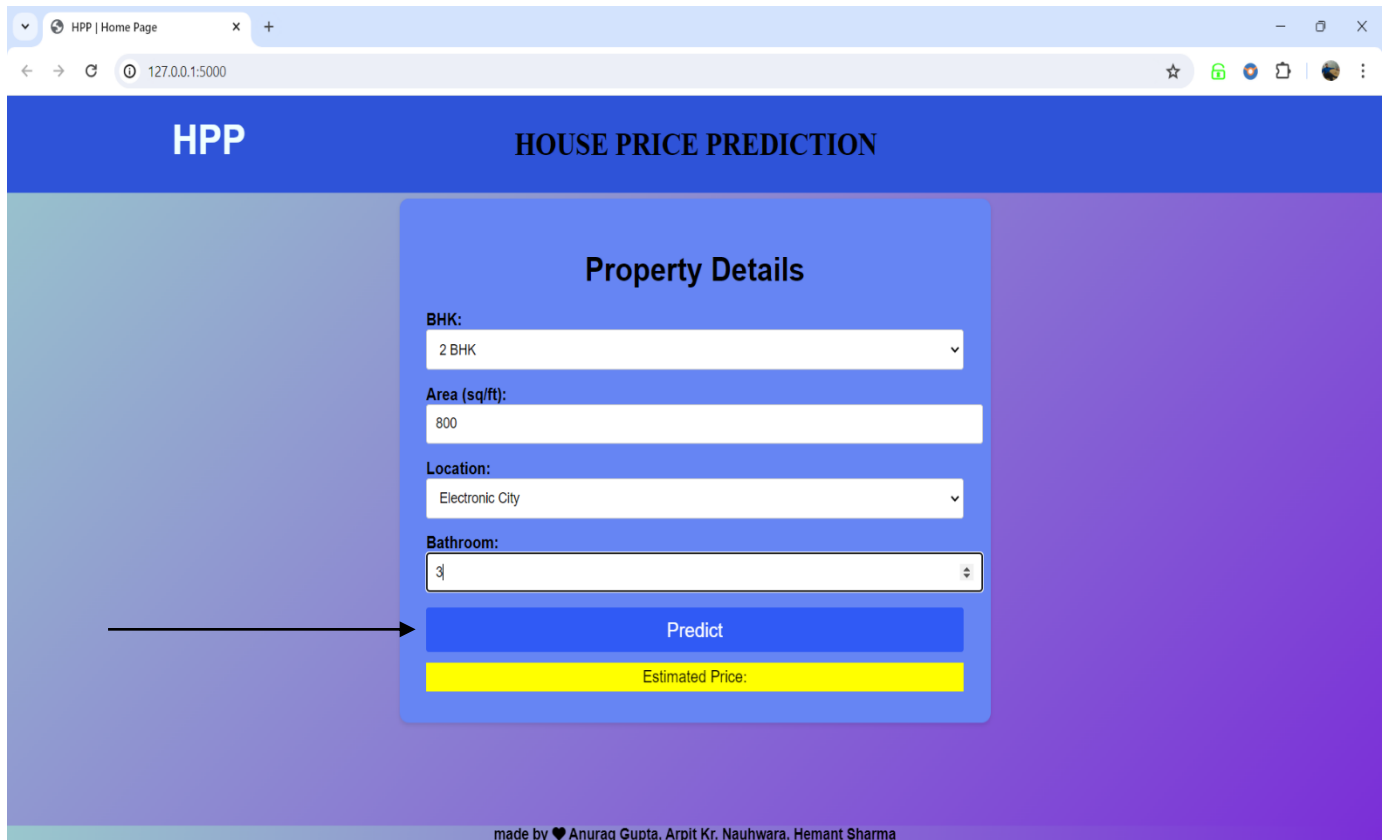
Bathroom:
3

Predict

Estimated Price:

made by ♥ Anurag Gupta, Arpit Kr. Nauhara, Hemant Sharma

Step 5: Click on Predict Button



This screenshot is identical to the one above, showing the same web browser window and form. However, a black arrow now points from the left towards the blue 'Predict' button, indicating the next step in the process.

