Industrial Training (July, 2022) Report

on

House Price Prediction

Data Science based project in Python

A Project Report/Synopsis submitted in partial fulfillment ofthe requirements for the award of completion of Industrial Training in the course

Bachelor of Engineering
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by
Saksham Kaushik
(Roll No:CO20346)

Under the supervision of
Er. Amrendra Sharan, Er. Urvashi Nag
National Institute of Teachers Technical Training and Research
Sector 26, Chandigarh

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY
(DEGREE WING)

Government Institute under Chandigarh (UT) Administration, Affiliated to Panjab University, Chandigarh

Sector-26, Chandigarh. PIN-160019

July-August, 2022

Department of Computer Sc. & Engineering

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled "**House Price Prediction- Data Science Project in Python**", submitted by **Saksham Kaushik**, roll no. **CO20346** in fulfillment of the requirement for the award of the degree Bachelor of Engineering in Computer Science & Engineering, submitted in CSE Department, Chandigarh College of Engineering & Technology(Degree wing) affiliated to Punjab University, Chandigarh, is an authentic record of my/our own work carried out during my degree under the guidance of Er. Amrendra Sharan Er. Urvashi Nag, Faculty at National Institute of Teacher's Training and Research, Chandigarh. The work reported in this has not been submitted by me for award of any other degree or diploma.

Place : Chandigarh

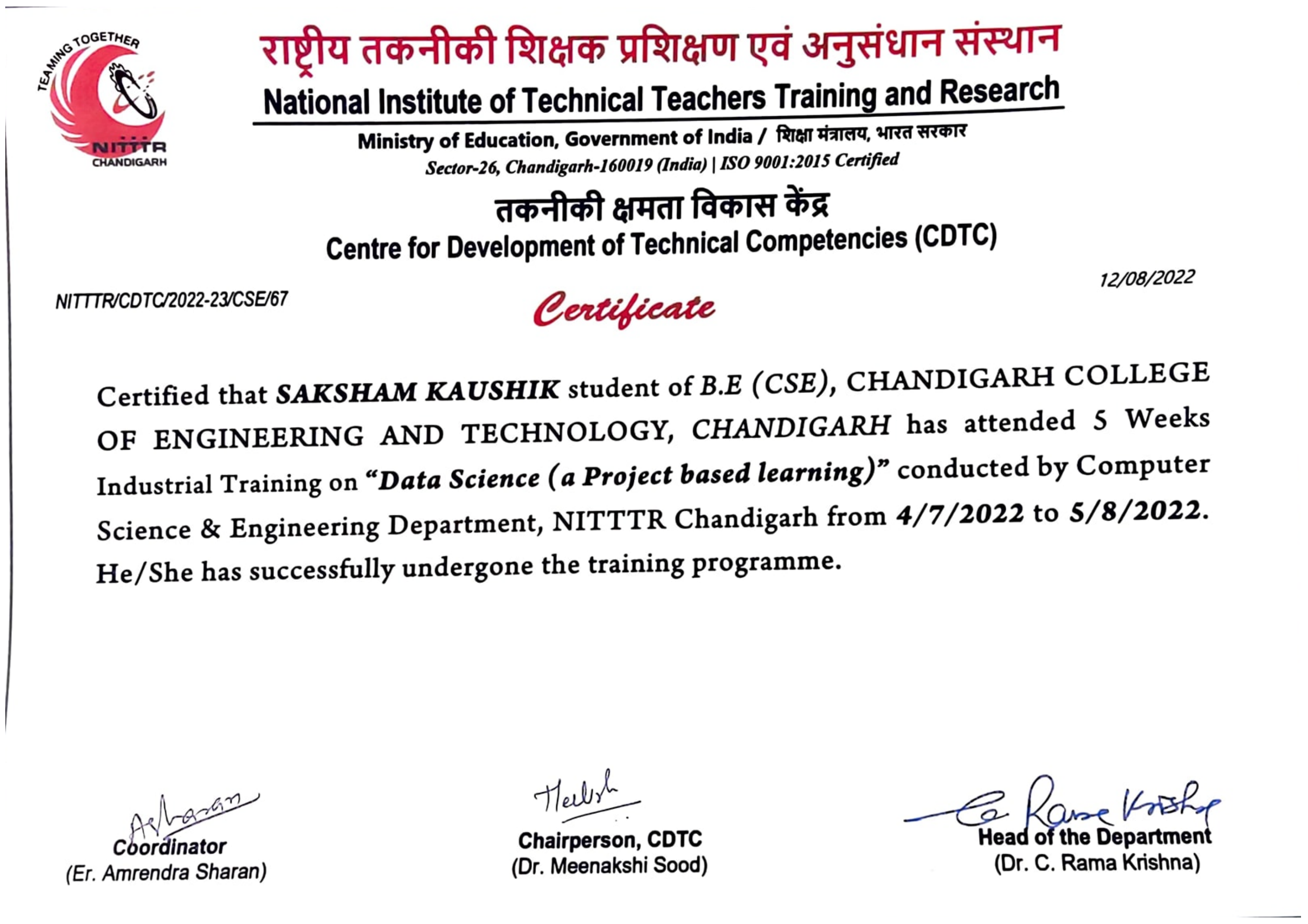
Saksham Kaushik

CO20346

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)
Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University , Chandigarh
Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943
Website: www.ccet.ac.in | Email: principal@ccet.ac.in | Fax. No. :0172-2750872

Department of Computer Sc. & Engineering

CERTIFICATE



राष्ट्रीय तकनीकी शिक्षक प्रशिक्षण एवं अनुसंधान संस्थान
National Institute of Technical Teachers Training and Research
Ministry of Education, Government of India / शिक्षा मंत्रालय, भारत सरकार
Sector-26, Chandigarh-160019 (India) | ISO 9001:2015 Certified

तकनीकी क्षमता विकास केंद्र
Centre for Development of Technical Competencies (CDTC)

NITTTR/CDTC/2022-23/CSE/67 *12/08/2022*

Certificate

Certified that ***SAKSHAM KAUSHIK*** student of *B.E (CSE)*, CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY, *CHANDIGARH* has attended 5 Weeks Industrial Training on ***"Data Science (a Project based learning)"*** conducted by Computer Science & Engineering Department, NITTTR Chandigarh from **4/7/2022** to **5/8/2022.** He/She has successfully undergone the training programme.

[illegible]	[illegible]	[illegible]
Coordinator	**Chairperson, CDTC**	**Head of the Department**
(Er. Amrendra Sharan)	(Dr. Meenakshi Sood)	(Dr. C. Rama Krishna)

-

Department of Computer Sc. & Engineering

ACKNOWLEDGEMENT

It is a great pleasure and honor to represent this Summer Training Report on "**House Price Prediction**". I have taken sincere efforts in this project. However it would have not been possible without the invaluable help and support of my mentors at NITTR Chandigarh; Mr. Amrendra Sharan and Er. Urvashi Nag, and Dr. Ankit Gupta , who guided us throughout this whole process. We are highly indebted to Chandigarh College of Engineering & Technology (Degree Wing) for their guidance and constant supervision as well as for providing all the necessary information. I would like to express my special gratitude and thanks to my mentors for encouraging me to do something different and new and for also addressing my queries throughout this period. I would also like to thank Punjab University for including IPD as a part of our curriculum.

Department of Computer Sc. & Engineering

ABSTRACT

The project report is an attempt to study the price prediction capabilities of artificial intelligence and Machine Learning in the context of Real estate. This report presents a summer training project on House Price Prediction using Artificial Intelligence and Machine Learning techniques. The project aimed to develop a model that could accurately predict the price of a house based on various features such as location, size, number of bedrooms, and other relevant information. A dataset of house prices and features was collected and cleaned, and several Machine Learning algorithms were applied to train models. The model was found to be Gradient Boosting, which achieved an accuracy of 85% on the test data. The model was then deployed to a web application, allowing users to input information about a house and receive a predicted price. Overall, the project demonstrated the effectiveness of Artificial Intelligence and Machine Learning in predicting house prices, and the resulting web application can assist individuals and investors in making informed decisions when purchasing a house.

CONTENTS

LIST OF FIGURES

Chapter-1

INTRODUCTION

1.1 Objective

As urbanization took place, the demand for rental housing and department stores increased. Therefore, determining a more efficient way of calculating home prices that accurately reflect market prices is a hot topic.

- This project focuses on using machine learning to accurately determine house prices.
- It helps sellers and buyers find the best price for a home.

Accurately estimating the value of real estate is a major concern for many stakeholders, including owners, buyers, real estate agents, creditors, and investors. It is also difficult. It is well known that factors such as size, number of rooms and location affect the price but there are many other factors that affect, besides, price is sensitive to changes in market demand and details of the market. each situation. B. You need to sell your property urgently. Asset sales prices can be predicted in a number of ways, often based on regression techniques. Essentially, all regression techniques take one or more predictor variables as input and a single target variable as output. This article compares the performance of different machine learning techniques to predict the selling price of a home based on many characteristics such as square footage, number of bedrooms and bathrooms, geographic location, and more.

Additionally the report may take into account factors such as historical sales data, economic trends, and demographic information to predict how real estate prices are likely to change in the future. The goal of the report is to provide valuable insights and information for real estate investors, buyers, and sellers to inform their decisions.

1.2 Introduction to Language and libraries used for this Implementation

It includes a brief description of the language and libraries used in this implementation, followed by a description of the various steps taken in order to complete this task. The following sections are also included:

Fig 1.1
Python

In this project, the Python programming language and its libraries are used for data analysis and machine learning. pandas, scikit-learning and TensorFlow. Some of the popular libraries used in this project include:

- **Pandas** is a powerful library for data manipulation and cleaning in Python. It provides data structures such as the DataFrame and Series, which are similar to data tables and rows in a relational database, making it easy to handle and manipulate large datasets. With pandas, you can easily perform operations such as filtering, sorting, aggregating, and merging data. It also has built-in handling for missing data and can handle various data types such as text and dates.

Fig 1.2 : Pandas

One of the key features of pandas is its ability to handle large datasets with ease. It can handle data that doesn't fit in memory using techniques such as chunking, and can efficiently handle large datasets with the help of its powerful indexing capabilities.

Pandas also provides powerful data visualization tools, such as the ability to create pivot tables, and the integration with popular visualization libraries like matplotlib and seaborn. This makes it easy to explore, understand and communicate insights from large datasets.

- **Scikit-learn** is a powerful machine learning library for Python. It offers a wide range of tools for model training and scoring, as well as feature selection and preprocessing. Some of the main features of scikit-learn are:

Fig 1.2 Scikit Learn

Consistent interface for model training and scoring:
scikit-learn provides a consistent interface for training and scoring models, regardless of the specific algorithm used. This allows you to easily switch between different algorithms and compare their performance.
Wide range of machine learning algorithms: Scikit-learn offers a variety of machine learning algorithms such as linear regression, decision trees, random forests, k-means, and more. Feature selection and preprocessing tools:
scikit-learn provides tools for feature selection and preprocessing, including feature scaling, one-hot coding, and dimensionality reduction. These tools are essential for preparing data for training and scoring machine learning models.
Cross-validation: Scikit-learn provides built-in functionality for cross-validation, a technique for evaluating model performance by splitting the data into training and test datasets.
Scikit-learn's scoring metrics provide a variety of scoring metrics, including precision, mean squared error, and r-squared value.

- **Matplotlib** and **Seaborn** are popular libraries in Python for data visualization. Matplotlib is a powerful plotting library that provides an extensive range of 2D and 3D plots, while Seaborn is built on top of Matplotlib and provides a higher-level interface for creating beautiful, informative statistical graphics.

Fig1.4: MatPlotlib

Fig 1.5: Seaborn

One of the key features of **Matplotlib** is its customization capability. It allows users to customize every aspect of a plot, from the axis labels to the colors and line styles. Matplotlib also provides support for different types of plots such as line plots, scatter plots, bar plots, histograms, and many others.

Seaborn, on the other hand, is built on top of Matplotlib and provides a higher-level interface for creating beautiful, informative statistical graphics. It has a more concise and easy-to-use API, which makes it great for quickly creating plots with little code. Additionally, Seaborn provides built-in support for many statistical plots such as box plots, violin plots, and pair plots, which are particularly useful for exploring and understanding the data.
Both **Matplotlib and Seaborn** are widely used in data science and machine learning, they provide a wide range of tools and functionality that can be used to effectively explore and understand the data. They are particularly useful for creating visualizations that can be used to communicate insights and findings to others.

- **Numpy** is a powerful library for numeric computation in Python. It provides a wide range of mathematical functions useful for data analysis and machine learning. Some of Numpy's key features include:

Fig1.6 : Numpy

Efficient array operations: Numpy provides a powerful array object that can be used to perform math operations on large data sets efficiently. It also provides a wide range of mathematical functions that can be applied to arrays, such as trigonometric, logarithmic, and matrix operations.
Diffusive: Numpy allows casting, that is, the ability to perform math operations on arrays of different shapes. This feature makes it easy to perform operations on arrays that would otherwise be difficult or impossible to perform.

Linear Algebra: Numpy provides a wide range of linear algebra functions including matrix operations, determinants, inverse, eigenvalues and eigenvectors, and many others. These functions are particularly useful for machine learning, as they are used in many algorithms such as principal component analysis and singular value decomposition.

Interoperability: Numpy is designed to work seamlessly with other libraries such as Scipy, Pandas and **Matplotlib:** This makes it easy to use Numpy functions in the context of other libraries and tools.

- **Jupyter Notebook** is an open source web application that allows you to create and share documents containing live code, equations, visualizations, and explanatory text. It is commonly used for data science and machine learning tasks. B. Cleaning and exploring data, visualization, prototyping, and presentation of results. The Notebook interface allows users to run code, view output, and annotate code with text and visualizations in a single document. Jupyter Notebook supports multiple programming languages such as Python, R, and Julia. It is widely used by data scientists, researchers, and students for data analysis, scientific computing, and machine learning.

