MINI-PROJECT

(2020-21)

HOUSE PRICE PREDICTION USING MACHINE LEARNING

PROJECT REPORT

DEPARTMENT OF COMPUTER ENGINEERING & APPLICATIONS

INSTITUE OF ENGINEERING AND TECHNOLOGY

Team Members:

Shivam Tripathi

(181500675)

Shubham Singh

(181500698)

Supervised By:

Mr. Vinay Agrawal

Assistant Professor

Department of Computer Engineering and ApplicationsGLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,Mathura – 281406

Declaration

We hereby declare that the work which is being presented in the Mini Project **"House Price Prediction",** in partial fulfillment of the requirements for Mini Project Lab is an authentic record of my own work carried under the supervision of **Mr. Vinay Agrawal, Assistant Professor.**

Shivam Tripathi

Shubham Singh

Department of Computer Engineering and ApplicationsGLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,Mathura – 281406

Certificate

This is to certify that the project entitled "**House Price Prediction**" carried out in Mini Project – II Lab is a bonafide work done by **Shivam Tripathi(181500675) and Shubham Singh(181500698)** and is submitted in partial fulfillment of the requirements for the award of the degree **Bachelor of Technology (Computer Science & Engineering).**

Signature of Supervisor:

Name of Supervisor: Mr. Vinay Agrawal

Acknowledgement

It gives us a great sense of pleasure to present the report of the B. Tech Mini Project undertaken during B. Tech. Third Year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that we have received.

We owe a special debt of gratitude to Mr. Vinay Agrawal, Assistant Professor, for his constant support and guidance throughout the course of the work. His sincerity, thoroughness and perseverance have been a constant source of inspiration. He has showered with all his extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught about the latest industry-oriented technologies.

We would not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the developmentof the project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Shivam Tripathi

Shubham Singh

Abstract

House price forecasting is an important topic of real estate. The literature attempts to derive useful knowledge from historical data of property markets. Machine learning techniques are applied to analyze historical property transactions in Banglore to discover useful models for house buyers and sellers. Revealed is the high discrepancy between house prices in the most expensive and most affordable suburbs in the city of Banglore. Moreover, experiments demonstrate that the Multiple Linear Regression that is based on mean squared error measurement is a competitive approach.

Table of Contents

Introduction

Housing prices are an important reflection of the economy, and housing
price ranges are of great interest for both buyers and sellers. In this project. house prices will be predicted given explanatory variables that cover many aspects of residential houses. As continuous house prices, they will be predicted with various regression techniques as individual price ranges,they will be predicted with classification methods.
The goal of this project is to create a regression model and a classification model that are able to accurately estimate the price of the house.

Software Requirement Analysis

This data science project walks through the step by step process of how to build a real estate price prediction website. We will first build a model using sklearn and linear regression using Banglore home prices dataset from kaggle.com. Second step would be to write a python flask server that uses the saved model to serve HTTP requests. Third component is the website built in HTML, CSS and JavaScript that allows users to enter home square ft area, bedrooms etc and it will call a python flask server to retrieve the predicted price. During model building we will cover almost all data science concepts such as data load and cleaning, outlier detection and removal, feature engineering, dimensionality reduction, grid searchcv for hyper parameter tuning, k fold cross validation etc.

Development Tools Used:

Python Programming:

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.
Python is a programming language that lets you work quickly and integrate systems more efficiently.
Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Numpy and Pandas:

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc.

Operations using NumPy:

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Matplotlib:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

Sklearn:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Installation:

If you already installed NumPy and Scipy, following are the two easiest ways to install scikit-learn −

Using pip

pip install -U scikit-learn

Using conda

conda install scikit-learn

Features:

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows −

- **Supervised Learning algorithms** − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.
- **Unsupervised Learning algorithms** − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.
- **Clustering** − This model is used for grouping unlabeled data.
- **Cross Validation** − It is used to check the accuracy of supervised models on unseen data.
- **Dimensionality Reduction** − It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.

- **Ensemble methods** − As name suggest, it is used for combining the predictions of multiple supervised models.
- **Feature extraction** − It is used to extract the features from data to define the attributes in image and text data.
- **Feature selection** − It is used to identify useful attributes to create supervised models.
- **Open Source** − It is open source library and also commercially usable under BSD license.