HPP **HOUSE PRICE PREDICTION**

Property Details

BHK:
2 BHK

Area (sq/ft):
800

Location:
Electronic City

Bathroom:
3

Predict

Estimated Price:

made by ♥ Anurag Gupta, Arpit Kr. Nauhara, Hemant Sharma

Output:

The screenshot shows a web browser window with the address bar displaying 'HPP | Home Page' and the URL '127.0.0.1:5000'. The page has a blue header with the text 'HPP' and 'HOUSE PRICE PREDICTION'. The main content area has a purple gradient background. In the center, there is a light blue box titled 'Property Details' containing the following form elements:

- BHK:** A dropdown menu with '1 BHK' selected.
- Area (sq/ft):** A text input field with the placeholder 'Enter area'.
- Location:** A dropdown menu.
- Bathroom:** A text input field with the placeholder 'Enter number of bathrooms'.
- Predict:** A blue button.
- Estimated Price: 39 Lakhs:** A yellow box displaying the predicted price, with arrows pointing to it from the left and right sides.

At the bottom of the page, there is a footer that reads: 'made by ❤️ Anurag Gupta, Arpit Kr. Nauhwara, Hemant Sharma'.

11. Github & Open Source

GitHub is a platform that revolutionized the way developers collaborate on code, manage projects, and build software together. Founded in 2008 by Tom Preston-Werner, Chris Wanstrath, and PJ Hyett, GitHub quickly became the go-to platform for version control using Git, the distributed version control system created by Linus Torvalds.

At its core, GitHub provides a centralized hub for hosting Git repositories. This means developers can store their code in the cloud, making it accessible from anywhere with an internet connection. But GitHub offers much more than just a place to store code. It provides a range of tools and features designed to streamline the development process and foster collaboration among teams.

One of GitHub's most powerful features is its robust issue tracking system. Developers can use it to report bugs, suggest new features, or discuss ideas. These issues can be assigned to team members, labeled for easy organization, and linked to specific commits or branches in the codebase. This makes it easy to keep track of what needs to be done and who's responsible for doing it.

GitHub also makes it easy to review and merge code changes through its pull request system. When a developer wants to contribute to a project, they create a pull request, which allows other team members to review the proposed changes, leave comments, and suggest improvements. Once the changes have been reviewed and approved, they can be merged into the main codebase with the click of a button.

In addition to these collaboration features, GitHub also offers a wide range of integrations and extensions that extend its functionality even further. Developers can integrate GitHub with popular project management tools like Jira and Trello, continuous integration services like Travis CI and CircleCI, and code quality analysis tools like Codecov and Codacy.

GitHub has become an essential tool for developers and organizations of all sizes, from individual hobbyists to large enterprise teams. Its user-friendly interface, powerful collaboration features, and extensive ecosystem of integrations make it the platform of choice for millions of developers around the world. Whether you're working on a small open-source project or a large-scale enterprise application, GitHub provides the tools you need to build better software, together.

GitHub repository page for **house_price_prediction** by **anuraggupta19**. The repository is public and contains 13 commits. The file list includes:

File	Commit	Last Commit
artifacts	first commit	last week
media	HPP_Home_Page.mp4	last week
templates	first commit	last week
.gitignore	first commit	last week
README.md	README.md	last week
Requirements.txt	first commit	last week
app.py	first commit	last week

The README section is titled **House Price Predication Project**.

GitHub repository page for **house_price_prediction** by **anuraggupta19**. The repository is public and contains 13 commits. The file list includes:

File	Commit	Last Commit
artifacts	first commit	last week
media	HPP_Home_Page.mp4	last week
templates	first commit	last week
.gitignore	first commit	last week
README.md	README.md	last week
Requirements.txt	first commit	last week
app.py	first commit	last week

The README section is titled **House Price Predication Project**.

Objective:

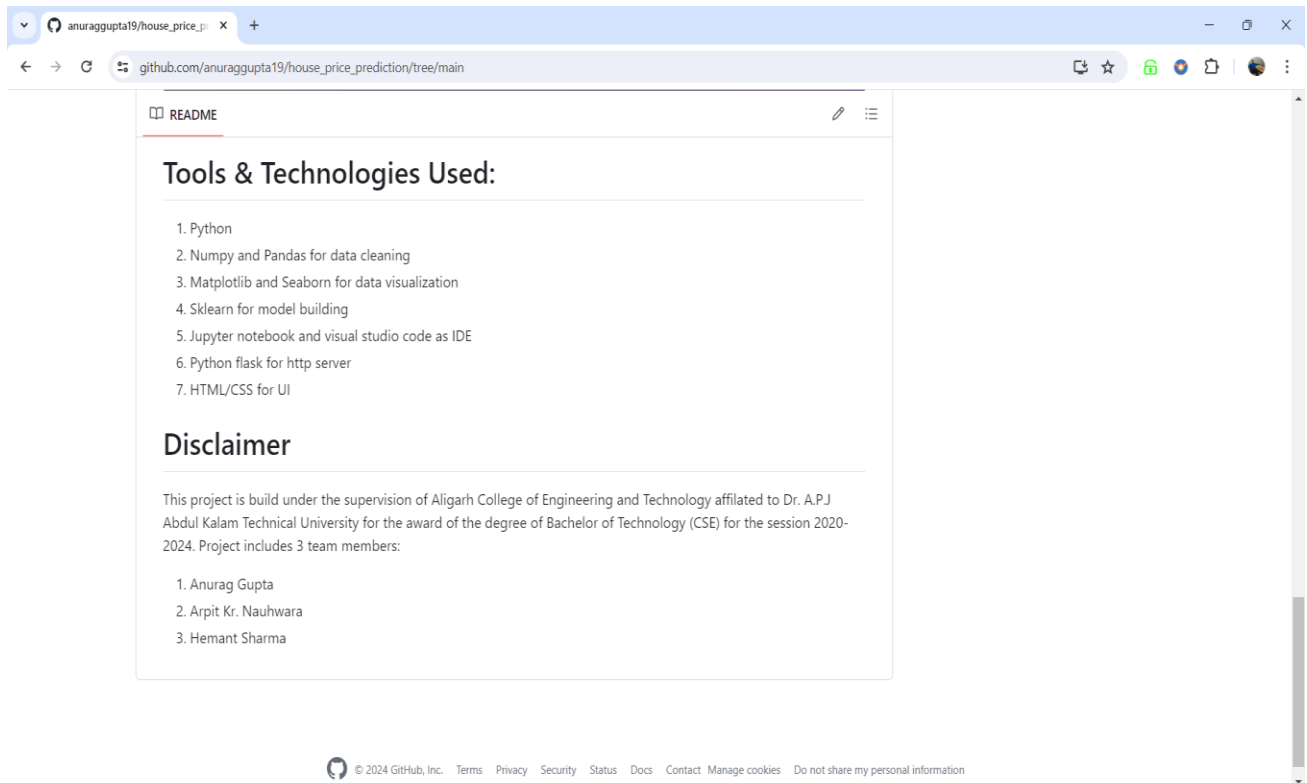
The primary objective of the House Price Analysis & Prediction project is to develop a robust and accurate machine learning model that can analyze and predict the prices of properties in Bangalore. We will first build a model using sklearn and linear regression using bangalore home prices dataset from kaggle.com. Second step would be to write a nuthon flask server that uses the saved model to serve http requests. Third component is the website built in html & CSS.

House Price Prediction Interface:

The interface shows a form for entering property details:

- BHK: 1 BHK
- Area (sqft): Enter area
- Location: 1st Block, Jayanagar
- Bathroom: Enter number of bathrooms
- Predict button
- Estimated Price: [Yellow bar]

made by ❤️ Anurag Gupta, Apjit KC, Nishu Sharma, Hemant Sharma



Github Link:

https://github.com/anuraggupta19/house_price_prediction.git

12. Contributions

Anurag Gupta (2001090100004):

- To study and convert data science prototypes.
- To design and develop Machine Learning systems and schemes.
- To train and re-train ML systems and models as and when necessary.
- To extend and enrich existing ML frameworks and libraries.
- To develop Machine Learning apps according to customer/client requirements.
- To design the website and implement the CSS.
- To develop the backend by implementing Flask.

Arpit Kumar Nauhwara (2001090100006):

- Identifying specific tasks and responsible parties will help with budgeting, implementation, and preservation of data resources
- Outline the rights and responsibilities of all project participants in relation to the management and retention of research data
- To collect the data from trusted source.
- To implement the HTML which is core of website.

Hemant Sharma (2001090100021):

- Solving data-related problems and coding issues
- Interpreting data sets for extracting relevant information benefitting the organization
- Processing data using advanced tools and summarizing it to tell a story
- Facilitating strategic decision-making within the company
- Compiling reports based on data analyzed and creating data dashboards, visualizations and graphs
- Producing and tracking key performance indicators

REFERENCE

1. www.kaggle.com
2. <https://numpy.org/>
3. <https://matplotlib.org/>
4. <https://seaborn.pydata.org/>
5. <https://scikit-learn.org/stable/index.html>
6. <https://www.python.org/>
7. <https://jupyter.org/>
8. <https://flask.palletsprojects.com/en/3.0.x/>
9. <https://pandas.pydata.org/>