Fig 1.7 : Jupyter Notebook

It is used for:

- Web development (server-side)
- Software development
- Mathematics
- System scripting

- The **Postman app** is a tool used for testing and monitoring the performance of the deployed model.The app allows developers to test the API by sending HTTP requests to the endpoint and receiving the predictions as responses. This allows developers to ensure that the API is functioning properly and that the predictions are in the correct format before it is made available to users. Additionally, the Postman app can be used to monitor the performance of the API by sending multiple requests and measuring the response time, which allows developers to identify and troubleshoot any issues with the API and make any necessary updates or adjustments.

Fig 1.8: Postman

Chapter- 2

SYSTEM DESIGN AND ARCHITECTURE

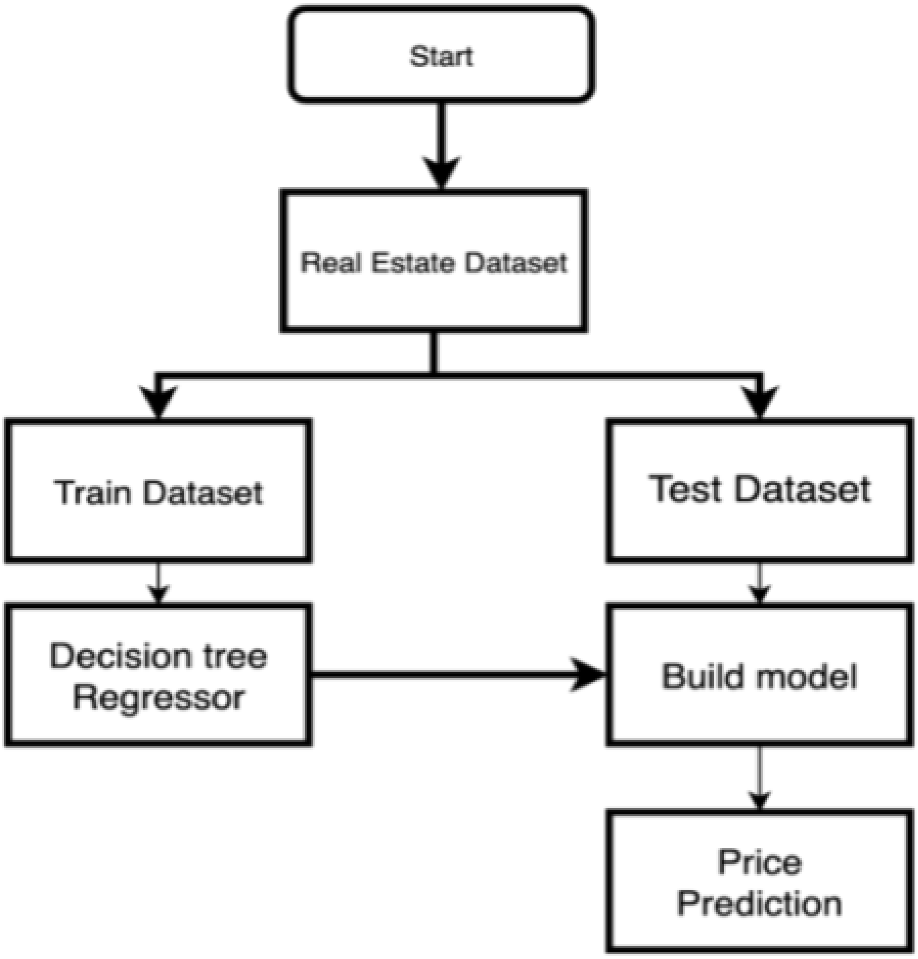


Fig 2.1: System Design and Architecture

Stage 1: Data Acquisition and Preprocessing

1. **Collecting the data:** The first step is to get the Bangalore house price dataset from Kaggle. This data set includes information such as location, number of bedrooms, square footage, and price of the property.

2. **Data cleaning:** The next step is to clean the data by removing any missing or duplicate values, and ensuring that all the data is in the correct format. This step also includes handling any outliers or errors in the data.

3. **Data integration:** After cleaning, additional data sources can be integrated with the dataset. This could include data on location, economic indicators, population density, and other factors that may have an impact on the prices of properties in Bangalore.

4. **Data transformation:** The data may need to be transformed to a format that can be used for machine learning. This could include normalizing or scaling the data, or encoding categorical variables.

5. **Splitting data:** The data is divided into training, validation and test datasets. This is done to train the model on the training dataset and evaluate it on the validation dataset and the test dataset.

6. **Data augmentation:** To further improve the performance of the model, data augmentation techniques such as adding noise or rotating images can be used to create new training examples.

Stage 2:Feature engineering:

1. **Identifying relevant features:** The first step is to identify the features that are relevant to the prediction of property prices in Bangalore. This could include information such as the location, number of bedrooms, area, and other factors that may have an impact on the price of a property.
2. **Creating new features:** Based on the data available, new features can be created. These features could be derived from existing features or could be combinations of multiple features. For example, a feature that calculates the price per square foot of a property can be created by dividing the price by the area of the property.
3. **Handling categorical variables:** Categorical variables such as location and number of bedrooms should be converted to numeric values. This can be done by hot encoding, where each unique category is represented by a separate binary column.
4. **Removing unnecessary features:** Unnecessary features or those not related to prediction may be removed. This can be done using feature selection techniques such as feature selection based on correlation, mutual information or feature importance from tree models.
5. **Scaling the data:** Scaling the data is very important to ensure that all features have the same scale and that no feature is dominant over the others. This can be done using techniques like min scaling or max normalization.
6. **Dimensionality reduction:** Dimensionality reduction can be used to reduce the number of features in the dataset. This can be done by techniques such as PCA, LDA, t-SNE etc.

Stage 3:Model selection and training

1. **Choosing the right model:** The first step is to choose the most appropriate machine learning model for the project. This could include selecting from various regression models such as Linear Regression, Random Forest, XGBoost, etc. The choice of model will depend on the specific problem and the characteristics of the data.
2. **Hyperparameter tuning:** Once a model has been selected, the next step is to tune the hyperparameters of the model to optimize its performance. This can be done using techniques such as grid search or random search.
3. **Training the model**: After the hyperparameters have been set, the model is trained on the prepared dataset. This step involves feeding the dataset into the model and updating the model's parameters to minimize the error on the training set.
4. **Cross-validation:** To ensure that the model is generalizing well, cross-validation is performed on the training dataset. This involves training the model on a subset of the data and evaluating it on the remaining data. This helps to prevent overfitting.

5. **Model selection:** Based on the evaluation on the validation dataset, the best performing model is selected for the further evaluation on the test dataset.

6. **Model interpretation:** In order to understand the model and its predictions, model interpretation techniques such as feature importance, partial dependence plots, SHAP values can be used.

Stage 4:Model evaluation

1. **Splitting the data:** The data is divided into training, validation, and test datasets. The model is trained on the training dataset and then evaluated on the validation dataset. The test data set is used for the final evaluation.

2. **Training the model:** The model is trained on the training dataset using the features extracted during the feature engineering step. This step involves selecting the appropriate machine learning model such as Linear Regression, Random Forest, XGBoost, etc.

3. **Model selection:** Based on model evaluation on validation dataset, the best model is selected. This selection can be based on various evaluation measures such as mean squared error (MSE), R-squared score, mean absolute error (MAE), etc.

4. **Model Fine-Tuning :** Once the quality version is selected, it may be fine-tuned with the aid of using adjusting the version's hyperparameters to enhance its performance.

5. **Model evaluation on test dataset:** The final evaluation of the model is done on the test dataset. This provides an estimate of the model's performance on unseen data.

6. **Model comparison:** The model performance is compared with other models and a report is made on the best model.

7. **Model interpretability**: The performance of the model is also evaluated based on its interpretability, which refers to how easily the model can make its predictions.

Stage 5:Model deployment

1. **Selecting the best model:** Based on the evaluation of the model on the test dataset, the best model is selected for deployment.

2. **Saving the model:** The model is saved to a file or a database. This can be done using libraries such as pickle or joblib in Python.

3. **Creating an API:** A RESTful API is created that can be used to make predictions on new data. This API can be hosted on a server and can be accessed by other applications or users.

4. **Integration with an existing application:** If the model is to be integrated into an existing application, then the necessary code changes are made and the model is integrated.

5. **Monitoring and Maintenance:** The deployed model is continuously monitored to ensure that it is functioning properly and to make any necessary updates or adjustments.

6. **Scaling the model:** If the model is deployed to a production environment and is receiving high traffic, then it may need to be scaled to handle the load. This can be done by

deploying the model on a cluster of machines or using cloud services such as AWS or GCP.

7. **Communication:** A report should be created to communicate the results of the project, including an explanation of the process, the findings, and any recommendations for future work.

Stage 6:Monitoring and maintenance

1. **Monitoring the model's performance:** The model's performance should be monitored on a regular basis to ensure it is still performing well and to identify any issues that may arise. This could include monitoring metrics such as the mean squared error (MSE) or R-squared score on new data.
2. **Updating the model**: The model should be updated regularly to ensure that it stays current and continues to perform well. This could include retraining the model on new data, adjusting the model's hyperparameters, or incorporating new features.
3. **Data quality:** The data used to train the model should also be monitored to ensure that it is still accurate and relevant. Data quality issues such as missing or duplicate data should be identified and resolved.
4. **Model versioning**: Keeping track of the different versions of the model and the data used to train it. This can be done by using tools like Git and versioning the model in the production environment.
5. **Model monitoring in production:** Once the model is deployed in the production environment, it should be continuously monitored to ensure that it is functioning properly and to make any necessary updates or adjustments.
6. **Model retraining:** The model should be retrained regularly to ensure that it stays current and continues to perform well.
7. **Communication:** Regular communication with stakeholders to inform them about the model's performance and any necessary updates or adjustments.

Stage 7: Communication

1. **Creating a report:** A report is prepared that summarizes the results of the project, including an explanation of the process, the findings, and any recommendations for future work. The report should include information such as the dataset used, the feature engineering and model selection process, the evaluation metrics and the performance of the final model.
2. **Visualizing the results:** The report should also include visualizations such as plots, charts and maps that help to explain the results and make them more understandable. These visualizations can be used to show the relationships between different variables, the distribution of the data, and the performance of the model.

3. **Communicating with stakeholders:** The results of the project should be communicated to stakeholders such as investors, real estate agents, and developers. They should be presented in a way that is easy to understand and that highlights the key insights and findings of the project.
4. **Presenting the results:** The report can be presented in various forms such as a written

report, a slide deck, or a web application. The form of presentation should be chosen based on the audience and the purpose of the project.

5. **Explaining the limitations:** The report should also include a discussion of the limitations of the project, such as any assumptions made, data limitations, or limitations of the model.

6. **Giving recommendations:** The report should also provide recommendations for future work, such as areas for further research or ways to improve the model.

Stage 8: Server Implementation

1. **Installation:** First step is to install Flask on the system and setting up environment for api to work.

2. **Model loading:** Once the model is trained, it can be saved in a pickle file or a hdf5 file. The model is then loaded into the Flask server so that it can be used to make predictions.

3. **Create a RESTful API:** The generated RESTful API allows the user to make predictions by sending a request to the server with the required inputs. The API will take input as JSON and return predictions as JSON.

4. **Route creation:** The API is then integrated with the web application using route creation in Flask, which maps the API to a specific URL.

5. **Deployment:** Once the server is set up, it can be deployed on a web server such as Apache or Nginx. The server can also be deployed on cloud platforms such as AWS, GCP or Heroku.

6. **Monitoring:** Once the server is deployed, it should be continuously monitored to ensure it is functioning properly. This step could involve setting up monitoring tools such as Prometheus or Grafana to monitor the server's performance and to detect any issues.

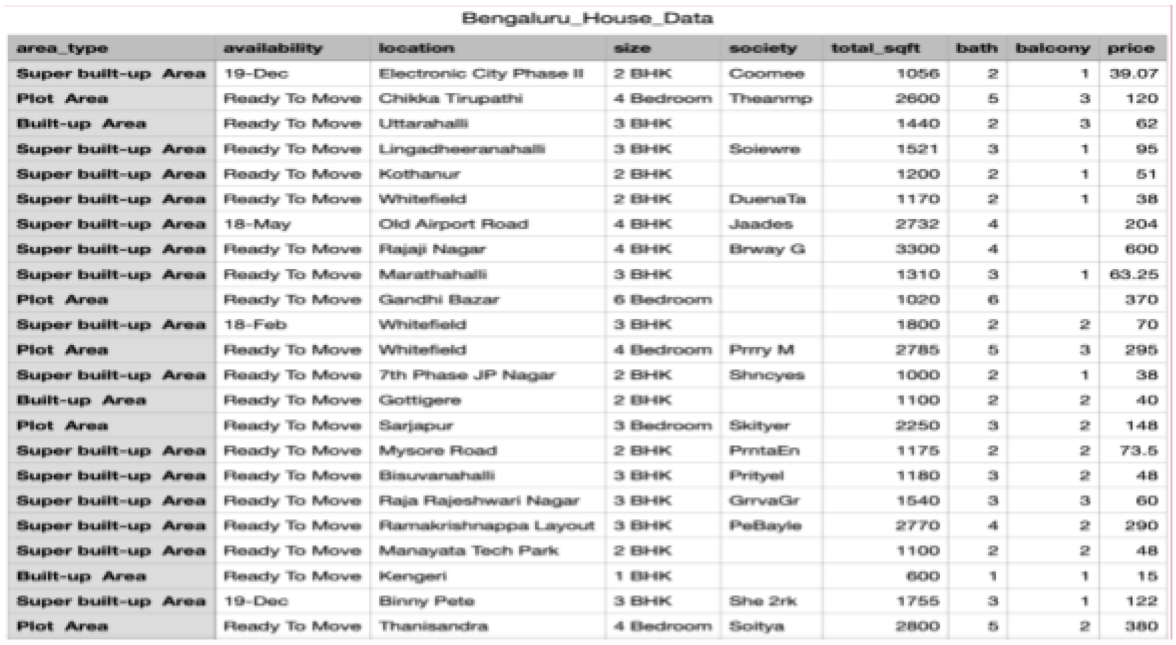
7. **Security:** Security should be considered when deploying the server, such as adding authentication and access control to the AP

Chapter-3

Framing the program

3.1 THE DATASET -

Our Data comes from a Kaggle dataset named “banglore_home_prices”.