Python Flask:

Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by Armin Ronacher who leads an international group of python enthusiasts (POCCO). It is based on WSGI toolkit and jinja2 template engine. Flask is considered as a micro framework.

Tools used:

HTML:

Hypertext Markup Language (**HTML**) is the standard markup language for creating web pages and web application. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.

Web Browser receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML Elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <img/> and <input/>directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

HTML code ensures the proper formatting of text and images so that your Internet browsermay display them as they are intended to look. Without HTML, a browser would not know how to display text as elements or load images or other elements. HTML also provides a basic structure of the page, upon which Cascading Style Sheets are overlaid to change its appearance. One could think of HTML as the bones (structure) of a web page, and CSS as its skin (appearance).

CSS (Cascading Style Sheets)

Cascading **S**tyle **S**heets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs,variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

Advantage Of CSS:

- **CSS saves time** − You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** − If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
- **Easy maintenance** − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** − CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

- **Multiple Device Compatibility** − Style sheets allow content to be optimized for more than one type of device. By using the same HTML document,
- different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** − Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

JAVA SCRIPT (JS)

JavaScript is a high-level, interpreted programming language that conforms to the ECMAScript specification. It is a programming language that is characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

The terms *Vanilla JavaScript* and *Vanilla JS* refer to JavaScript not extended by any frameworks or additional libraries. Scripts written in Vanilla JS are plain JavaScript code.

Although there are similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design. JavaScript was influenced by programming languages such as Self and Scheme.

Jupyter Notebook

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell. To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library[is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

VISUAL STUDIO :

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source control systems (like Subversion and Git) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C,[6] C++, C++/CLI, Visual Basic .NET, C#, F#,JavaScript, TypeScript, XML, XSLT, HTML, and CSS. Support for other languages such as Python,Ruby, Node.js, and M among others is available via plug-ins. Java (and J#) were supported in the past.

Pycharm:

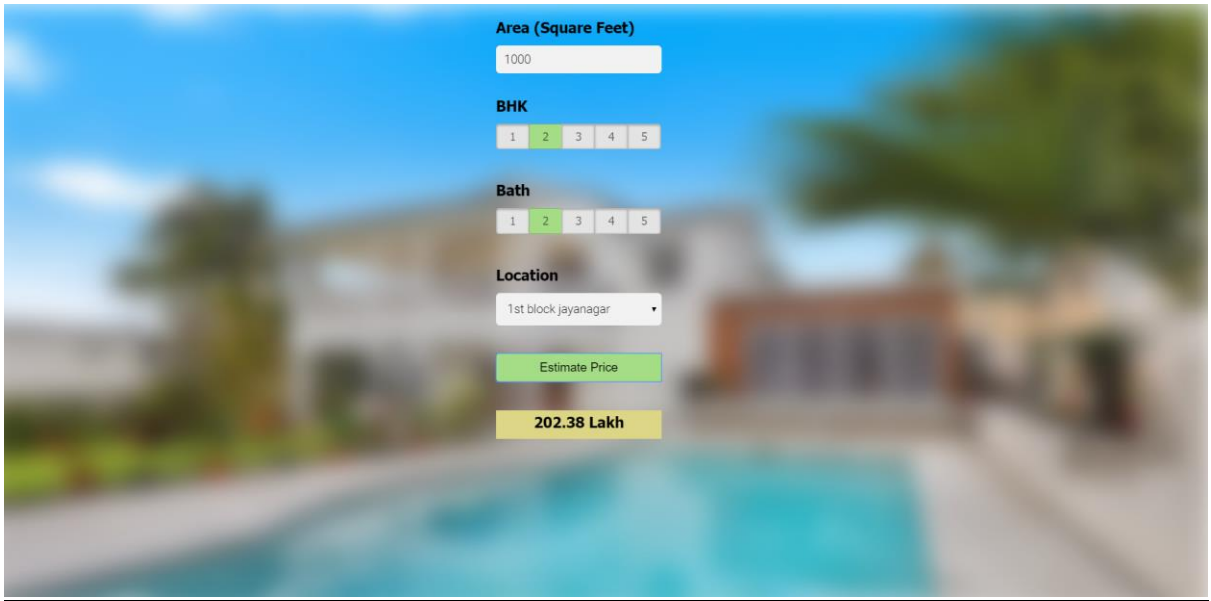
PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as datascience with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition withextra features – released under a proprietary license**.**