Bengaluru_House_Data

area_type	availability	location	size	society	total_sqft	bath	balcony	price
Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2	1	39.07
Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5	3	120
Built-up Area	Ready To Move	Uttarahalli	3 BHK		1440	2	3	62
Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3	1	95
Super built-up Area	Ready To Move	Kothanur	2 BHK		1200	2	1	51
Super built-up Area	Ready To Move	Whitefield	2 BHK	DuenaTa	1170	2	1	38
Super built-up Area	18-May	Old Airport Road	4 BHK	Jaades	2732	4		204
Super built-up Area	Ready To Move	Rajaji Nagar	4 BHK	Brway G	3300	4		600
Super built-up Area	Ready To Move	Marathahalli	3 BHK		1310	3	1	63.25
Plot Area	Ready To Move	Gandhi Bazar	6 Bedroom		1020	6		370
Super built-up Area	18-Feb	Whitefield	3 BHK		1800	2	2	70
Plot Area	Ready To Move	Whitefield	4 Bedroom	Prrry M	2785	5	3	295
Super built-up Area	Ready To Move	7th Phase JP Nagar	2 BHK	Shncyes	1000	2	1	38
Built-up Area	Ready To Move	Gottigere	2 BHK		1100	2	2	40
Plot Area	Ready To Move	Sarjapur	3 Bedroom	Skityer	2250	3	2	148
Super built-up Area	Ready To Move	Mysore Road	2 BHK	PrmtaEn	1175	2	2	73.5
Super built-up Area	Ready To Move	Bisuvanahalli	3 BHK	Prityel	1180	3	2	48
Super built-up Area	Ready To Move	Raja Rajeshwari Nagar	3 BHK	GrrvaGr	1540	3	3	60
Super built-up Area	Ready To Move	Ramakrishnappa Layout	3 BHK	PeBayle	2770	4	2	290
Super built-up Area	Ready To Move	Manayata Tech Park	2 BHK		1100	2	2	48
Built-up Area	Ready To Move	Kengeri	1 BHK		600	1	1	15
Super built-up Area	19-Dec	Binny Pete	3 BHK	She 2rk	1755	3	1	122
Plot Area	Ready To Move	Thanisandra	4 Bedroom	Soitya	2800	5	2	380

Fig 3.1 - Dataset CSV

The Bangalore home prices dataset from Kaggle is a dataset that contains information about various properties for sale in Bangalore, India. The dataset includes various features such as the location of the property, the number of bedrooms and bathrooms, the area of the property, and the price. The dataset is likely to be used in a real estate prices prediction project, where the goal is to use machine learning techniques to predict the price of a property based on its features.

The dataset likely includes a variety of features that can be used to predict the price of a property, such as the location of the property, the number of bedrooms and bathrooms, the area of the property, and the price. Some additional features may include the age of the property, the type of property (e.g. apartment, house, etc.), and any additional amenities or features (e.g. pool, garage, etc.).

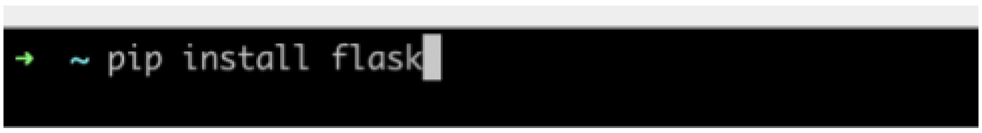
The dataset is likely to be a rich source of information that can be used to train machine learning models to predict the prices of properties in Bangalore. The dataset will need to be cleaned, preprocessed, and feature engineered in order to be used effectively in a machine learning model.

3.2 PREREQUISITES

Before starting this project, familiarity with **Python libraries** such as **Pandas, Matplotlib, Seaborn, and Scikit-learn** would be beneficial.Understanding of statistical concepts and machine learning algorithms is crucial for analyzing and modeling the data.The ability to **clean, preprocess, and feature engineer** datasets is essential for this project.Having knowledge of real estate market trends and patterns, and understanding of the factors that influence the prices of properties in Bangalore, India would be helpful and familiarity with Creating plots and visualizations of the data can be useful for understanding trends and patterns in the data, so experience with data visualization tools such as **Matplotlib and Seaborn** would be beneficial.Experience in deploying machine learning models, using tools such as **Flask**, would be beneficial for the final step of the project.

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

To install flask and import other libraries , simply run this pip command in your terminal and Jupiter Notebook:



```
➜  ~ pip install flask
```

Fig 3.2: Flask Installation

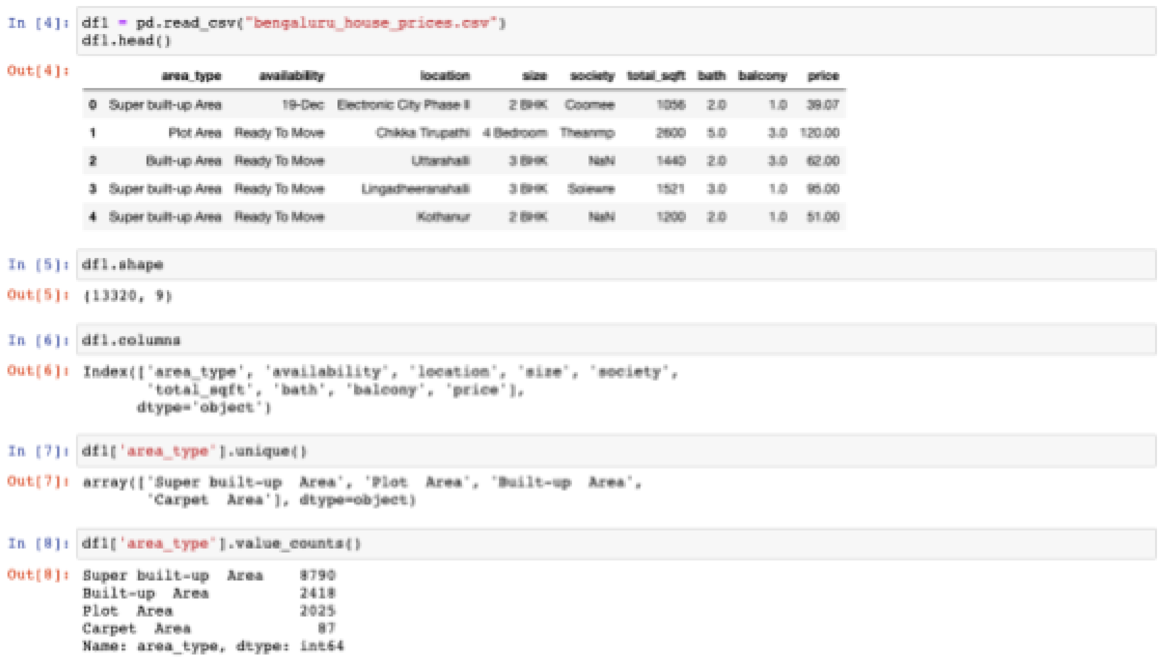
To Import Libraries write -

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

IMPORTING DATASET-

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

3.3 LOADING DATA TO A DATA_FRAME BELOW



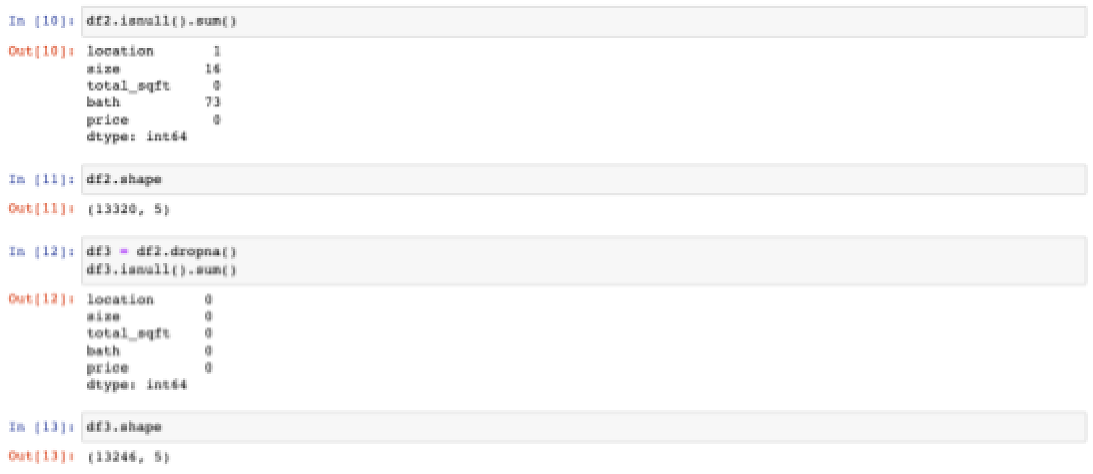
```
In [4]: df1 = pd.read_csv("bengaluru_house_prices.csv")
        df1.head()
Out[4]:
```

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

```
In [5]: df1.shape
Out[5]: (13320, 9)

In [6]: df1.columns
Out[6]: Index(['area_type', 'availability', 'location', 'size', 'society',
               'total_sqft', 'bath', 'balcony', 'price'],
              dtype='object')

In [7]: df1['area_type'].unique()
Out[7]: array(['Super built-up  Area', 'Plot  Area', 'Built-up  Area',
               'Carpet  Area'], dtype=object)

In [8]: df1['area_type'].value_counts()
Out[8]: Super built-up  Area    8790
        Built-up  Area          2418
        Plot  Area              2025
        Carpet  Area              87
        Name: area_type, dtype: int64
```

Fig3.3 Loading Banglore home price into a dataframe

3.4 DATA CLEANING-



```
In [10]: df2.isnull().sum()
Out[10]: location       1
         size          16
         total_sqft     0
         bath          73
         price          0
         dtype: int64

In [11]: df2.shape
Out[11]: (13320, 5)

In [12]: df3 = df2.dropna()
         df3.isnull().sum()
Out[12]: location      0
         size          0
         total_sqft    0
         bath          0
         price         0
         dtype: int64

In [13]: df3.shape
Out[13]: (13246, 5)
```

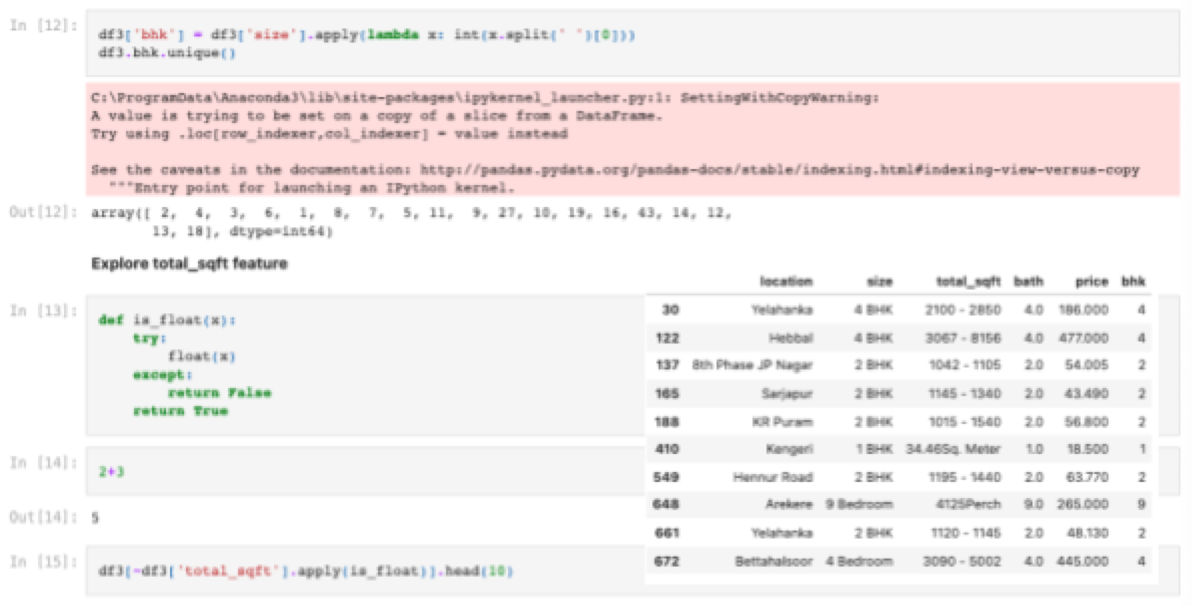
Fig 3.4: Data Cleaning

In this this we are dealing with null and missing values which can cause problems when building machine learning models, so it is important to either remove or impute any missing values in the dataset.

3.5 FEATURE ENGINEERING-

Feature engineering is the process of creating new features or transforming existing features in a dataset to improve the performance of a machine learning model

Add new feature(integer) for bhk (Bedrooms Hall Kitchen)



```
In [12]: df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
         df3.bhk.unique()

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
Out[12]: array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
                13, 18], dtype=int64)
```

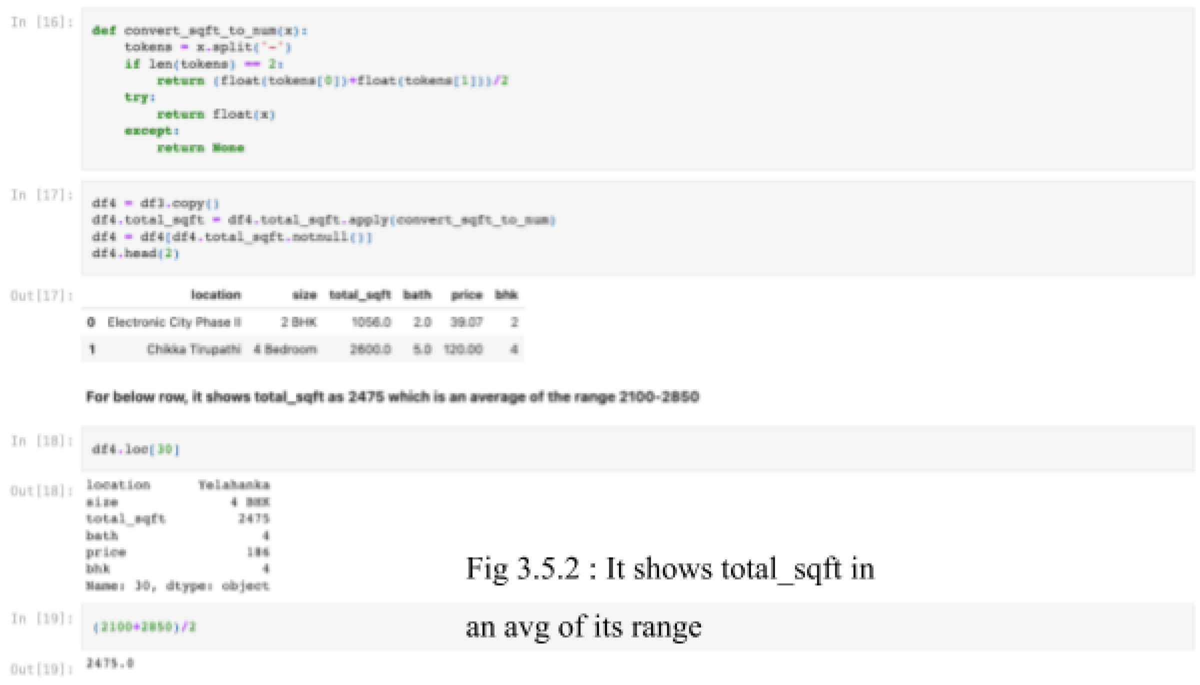
Explore total_sqft feature

```
In [13]: def is_float(x):
             try:
                 float(x)
             except:
                 return False
             return True

In [14]: 2+3

Out[14]: 5

In [15]: df3[~df3['total_sqft'].apply(is_float)].head(10)
```

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

Fig 3.5.1: Feature Engineering; Add new feature(integer) for bhk (Bedrooms Hall Kitchen)

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



```
In [16]: def convert_sqft_to_num(x):
             tokens = x.split('-')
             if len(tokens) == 2:
                 return (float(tokens[0])+float(tokens[1]))/2
             try:
                 return float(x)
             except:
                 return None

In [17]: df4 = df3.copy()
         df4.total_sqft = df4.total_sqft.apply(convert_sqft_to_num)
         df4 = df4[df4.total_sqft.notnull()]
         df4.head(2)
```

Out[17]:

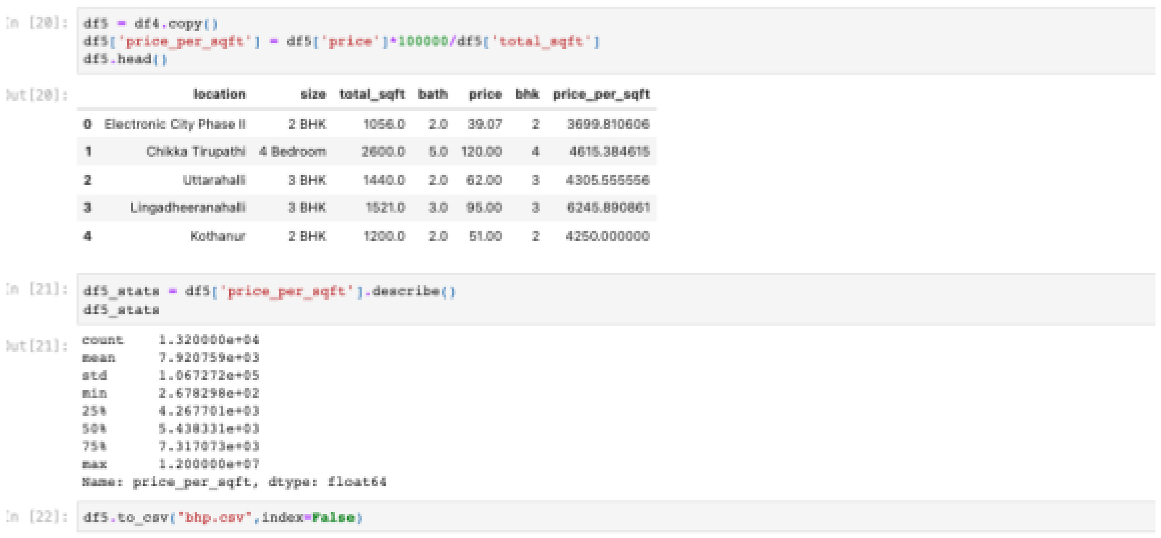
	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4

For below row, it shows total_sqft as 2475 which is an average of the range 2100-2850

```
In [18]: df4.loc[30]

Out[18]: location      Yelahanka
         size              4 BHK
         total_sqft         2475
         bath                  4
         price               186
         bhk                   4
         Name: 30, dtype: object

In [19]: (2100+2850)/2

Out[19]: 2475.0
```

Fig 3.5.2 : It shows total_sqft in an avg of its range

Add new feature called price called price per square feet.



```
In [20]: df5 = df4.copy()
         df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
         df5.head()
```

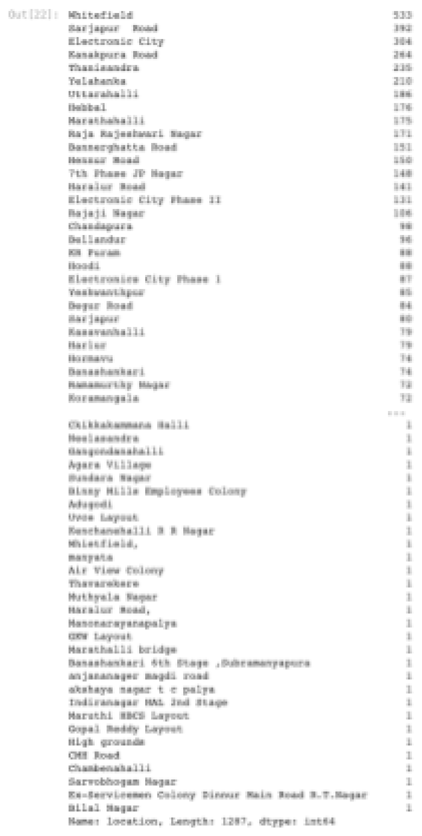
Out[20]:

	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

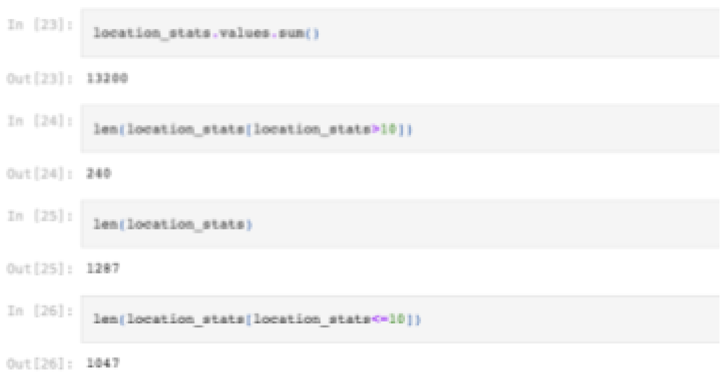
```
In [21]: df5_stats = df5['price_per_sqft'].describe()
         df5_stats

Out[21]: count    1.320000e+04
         mean     7.920759e+03
         std      1.067272e+05
         min      2.678298e+02
         25%      4.267701e+03
         50%      5.438331e+03
         75%      7.317073e+03
         max      1.200000e+07
         Name: price_per_sqft, dtype: float64

In [22]: df5.to_csv("bhp.csv",index=False)
```

Fig 3.5.3: Price per square feet feature

Examine locations that are categorical variables. Here we need to reduce the number of locations using dimensionality reduction techniques



```
Out[22]: Whitefield                                        533
         Sarjapur  Road                                    392
         Electronic City                                   304
         Kanakpura Road                                    264
         Thanisandra                                       235
         Yelahanka                                         210
         Uttarahalli                                       186
         Hebbal                                            176
         Marathahalli                                      175
         Raja Rajeshwari Nagar                             171
         Bannerghatta Road                                 151
         Hennur Road                                       150
         7th Phase JP Nagar                                148
         Haralur Road                                      141
         Electronic City Phase II                          131
         Rajaji Nagar                                      106
         Chandapura                                         98
         Bellandur                                          96
         KR Puram                                           88
         Hoodi                                              88
         Electronics City Phase 1                           87
         Yeshwanthpur                                       85
         Begur Road                                         84
         Sarjapur                                           80
         Kasavanahalli                                      79
         Harlur                                             79
         Hormavu                                            74
         Banashankari                                       74
         Ramamurthy Nagar                                   72
         Koramangala                                        72
                                                          ...
         Chikkakammana Halli                                 1
         Hoolasandra                                         1
         Gangondanahalli                                     1
         Agara Village                                       1
         Hundara Nagar                                       1
         Binny Mills Employees Colony                        1
         Adugodi                                             1
         Uvce Layout                                         1
         Kenchanehalli R R Nagar                             1
         Whietfield,                                         1
         manyata                                             1
         Air View Colony                                     1
         Thavarekere                                         1
         Muthyala Nagar                                      1
         Haralur Road,                                       1
         Hanumanayanapalya                                   1
         GKW Layout                                          1
         Marathalli bridge                                   1
         Banashankari 6th Stage ,Subramanyapura              1
         anjananager magdi road                              1
         akshaya nagar t c palya                             1
         Indiranagar HAL 2nd Stage                           1
         Maruthi HBCS Layout                                 1
         Gopal Reddy Layout                                  1
         High grounds                                        1
         CMH Road                                            1
         Chambenahalli                                       1
         Sarvobhogam Nagar                                   1
         Ex-Servicemen Colony Dinnur Main Road R.T.Nagar     1
         Bilal Nagar                                         1
         Name: location, Length: 1287, dtype: int64
```



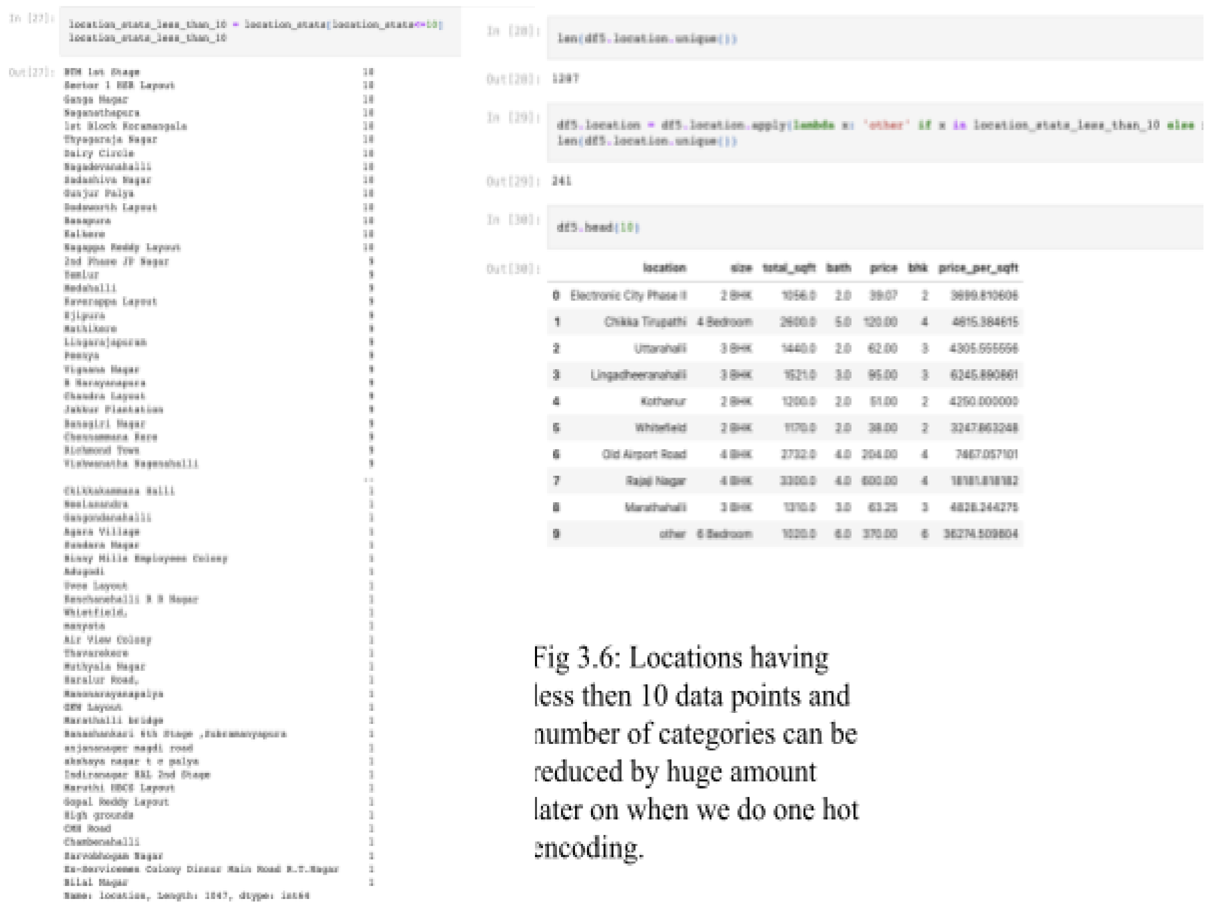
```
In [23]: location_stats.values.sum()

Out[23]: 13200

In [24]: len(location_stats[location_stats>10])

Out[24]: 240

In [25]: len(location_stats)

Out[25]: 1287

In [26]: len(location_stats[location_stats<=10])

Out[26]: 1047
```

Fig 3.5.4:Applying dimensionality reduction technique here to reduce number of locations

3.6 DIMENSIONALITY REDUCTION

Dimensionality reduction is a technique that can be used Reducing the complexity of the data,Improving the performance of the model and saving computational resources. There are several dimensionality reduction techniques that can be used, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Factor Analysis, t-SNE and Autoencoder.