Features of Pycharm:

- Coding assistance and analysis, with code completion , syntax and error highlighting,linter integration, and quick fixes
- Project and code navigation: specialized project views, file structure views and quickjumping between files, classes, methods and usages
- Python refactoring: includes rename, extract method, introduce variable, introduceconstant, pull up, push down and others
- Support for web frameworks: Django and Flask
- Integrated Python debugger
- Integrated unit testing, with line-by-line code coverage.
- Version control integration: unified user interface
- Support for scientific tools like matplotlib, numpy and scipy

Software Design



Implementation Details:

This data science project walks through the step by step process of how to build a real estate price prediction website. We will first build a model using sklearn and linear regression using Banglore home prices dataset from kaggle.com.
Second step would be to write a python flask server that uses the saved model to serve HTTP requests.
Third component is the website built in HTML, CSS and JavaScript that allows users to enter home square ft area, bedrooms etc and it will call a python flask server to retrieve the predicted price.
During model building we will cover almost all data science concepts such as data load and cleaning, outlier detection and removal, feature engineering, dimensionality reduction, grid searchcv for hyper parameter tuning, k fold cross validation etc.

Code:

Real Estate Price Prediction Using Machine Learning Algorithms (1).ipynb

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
```

In [2]:
```
housing1 = pd.read_csv("Real_Estate_Price_Prediction_Dataset.csv")
```

In [3]:
```
housing1.head()
```

Out[3]:

	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 [4]:
```
housing1.shape
```

Out[4]:
```
(13320, 9)
```

In [5]:
```
housing1.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
```

```
In [6]: housing1.groupby('area_type')['area_type'].agg('count')
Out[6]: area_type
        Built-up  Area          2418
        Carpet  Area            87
        Plot  Area              2025
        Super built-up  Area    8790
        Name: area_type, dtype: int64

In [7]: housing2 = housing1.drop(['area_type','society','balcony','availability'],axis='columns')

In [8]: housing2.head()
```

Out[8]:

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

```
In [9]: housing2.isnull().sum()
Out[9]: location       1
        size          16
        total_sqft     0
        bath          73
        price          0
        dtype: int64

In [10]: housing3 = housing2.dropna()

In [11]: housing3.isnull().sum()
Out[11]: location      0
         size          0
         total_sqft    0
         bath          0
         price         0
         dtype: int64

In [12]: housing3.shape
Out[12]: (13246, 5)

In [13]: housing3['size'].unique()
Out[13]: 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)
```

Feature Engineering

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

```
In [14]: housing3['bhk']=housing3['size'].apply(lambda x: int(x.split(' ')[0]))
<ipython-input-14-4955814f9b3c>: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: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  housing3['bhk']=housing3['size'].apply(lambda x: int(x.split(' ')[0]))

In [15]: housing3.head()
```

Out[15]:

	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 [16]: housing3['bhk'].unique()
Out[16]: array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
                13, 18], dtype=int64)

In [17]: housing3[housing3.bhk>20]
```

Out[17]:

	location	size	total_sqft	bath	price	bhk
1718	2Electronic City Phase II	27 BHK	8000	27.0	230.0	27
4684	Munnekollal	43 Bedroom	2400	40.0	660.0	43

```
In [18]: housing3.total_sqft.unique()
Out[18]: array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
               dtype=object)
```

Explore total_sqft feature

```
In [19]: def is_float(x):
             try:
                 float(x)
             except:
                 return False
             return True
```

```
In [20]: housing3[~housing3['total_sqft'].apply(is_float)].head(10)
```

Out[20]:

	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

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 simpl

```
In [21]: 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 [22]: housing4 = housing3.copy()
         housing4.total_sqft = housing4.total_sqft.apply(convert_sqft_to_num)
         housing4 = housing4[housing4.total_sqft.notnull()]
         housing4 = housing4[housing4.total_sqft.notnull()]
         housing4.head(3)
```

Out[22]:

	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

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

```
In [23]: housing4.loc[30]
```

```
Out[23]: location      Yelahanka
         size              4 BHK
         total_sqft         2475
         bath                  4
         price               186
         bhk                   4
         Name: 30, dtype: object
```

```
In [24]: (2100+2850)/2
```

```
Out[24]: 2475.0
```

```
In [25]: housing4.head()
```

Out[25]:

	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

```
In [26]: housing5 = housing4.copy()
         housing5['price_per_sqft'] = housing5['price']*100000/housing5['total_sqft']
         housing5.head()
```

Out[26]:

	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

Add new feature called price per square feet

```
In [27]: housing5_stats = housing5['price_per_sqft'].describe()
         housing5_stats
```

```
Out[27]: 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 [28]: housing5.to_csv("bhp.csv",index=False)
```

```
In [29]: housing5.location = housing5.location.apply(lambda x: x.strip())
         location_stats = housing5['location'].value_counts(ascending=False)
         location_stats
```

```
In [29]: housing5.location = housing5.location.apply(lambda x: x.strip())
         location_stats = housing5['location'].value_counts(ascending=False)
         location_stats

Out[29]: Whitefield                                    533
         Sarjapur  Road                                392
         Electronic City                              304
         Kanakpura Road                               264
         Thanisandra                                  235
                                                     ...
         Lakshminarayanapura, Electronic City Phase 2   1
         basaveshwarnagar                               1
         Goraguntepalya                                 1
         BAGUR ROAD                                     1
         Michael Palaya                                 1
         Name: location, Length: 1287, dtype: int64

In [30]: location_stats.values.sum()

Out[30]: 13200

In [31]: len(location_stats[location_stats>10])

Out[31]: 240

In [32]: len(location_stats)

Out[32]: 1287

In [33]: len(location_stats[location_stats<=10])

Out[33]: 1047
```

Dimensionality Reduction

```
In [34]: location_stats_less_than_10 = location_stats[location_stats<=10]
         location_stats_less_than_10

Out[34]: Thyagaraja Nagar                              10
         Ganga Nagar                                   10
         Naganathapura                                 10
         Basapura                                      10
         Nagappa Reddy Layout                          10
                                                       ..
         Lakshminarayanapura, Electronic City Phase 2   1
         basaveshwarnagar                               1
         Goraguntepalya                                 1
         BAGUR ROAD                                     1
         Michael Palaya                                 1
         Name: location, Length: 1047, dtype: int64

In [35]: len(housing5.location.unique())

Out[35]: 1287

In [36]: housing5.location = housing5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
         len(housing5.location.unique())

Out[36]: 241
```

```
In [37]: housing5.head(10)

Out[37]:
```

	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

Outlier Removal Using Business Logic

```
In [38]: housing5[housing5.total_sqft/housing5.bhk<300].head()

Out[38]:
```

	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 [39]: housing5.shape

Out[39]: (13200, 7)
```

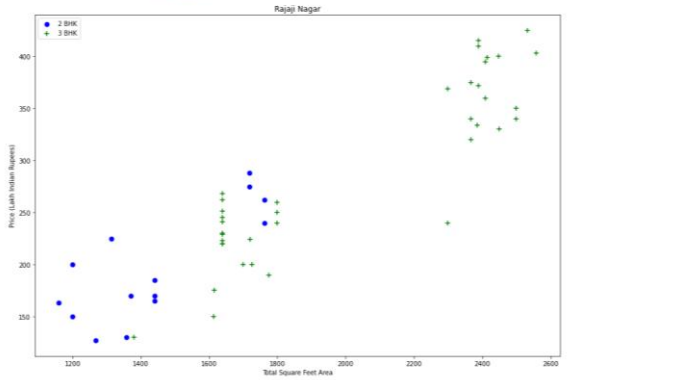
```
In [40]: housing6 = housing5[~(housing5.total_sqft/housing5.bhk<300)]
         housing6.shape
Out[40]: (12456, 7)
```

Outlier Removal Using Standard Deviation and Mean

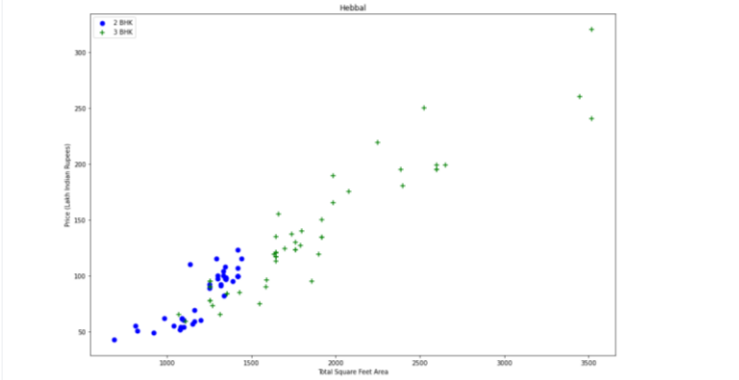
```
In [41]: housing6.price_per_sqft.describe()
Out[41]: count     12456.000000
         mean       6308.502826
         std        4168.127339
         min         267.829813
         25%        4210.526316
         50%        5294.117647
         75%        6916.666667
         max      176470.588235
         Name: price_per_sqft, dtype: float64
```