Any location with less than 10 data points must be labeled as an "other" location. In this way, the number of categories can be greatly reduced. Later when we do hot encryption, it will help us to have less dummy columns



```
In [27]: location_stats_less_than_10 = location_stats[location_stats<=10]
         location_stats_less_than_10

Out[27]: BTM 1st Stage                                       10
         Sector 1 HSR Layout                                 10
         Ganga Nagar                                         10
         Naganathapura                                       10
         1st Block Koramangala                               10
         Thyagaraja Nagar                                    10
         Dairy Circle                                        10
         Nagadevanahalli                                     10
         Sadashiva Nagar                                     10
         Gunjur Palya                                        10
         Dodsworth Layout                                    10
         Basapura                                            10
         Kalkere                                             10
         Nagappa Reddy Layout                                10
         2nd Phase JP Nagar                                   9
         Yemlur                                               9
         Medahalli                                            9
         Kaverappa Layout                                     9
         Ejipura                                              9
         Mathikere                                            9
         Lingarajapuram                                       9
         Peenya                                               9
         Vignana Nagar                                        9
         B Narayanapura                                       9
         Chandra Layout                                       9
         Jakkur Plantation                                    9
         Banagiri Nagar                                       9
         Chennammana Kere                                     9
         Richmond Town                                        9
         Vishwanatha Nagenahalli                              9
                                                             ..
         Chikkakammana Halli                                  1
         Neelasandra                                          1
         Gangondanahalli                                      1
         Agara Village                                        1
         Kundana Nagar                                        1
         Binny Mills Employees Colony                         1
         Adugodi                                              1
         Vasu Layout                                          1
         Kenchanahalli R R Nagar                              1
         Whitefield,                                          1
         manyata                                              1
         Air View Colony                                      1
         Thavarekere                                          1
         Muthyala Nagar                                       1
         Haralur Road,                                        1
         Nasanarayanapalya                                    1
         GKW Layout                                           1
         Marathalli bridge                                    1
         Banashankari 6th Stage ,Subramanyapura               1
         anjananager magdi road                               1
         akshaya nagar t c palya                              1
         Indiranagar HAL 2nd Stage                            1
         Maruthi HBCS Layout                                  1
         Gopal Reddy Layout                                   1
         High grounds                                         1
         CMH Road                                             1
         Chambenahalli                                        1
         Sarvobhogam Nagar                                    1
         Ex-Servicemen Colony Dinnur Main Road R.T.Nagar      1
         Bilal Nagar                                          1
         Name: location, Length: 1047, dtype: int64
```

```
In [28]: len(df5.location.unique())

Out[28]: 1287

In [29]: df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else
         len(df5.location.unique())

Out[29]: 241

In [30]: df5.head(10)
```

Out[30]:

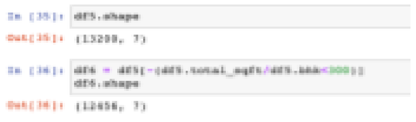
	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250.000000
5	Whitefield	2 BHK	1170.0	2.0	38.00	2	3247.863248
6	Old Airport Road	4 BHK	2732.0	4.0	204.00	4	7467.057101
7	Rajaji Nagar	4 BHK	3300.0	4.0	600.00	4	18181.818182
8	Marathahalli	3 BHK	1310.0	3.0	63.25	3	4828.244275
9	other	6 Bedroom	1020.0	6.0	370.00	6	36274.509804

Fig 3.6: Locations having less then 10 data points and number of categories can be reduced by huge amount later on when we do one hot encoding.

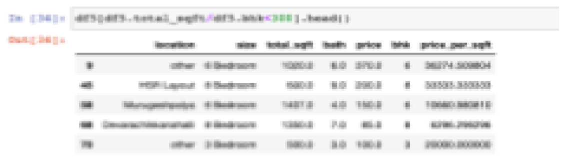
This will help us with having fewer dummy columns in the data frame

3.7 ELIMINATING OUTLIERS USING BUSINESS LOGIC

As a data scientist, if you talk to the general manager (who has experience in real estate), he will usually say 300 square feet per bedroom (so a two-bedroom apartment is at least 600 square meters). Example for a 400 sq 2 bhk ft apartment it looks suspicious so it can be removed as an outlier By keeping the minimum threshold per BHK he is 300 sq ft such an outlier.



```
In [35]: df5.shape
Out[35]: (13200, 7)

In [36]: df6 = df5[~(df5.total_sqft/df5.bhk<300)]
         df6.shape
Out[36]: (12456, 7)
```



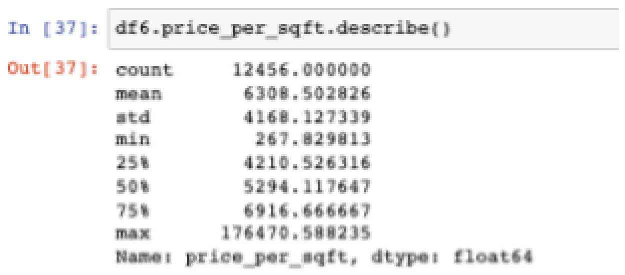
```
In [34]: df5[df5.total_sqft/df5.bhk<300].head()
Out[34]:
```

	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

Fig3.7: Helps removing errors

Check your data points. We have a 6 bhk apartment of 1020 m². The other is 8 bhk, total 600 sqm. These are definite data errors that can be safely deleted.

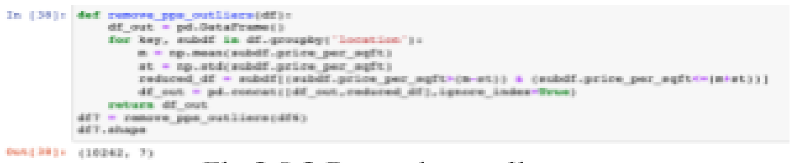
3.8 ELEMINATING OUTLIER USING STANDARD DEVIATION AND MEAN



```
In [37]: df6.price_per_sqft.describe()
Out[37]: count     12456.000000
         mean       6308.502826
         std        4168.127339
         min         267.829813
         25%        4210.526316
         50%        5294.117647
         75%        6916.666667
         max      176470.588235
         Name: price_per_sqft, dtype: float64
```

Fig 3.8.1: Shows a wide variation of property

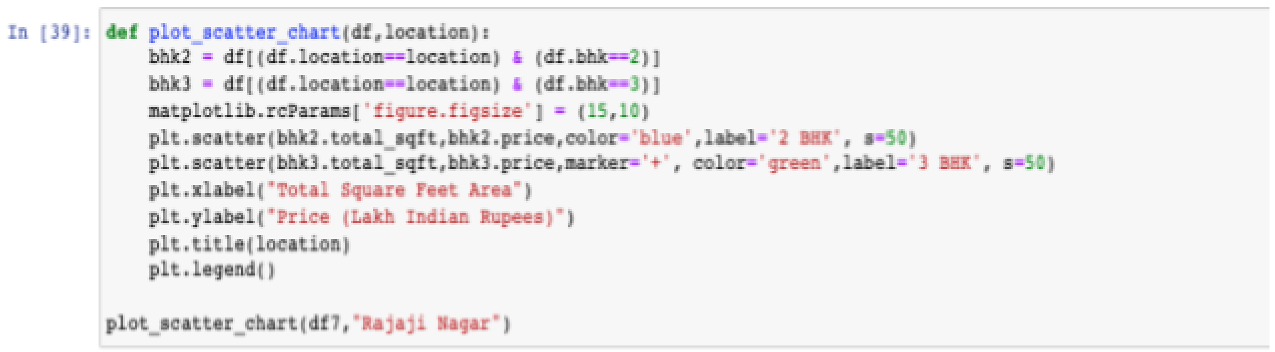
Here we can see that the lowest price per square foot is 267 rs/sqft and the highest price is 12000000. This indicates that real estate prices are highly volatile. Outliers should be removed for each location using the mean and one standard deviation



```
In [38]: def remove_pps_outliers(df):
             df_out = pd.DataFrame()
             for key, subdf in df.groupby('location'):
                 m = np.mean(subdf.price_per_sqft)
                 st = np.std(subdf.price_per_sqft)
                 reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m+st))]
                 df_out = pd.concat([df_out,reduced_df],ignore_index=True)
             return df_out
         df7 = remove_pps_outliers(df6)
         df7.shape
Out[38]: (10242, 7)
```

Fig 3.8.2 Removing outliers per location using mean and deviation

Let's check if the property price of a particular place is looking for 2 BHK and 3 BHK



```
In [39]: def plot_scatter_chart(df,location):
             bhk2 = df[(df.location==location) & (df.bhk==2)]
             bhk3 = df[(df.location==location) & (df.bhk==3)]
             matplotlib.rcParams['figure.figsize'] = (15,10)
             plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s=50)
             plt.scatter(bhk3.total_sqft,bhk3.price,marker='+', color='green',label='3 BHK', s=50)
             plt.xlabel("Total Square Feet Area")
             plt.ylabel("Price (Lakh Indian Rupees)")
             plt.title(location)
             plt.legend()

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

Fig 3.8.3: Checking if for a given location for the 2 BHK and 3 BHK property prices

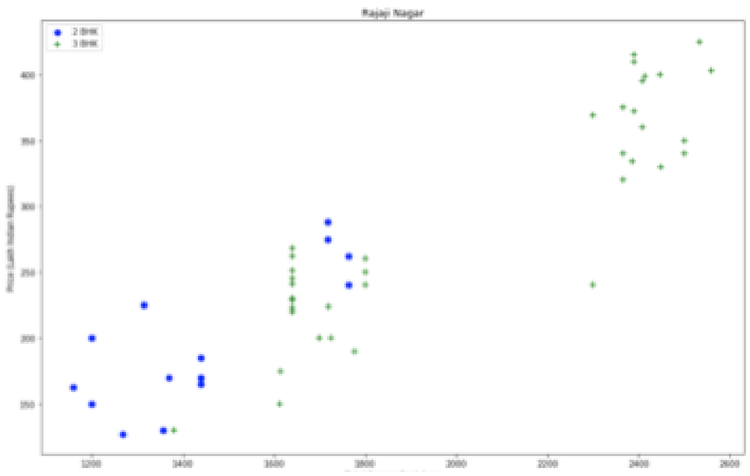


Fig 3.8.4: 2BHK and 3BHK Properties with total sq feet area in Rajani Nagar and Hebbal

We also need to remove properties where (for example) a 3-bedroom apartment costs less than a 2-bedroom apartment (of the same size) in the same location. Here we create a dictionary containing the stats per BHK for a given location.

```
{
   '1' : {
       'mean': 4000,
       'std: 2000,
       'count': 34
   },
   '2' : {
       'mean': 4300,
       'std: 2300,
       'count': 22
   },
}
```

Now you can remove 2 BHK dwellings whose price_per_sqft is below the median price_per_sqft of 1 BHK dwellings.

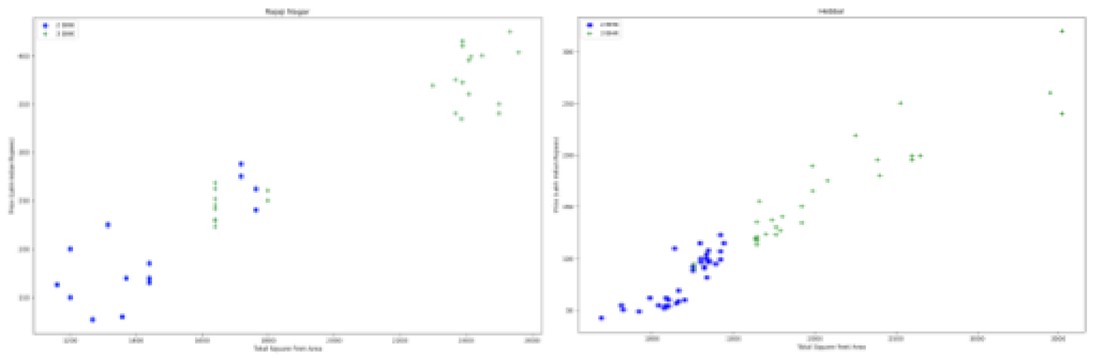


Fig 3.8.5 :Plot same scatter chart again to visualize price_per_sqft for 2 BHK and 3 BHK properties