```
In [42]: 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
         housing7 = remove_pps_outliers(housing6)
         housing7.shape
Out[42]: (10242, 7)
```

```
In [43]: 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(housing7,"Rajaji Nagar")
```

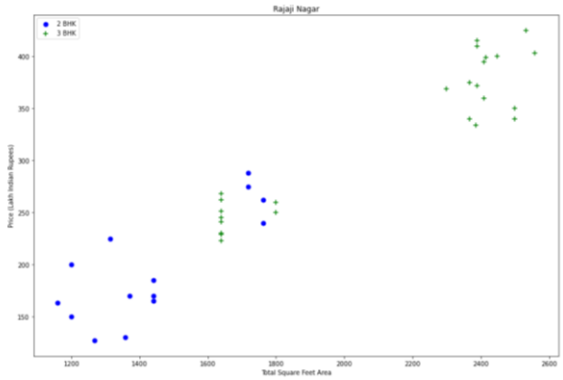



```
In [44]: plot_scatter_chart(housing7,"Hebbal")
```



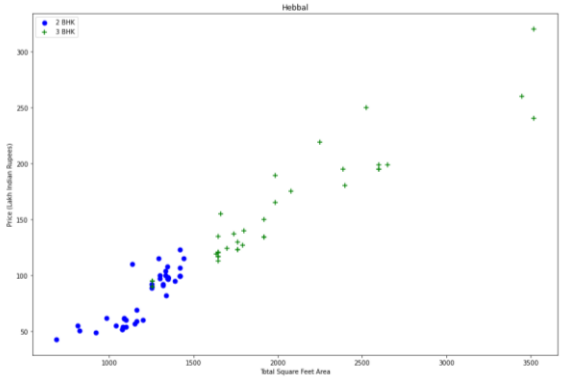
```
In [45]: def remove_bhk_outliers(df):
             exclude_indices = np.array([])
             for location, location_df in df.groupby('location'):
                 bhk_stats = {}
                 for bhk, bhk_df in location_df.groupby('bhk'):
                     bhk_stats[bhk] = {
                         '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.price_per_sqft<(stats['mean'])].index.values)
             return df.drop(exclude_indices,axis='index')
         housing8 = remove_bhk_outliers(housing7)
         # df8 = df7.copy()
         housing8.shape
Out[45]: (7317, 7)

In [46]: plot_scatter_chart(housing8,"Rajaji Nagar")
```

```
In [46]: plot_scatter_chart(housing8,"Rajaji Nagar")
```

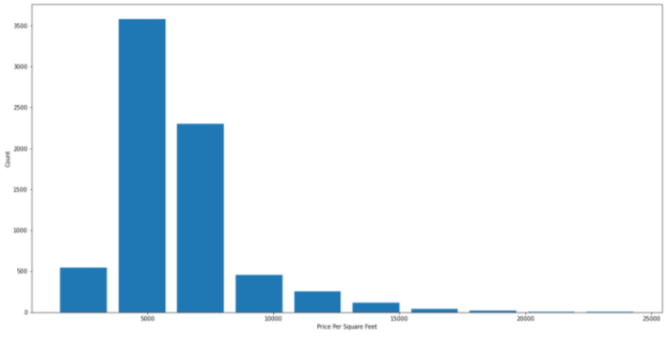


Total Square Feet Area

```
In [47]: plot_scatter_chart(housing8,"Hebbal")
```

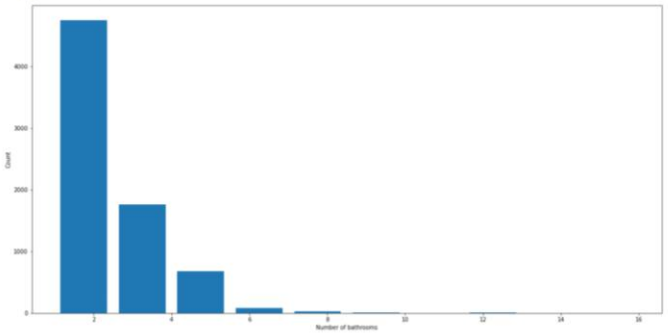