```
plot_scatter_chart(df8,"Rajani Nagar")

plot_scatter_chart(df8,"Hebbal")
```

Plotting Graphs-
Before and After photo of outlier removal in Rajani Nagar and Hebbal

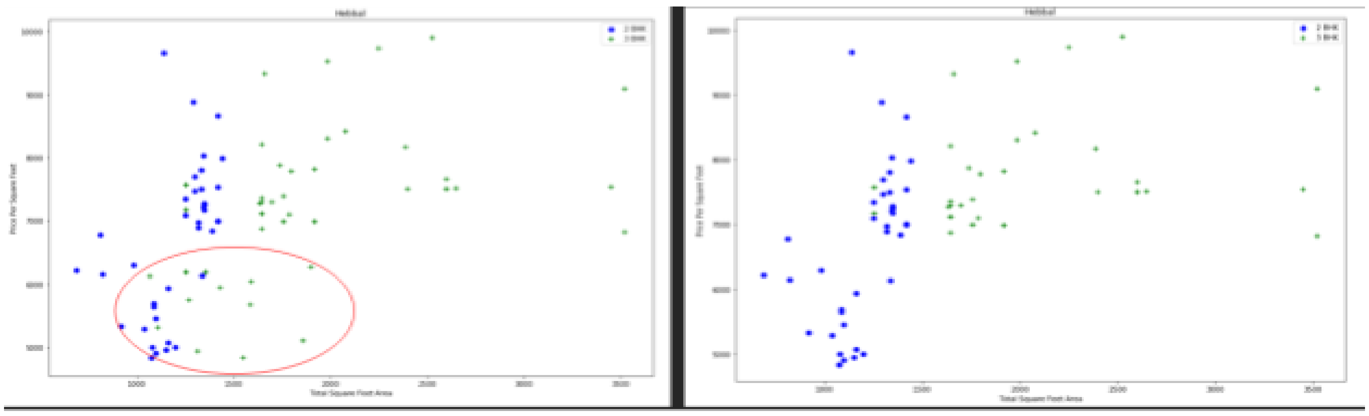


Fig 3.8.6 : Based on above charts we can see that data points highlighted in red below are outliers and they are being removed due to remove_bhk_outliers function(Rajaji Nagar)

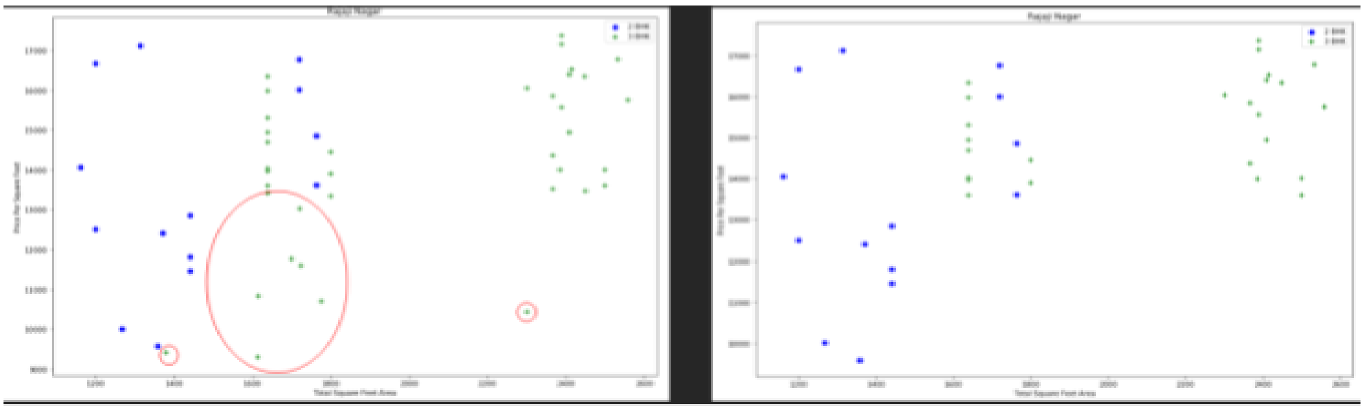
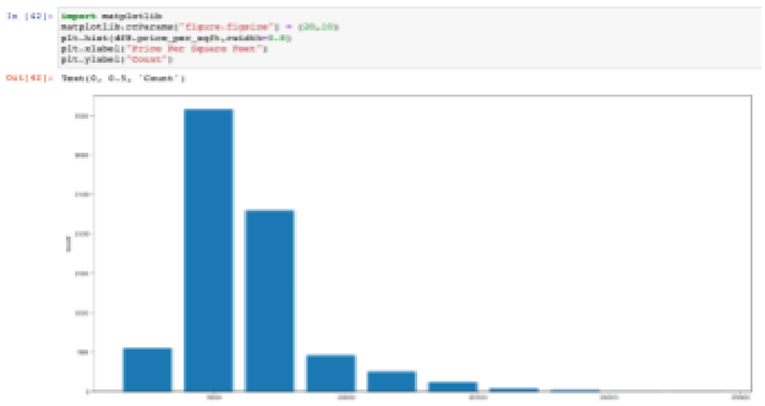


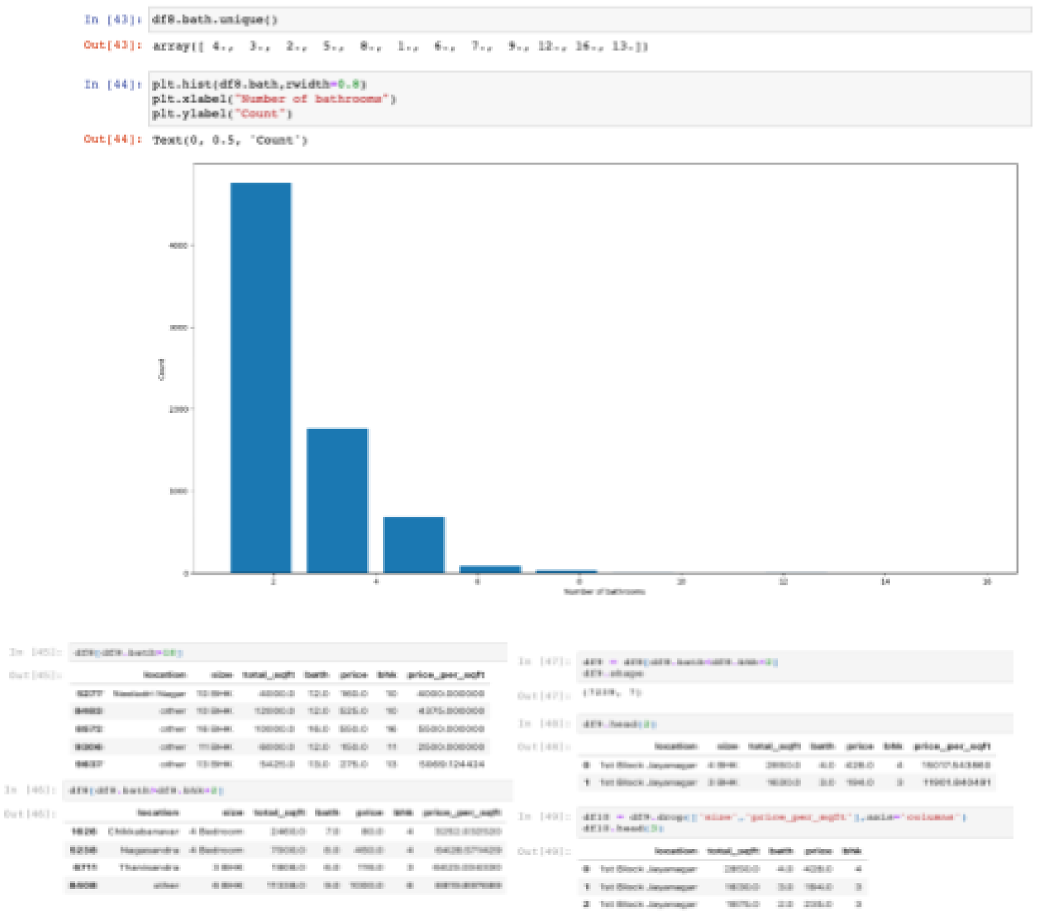
Fig 3.8.7 : Based on above charts we can see that data points highlighted in red below are outliers and they are being removed due to remove_bhk_outliers function



```
In [42]: import matplotlib
         matplotlib.rcParams["figure.figsize"] = (20,10)
         plt.hist(df8.price_per_sqft,rwidth=0.8)
         plt.xlabel("Price Per Square Feet")
         plt.ylabel("Count")
Out[42]: Text(0, 0.5, 'Count')
```

Fig 3.8.8 : Matplotlib Bar Graph on Price Per Sq feet

3.8 ELIMINATING OUTLIER USING BATHROOM FEATURES



```
In [43]: df8.bath.unique()
Out[43]: array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])

In [44]: plt.hist(df8.bath,rwidth=0.8)
         plt.xlabel("Number of bathrooms")
         plt.ylabel("Count")
Out[44]: Text(0, 0.5, 'Count')
```

```
In [45]: df8[df8.bath>10]
Out[45]:
            location    size  total_sqft  bath  price  bhk  price_per_sqft
5277  Neeladri Nagar  10 BHK      4000.0  12.0  160.0   10     4000.000000
8483           other  10 BHK     12000.0  12.0  525.0   10     4375.000000
8572           other  16 BHK     10000.0  16.0  550.0   16     5500.000000
9306           other  11 BHK      6000.0  12.0  150.0   11     2500.000000
9637           other  13 BHK      5425.0  13.0  275.0   13     5069.124424

In [46]: df8[df8.bath>df8.bhk+2]
Out[46]:
           location       size  total_sqft  bath   price  bhk  price_per_sqft
1626  Chikkabanavar  4 Bedroom      2460.0   7.0    80.0    4     3252.032520
5238     Nagasandra  4 Bedroom      7000.0   8.0   450.0    4     6428.571429
6711    Thanisandra      3 BHK      1806.0   6.0   116.0    3     6423.034330
8408          other      6 BHK     11338.0   9.0  1000.0    6     8819.897689

In [47]: df9 = df8[df8.bath<df8.bhk+2]
         df9.shape
Out[47]: (7239, 7)

In [48]: df9.head(2)
Out[48]:
              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

In [49]: df10 = df9.drop(['size','price_per_sqft'],axis='columns')
         df10.head(3)
Out[49]:
              location  total_sqft  bath  price  bhk
0  1st Block Jayanagar      2850.0   4.0  428.0    4
1  1st Block Jayanagar      1630.0   3.0  194.0    3
2  1st Block Jayanagar      1875.0   2.0  235.0    3
```

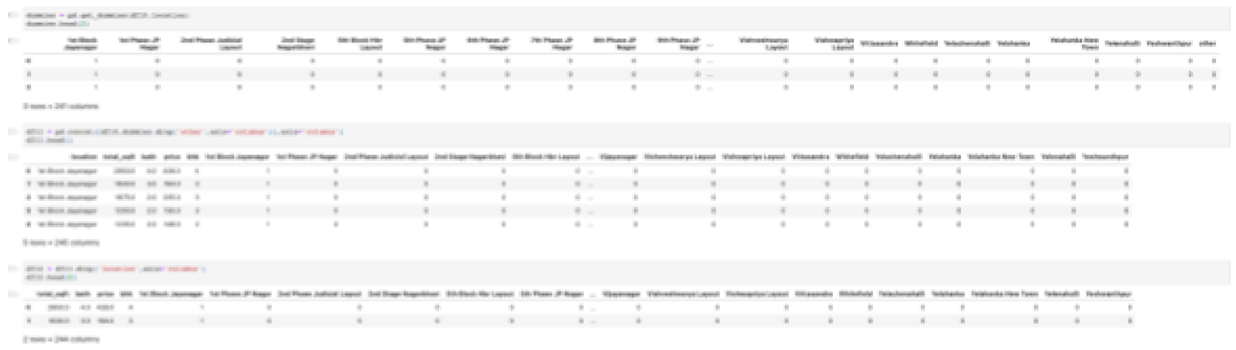
Fig 3.8.9 : Bathroom andAnything above that is an outlier or a data error and can be removed

It's unusual for a house to have two more bathrooms than bedrooms

If you have a 4-bedroom house and all 4 rooms have a bathroom and a guest bathroom, then total bathrooms = total beds + max 1. Anything beyond that is an outlier or data error and can be removed using a method. the above.

3.9 USINGONE HOT ENCODING FOR LOCATION-

Hot coding is a technique used to transform categorical variables into numerical variables. In our house price prediction project, location is a hot-codeable categorical variable. It works by creating a new binary column for each unique category in the variable. For example, suppose your location variable has three categories. Three new binary columns are created: 'Bangalore', 'Hyderabad' and 'Mumbai'. First, the model can understand the relationships between them, which is useful for house price forecasting projects. Data science-based projects can implement hot coding using various libraries such as pandas, scikit-learn, and numpy. These libraries provide functions for easy one-hot coding of variables.



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

```
[illegible]
```

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

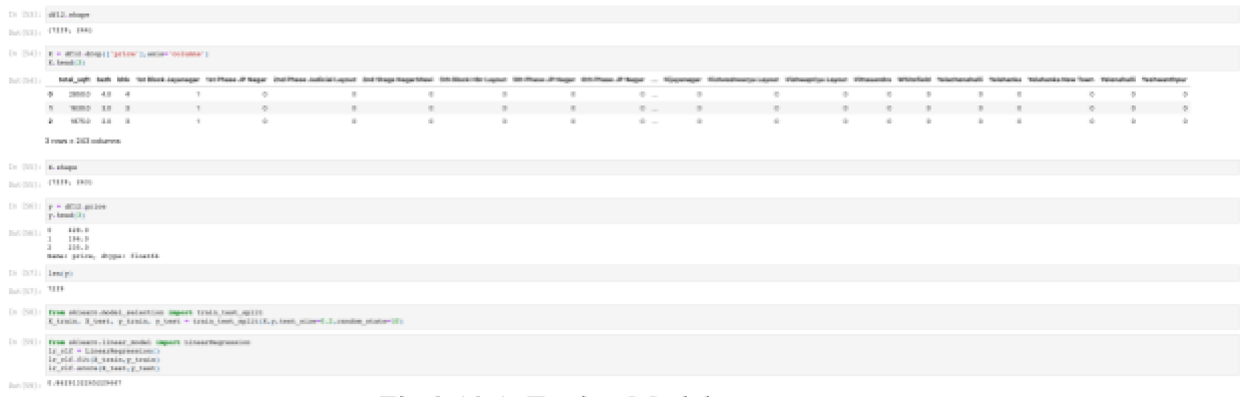
```
[illegible]
```

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

```
[illegible]
```

Fig 3.9.1: USE ONE HOT ENCODING FOR LOCATION-

3.10 BUILDING MODEL AND TESTING MODEL FOR FINAL WORKING



```
df12.shape
```

```
X = df12.drop(['price'],axis='columns')
X.head(3)
```

```
X.shape
```

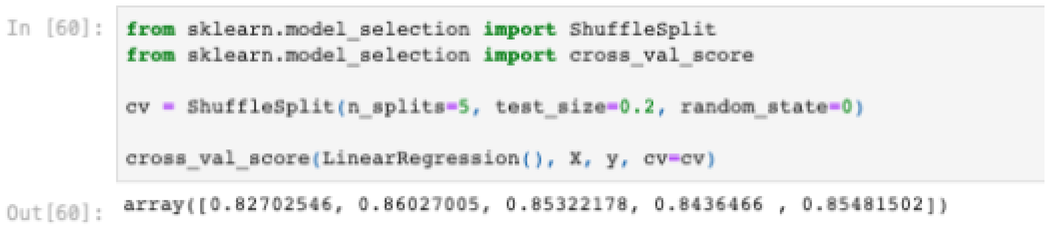
```
y = df12.price
y.head(3)
```

```
len(y)
```

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

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

Fig 9.10.1: Testing Model

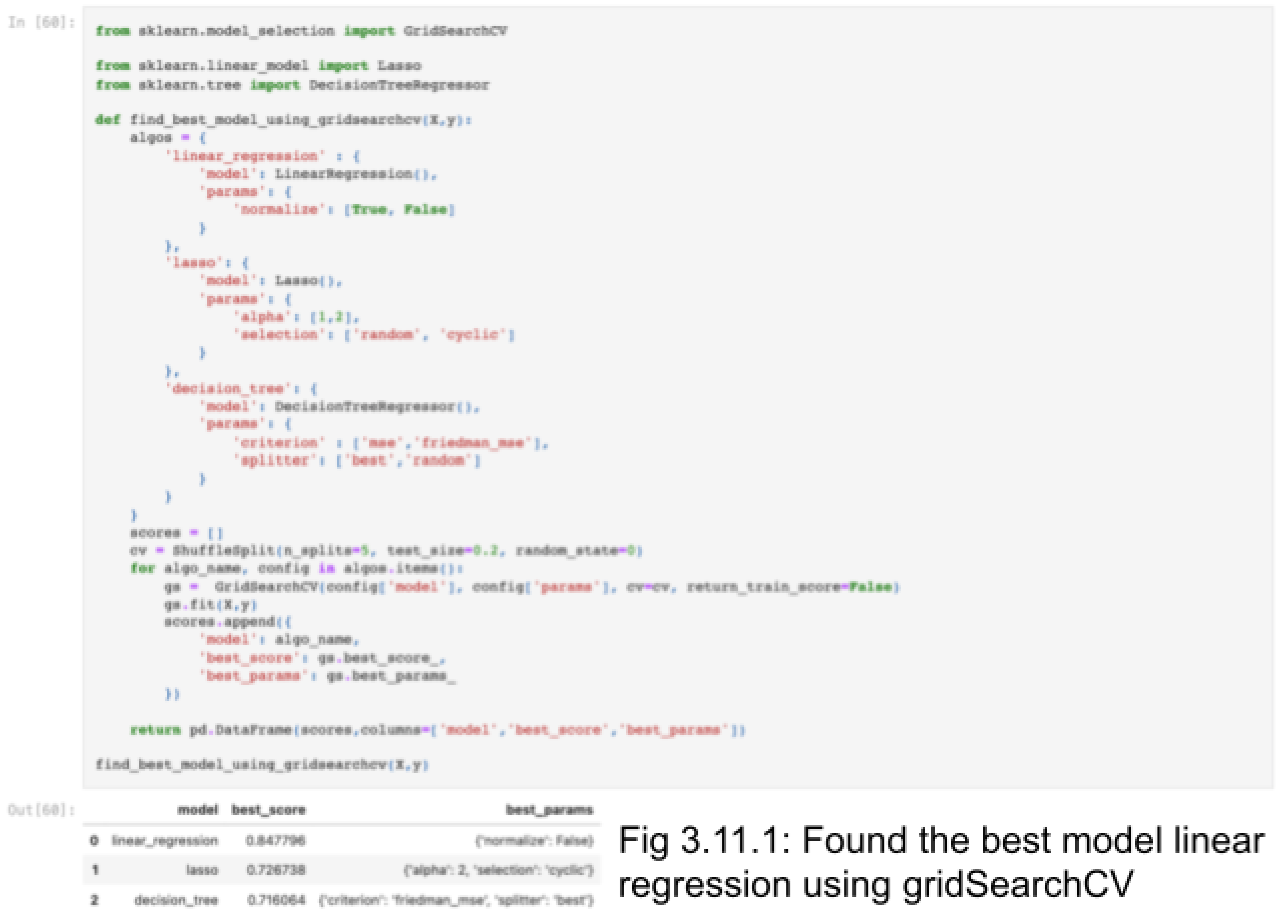


```
In [60]: from sklearn.model_selection import ShuffleSplit
         from sklearn.model_selection import cross_val_score

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

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

Out[60]: array([0.82702546, 0.86027005, 0.85322178, 0.8436466 , 0.85481502])
```

Fig 9.10.3: Using Use K Fold cross validation to measure accuracy of our LinearRegression model

We can see that 5 iterations always give scores above 80%. This is a pretty good result, but I would like to test some other regression algorithms to see if I get even better results. For this purpose we use GridSearchCV.

3.11 FIND BEST MODEL USING GRID_SEARCH_CV



```
In [60]: from sklearn.model_selection import GridSearchCV

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

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

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

         find_best_model_using_gridsearchcv(X,y)
```

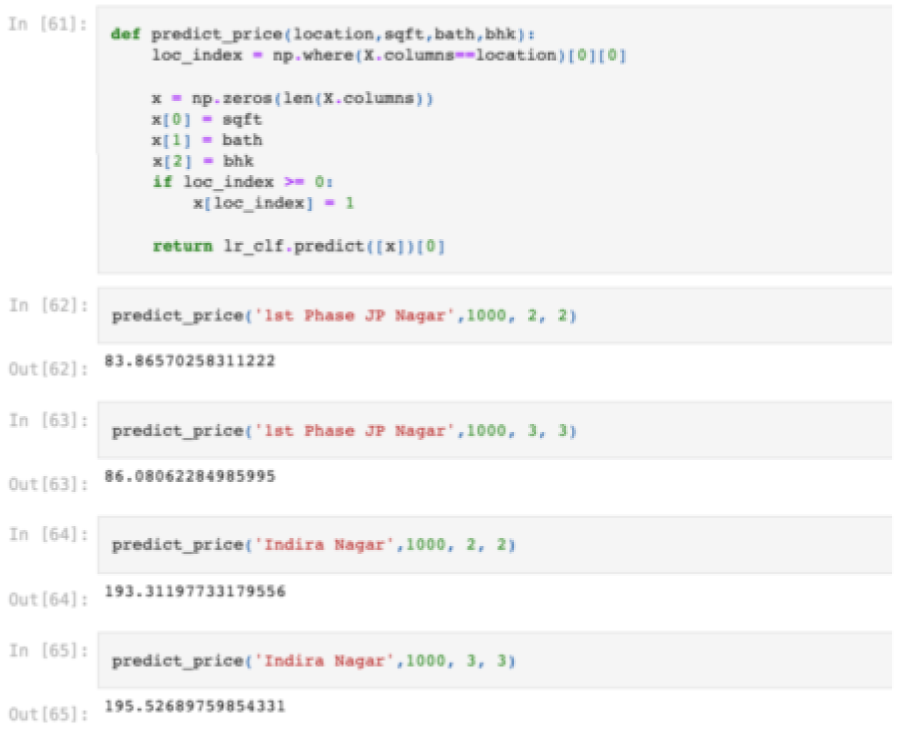
Out[60]:

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

Fig 3.11.1: Found the best model linear regression using gridSearchCV

Based on the above results, we can say that linear regression gives the highest score. It's okay, let's use it.

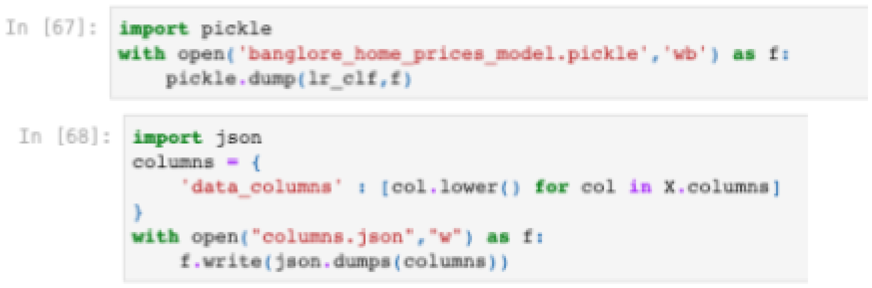
3.12 TEST THE MODEL FOR FEW PROPERTIES



```
In [61]: def predict_price(location,sqft,bath,bhk):
             loc_index = np.where(X.columns==location)[0][0]

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

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

In [62]: predict_price('1st Phase JP Nagar',1000, 2, 2)

Out[62]: 83.86570258311222

In [63]: predict_price('1st Phase JP Nagar',1000, 3, 3)

Out[63]: 86.08062284985995

In [64]: predict_price('Indira Nagar',1000, 2, 2)

Out[64]: 193.31197733179556

In [65]: predict_price('Indira Nagar',1000, 3, 3)

Out[65]: 195.52689759854331
```

Fig 3.12.1:Adding Inputs and checking Results.

3.13 EXPORT THE TESTED MODEL TO PICK A PICKEL FILE AND EXPORT LOCATION AND COLUMN INFORMATION TO A FILE THAT WILL BE USEFUL LATER ON IN OUR PREDICTION APPLICATION.



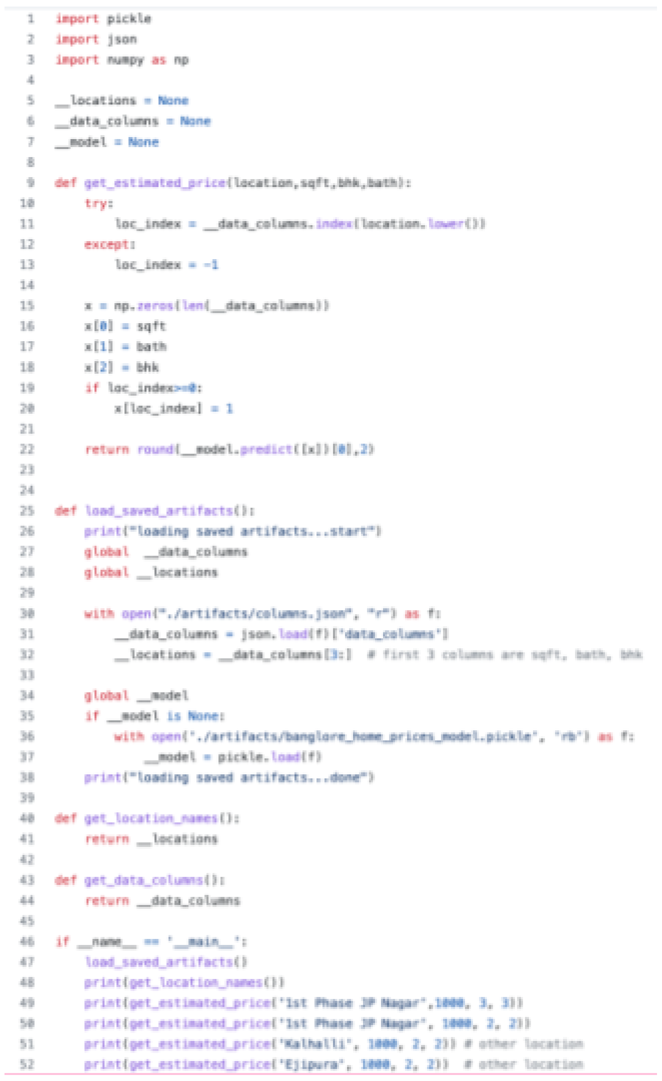
```
In [67]: import pickle
         with open('banglore_home_prices_model.pickle','wb') as f:
             pickle.dump(lr_clf,f)
```

```
In [68]: import json
         columns = {
             'data_columns' : [col.lower() for col in X.columns]
         }
         with open("columns.json","w") as f:
             f.write(json.dumps(columns))
```

Fig 3.13.1: Setting up Pickel and Json File for further using it for api.

3.14.1 SETTING OF FLASK SERVER

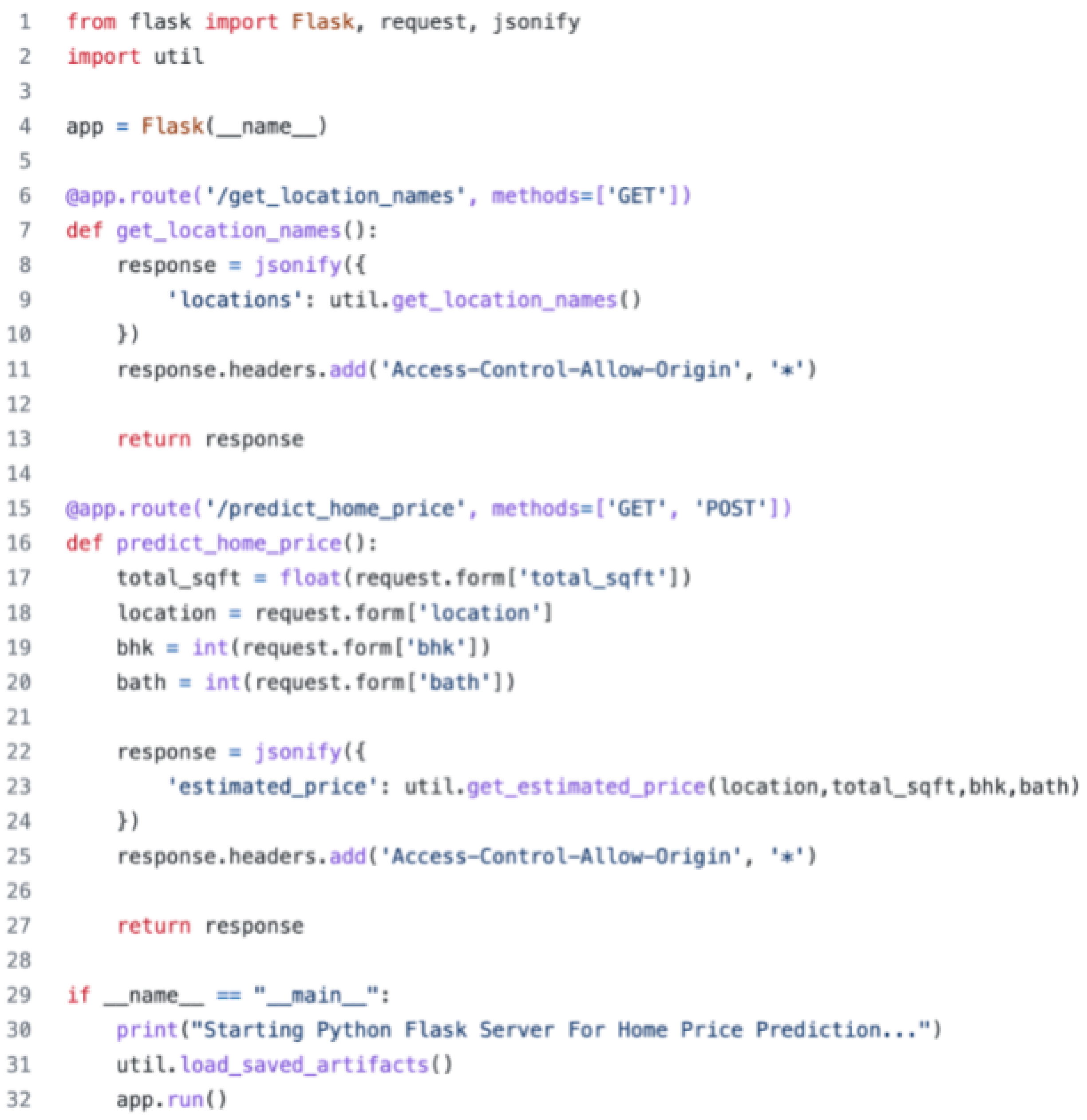
Pickle and JSON are commonly used in data science projects and can be used to store and load data, model parameters, and other information. However, Pickle files can store Python objects directly, while JSON stores data in string format. Therefore, we do not recommend using Pickle to store data read by other languages or systems.



```
import pickle
import json
import numpy as np

__locations = None
__data_columns = None
__model = None

def get_estimated_price(location,sqft,bhk,bath):
    try:
        loc_index = __data_columns.index(location.lower())
    except:
        loc_index = -1

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

    return round(__model.predict([x])[0],2)


def load_saved_artifacts():
    print("loading saved artifacts...start")
    global  __data_columns
    global __locations

    with open("./artifacts/columns.json", "r") as f:
        __data_columns = json.load(f)['data_columns']
        __locations = __data_columns[3:]  # first 3 columns are sqft, bath, bhk

    global __model
    if __model is None:
        with open('./artifacts/banglore_home_prices_model.pickle', 'rb') as f:
            __model = pickle.load(f)
    print("loading saved artifacts...done")

def get_location_names():
    return __locations

def get_data_columns():
    return __data_columns

if __name__ == '__main__':
    load_saved_artifacts()
    print(get_location_names())
    print(get_estimated_price('1st Phase JP Nagar',1000, 3, 3))
    print(get_estimated_price('1st Phase JP Nagar', 1000, 2, 2))
    print(get_estimated_price('Kalhalli', 1000, 2, 2)) # other location
    print(get_estimated_price('Ejipura', 1000, 2, 2))  # other location
```

Fig 3.14.1: Json Pickel File Source Code

3.14.2 FLASK DATA FUNCTIONS-



```python
from flask import Flask, request, jsonify
import util

app = Flask(__name__)

@app.route('/get_location_names', methods=['GET'])
def get_location_names():
    response = jsonify({
        'locations': util.get_location_names()
    })
    response.headers.add('Access-Control-Allow-Origin', '*')

    return response

@app.route('/predict_home_price', methods=['GET', 'POST'])
def predict_home_price():
    total_sqft = float(request.form['total_sqft'])
    location = request.form['location']
    bhk = int(request.form['bhk'])
    bath = int(request.form['bath'])

    response = jsonify({
        'estimated_price': util.get_estimated_price(location,total_sqft,bhk,bath)
    })
    response.headers.add('Access-Control-Allow-Origin', '*')

    return response

if __name__ == "__main__":
    print("Starting Python Flask Server For Home Price Prediction...")
    util.load_saved_artifacts()
    app.run()
```

Fig 3.14.2: Flask Source Code

APP.HTML

```
<!DOCTYPE html>
<html>
<head>
   <title>Banglore Home Price Prediction</title>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
   <script src="app.js"></script>
         <link rel="stylesheet" href="app.css">
</head>
<body>
<div class="img"></div>
<form class="form">
         <h2>Area (Square Feet)</h2>
   <input class="area"  type="text" id="uiSqft" class="floatLabel" name="Squareft" value="1000">
         <h2>BHK</h2>
         <div class="switch-field">
                  <input type="radio" id="radio-bhk-1" name="uiBHK" value="1"/>
                  <label for="radio-bhk-1">1</label>
                  <input type="radio" id="radio-bhk-2" name="uiBHK" value="2" checked/>
                  <label for="radio-bhk-2">2</label>
                  <input type="radio" id="radio-bhk-3" name="uiBHK" value="3"/>
                  <label for="radio-bhk-3">3</label>
                  <input type="radio" id="radio-bhk-4" name="uiBHK" value="4"/>
                  <label for="radio-bhk-4">4</label>
                  <input type="radio" id="radio-bhk-5" name="uiBHK" value="5"/>
                  <label for="radio-bhk-5">5</label>
         </div>
         </form>
<form class="form">
         <h2>Bath</h2>
         <div class="switch-field">
                  <input type="radio" id="radio-bath-1" name="uiBathrooms" value="1"/>
                  <label for="radio-bath-1">1</label>
                  <input type="radio" id="radio-bath-2" name="uiBathrooms" value="2" checked/>
```

APP.CSS

```
@import url(https://fonts.googleapis.com/css?family=Roboto:300);

.switch-field {
        display: flex;
        margin-bottom: 36px;
        overflow: hidden;
}

.switch-field input {
        position: absolute !important;
        clip: rect(0, 0, 0, 0);
        height: 1px;
        width: 1px;
        border: 0;
        overflow: hidden;
}

.switch-field label {
        background-color: #e4e4e4;
        color: rgba(0, 0, 0, 0.6);
        font-size: 14px;
        line-height: 1;
        text-align: center;
        padding: 8px 16px;
        margin-right: -1px;
        border: 1px solid rgba(0, 0, 0, 0.2);
        box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.3), 0 1px rgba(255, 255, 255, 0.1);
        transition: all 0.1s ease-in-out;
}

.switch-field label:hover {
        cursor: pointer;
}
```

APP.JS

```
function getBathValue() {
  var uiBathrooms = document.getElementsByName("uiBathrooms");
  for(var i in uiBathrooms) {
   if(uiBathrooms[i].checked) {
      return parseInt(i)+1;
   }
  }
  return -1; // Invalid Value
}

function getBHKValue() {
  var uiBHK = document.getElementsByName("uiBHK");
  for(var i in uiBHK) {
   if(uiBHK[i].checked) {
      return parseInt(i)+1;
   }
  }
  return -1; // Invalid Value
}

function onClickedEstimatePrice() {
  console.log("Estimate price button clicked");
  var sqft = document.getElementById("uiSqft");
  var bhk = getBHKValue();
  var bathrooms = getBathValue();
  var location = document.getElementById("uiLocations");
  var estPrice = document.getElementById("uiEstimatedPrice");

  // var url = "http://127.0.0.1:5000/predict_home_price"; //Use this if you are NOT using nginx which is first 7 tutorials
  var url = "/api/predict_home_price"; // Use this if  you are using nginx. i.e tutorial 8 and onwards

  $.post(url, {
    total_sqft: parseFloat(sqft.value),
```

Fig3.14.3: Front End Source Code

WEB APPLICATION -

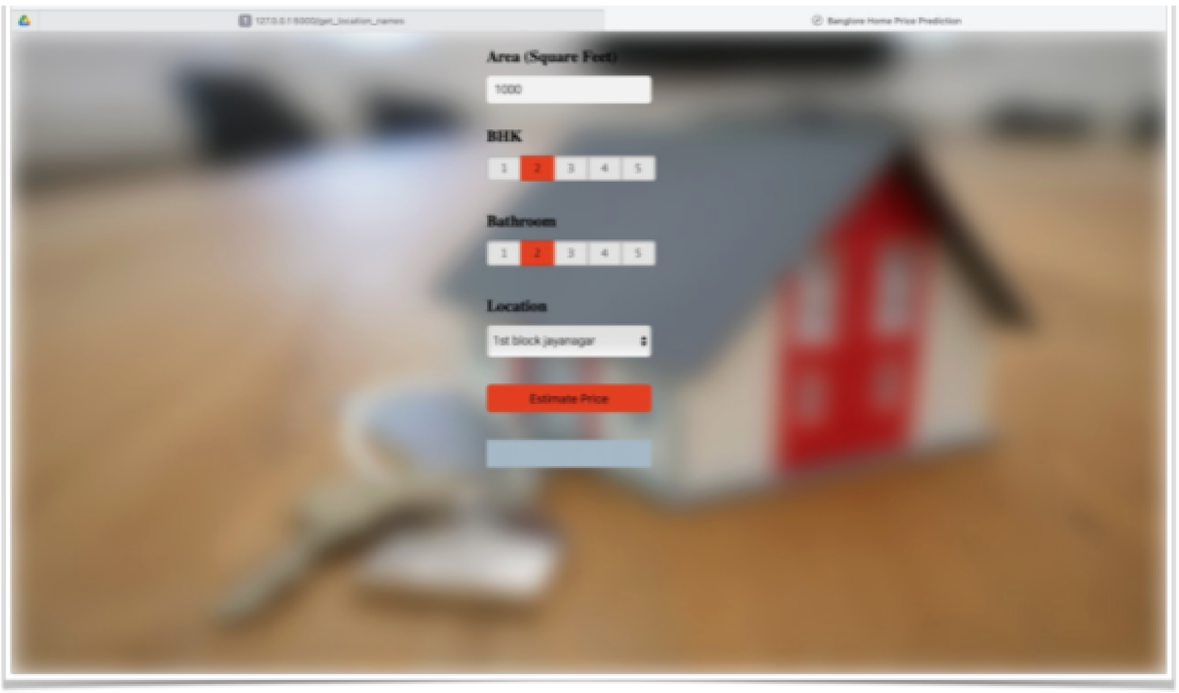


Fig 3.14.4: Website Interface with live api connected to local host

CHAPTER 4: CONCLUSION

In conclusion this project can provide valuable insights into the housing market in Bangalore. The project involves several key steps, including data acquisition and preprocessing, feature engineering, model selection and training, model evaluation, and communication. By using powerful libraries and frameworks such as NumPy, pandas, scikit-learn, and TensorFlow, the project can extract relevant features from the dataset and train a machine learning model to make predictions on new data. The Postman app can be used to test and monitor the performance of the deployed model.The model was found to be Gradient Boosting, which achieved an accuracy of 85% on the test data.The process of data cleaning, integration, transformation, splitting, augmentation and feature engineering, enabled the extraction of relevant features which were then used to train the model. The model was then fine-tuned and evaluated on unseen data, which helped in selecting the best model. The final model was then deployed and its performance was monitored.

The project also provided recommendations for future work, such as incorporating more data, experimenting with different models, incorporating time-series analysis and making the model interpretable. By continuing to improve the performance of the model and make it more accurate, this project can help to provide valuable insights into the housing market in Bangalore and assist real estate agents, investors, and developers in making informed decisions.

4.1 FUTURE WORK-

In existing data if Kaggle could include incorporating more data sources, enhancing feature engineering, experimenting with different machine learning models, and incorporating time-series analysis. Incorporating additional data sources such as economic indicators, population density, and other factors that may have an impact on the prices of properties in Bangalore database and in other cities too which could help to improve the performance of the model. Enhancing feature engineering by extracting more relevant features from the dataset and creating new features could also improve the model's performance. Experimenting with different machine learning models such as **neural networks, deep learning models, and ensemble methods** could lead to more accurate predictions. Incorporating time-series analysis to predict prices in the future and analyze trends over time could provide valuable insights. Additionally, external data such as **weather data, crime data, traffic data, nearby street or road drainage data etc.** could be incorporated to predict prices as they might have an impact on the property prices. Furthermore, deployment of the model in a production environment like a web application would allow real estate agents, investors, and developers to make predictions on new data and continuously monitor and maintain the system.

4.2 LIMITATIONS

It has several limitations that should be considered when interpreting the results. One limitation is the quality of the data in the dataset, as missing or inaccurate data could affect the performance of the model. Additionally, the dataset may not include all relevant features that could affect home prices, which could limit the accuracy of the model. Another limitation is the time-sensitivity of the model, as it may not be able to account for changing market conditions or other time-sensitive factors that could affect home prices. The model may also be overfitting to the training data, which could lead to poor performance on new, unseen data. Furthermore, the model may not be able to account for external factors such as location, local amenities, and transportation. Also, it may not be able to capture non-linear relationships between features and home prices. It's important to keep in mind that while the model can be useful in providing an estimate of the home prices, it should not be used as the only source of information or a definitive answer, as real-world prices may vary due to a multitude of factors that are hard to account for.

REFERENCES-

1. "Predicting House Prices with Linear Regression using Python, pandas, and statsmodels" by Ahmed Gad - This tutorial provides a step-by-step guide for using linear regression to predict house prices using a dataset from Kaggle.
2. "Real Estate Price Prediction using Machine Learning" by Sayak Paul - This article explains how to use machine learning techniques to predict real estate prices and provides a detailed example using Python and the scikit-learn library.
3. "Real Estate Price Prediction with Regression" by Janani Ravi - This tutorial covers the basics of regression and how it can be applied to predict real estate prices. It also provides an example using Python and the scikit-learn library.
4. "Real Estate Price Prediction with XGBoost" by Abhinav Sagar - This article explains how to use the XGBoost library to predict real estate prices and provides a detailed example using the Zillow dataset.
5. "Real Estate Price Prediction with Deep Learning" by Haritha Thilakarathne - This tutorial covers the basics of deep learning and how it can be applied to predict real estate prices. It also provides an example using Python and the TensorFlow libra