```
In [48]: import matplotlib
         matplotlib.rcParams["figure.figsize"] = (20,10)
         plt.hist(housing8.price_per_sqft,rwidth=0.8)
         plt.xlabel("Price Per Square Feet")
         plt.ylabel("Count")
Out[48]: Text(0, 0.5, 'Count')
```



Outlier Removal Using Bathrooms Feature

```
In [49]: housing8.bath.unique()
Out[49]: array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])

In [50]: plt.hist(housing8.bath,rwidth=0.8)
         plt.xlabel("Number of bathrooms")
         plt.ylabel("Count")
Out[50]: Text(0, 0.5, 'Count')
```




```
In [51]: housing8[housing8.bath>10]
```

Out[51]:

	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 [52]: housing8[housing8.bath>housing8.bhk+2]
```

Out[52]:

	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 [53]: housing9 = housing8[housing8.bath<housing8.bhk+2]
         housing9.shape
Out[53]: (7239, 7)

In [54]: housing9.head(2)
```

Out[54]:

	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 [55]: housing10 = housing9.drop(['size','price_per_sqft'],axis='columns')
         housing10.head(3)
```

Out[55]:

	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

Use One Hot Encoding For Location

```
In [56]: dummies = pd.get_dummies(housing10.location)
         dummies.head(3)
```

Out[56]:

	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	...	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield	Yelachenal
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

3 rows × 241 columns

```
In [57]: housing11 = pd.concat([housing10,dummies.drop('other',axis='columns')],axis='columns')
         housing11.head()
```

Out[57]:

	location	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefi
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	0	0	0	0	...	0	0	0	0	0
1	1st Block Jayanagar	1630.0	3.0	194.0	3	1	0	0	0	0	...	0	0	0	0	0
2	1st Block Jayanagar	1875.0	2.0	235.0	3	1	0	0	0	0	...	0	0	0	0	0
3	1st Block Jayanagar	1200.0	2.0	130.0	3	1	0	0	0	0	...	0	0	0	0	0
4	1st Block Jayanagar	1235.0	2.0	148.0	2	1	0	0	0	0	...	0	0	0	0	0

5 rows × 245 columns

```
In [58]: housing12 = housing11.drop('location',axis='columns')
         housing12.head(2)
```

Out[58]:

	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...	0	0	0	0	0
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...	0	0	0	0	0

2 rows × 244 columns

Model Building

```
In [59]: housing12.shape
```

Out[59]: (7239, 244)

```
In [60]: X = housing12.drop(['price'],axis='columns')
         X.head(3)
```

Out[60]:

	total_sqft	bath	bhk	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	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield
0	2850.0	4.0	4	1	0	0	0	0	0	0	...	0	0	0	0	0
1	1630.0	3.0	3	1	0	0	0	0	0	0	...	0	0	0	0	0
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	0	0	0	0	0

3 rows × 243 columns

```
In [61]: X.shape
```

Out[61]: (7239, 243)

```
In [62]: y = housing12.price
         y.head(3)
```

Out[62]:
```
0    428.0
1    194.0
2    235.0
Name: price, dtype: float64
```

```
In [63]: len(y)
```

Out[63]: 7239

```
In [64]: 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)
```

```
In [65]: from sklearn.linear_model import LinearRegression
         lr_clf = LinearRegression()
         lr_clf.fit(X_train,y_train)
         lr_clf.score(X_test,y_test)
```

Out[65]: 0.862913224522944 3

Use K Fold cross validation to measure accuracy of our LinearRegression model

```
In [66]: 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[66]: array([0.82702546, 0.86027005, 0.85322178, 0.8436466 , 0.85481502])

In [67]:
```
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[67]:

	model	best_score	best_params
0	linear_regression	0.847796	{'normalize': False}
1	lasso	0.726840	{'alpha': 2, 'selection': 'random'}
2	decision_tree	0.719350	{'criterion': 'friedman_mse', 'splitter': 'best'}

Based on above results we can say that LinearRegression gives the best score. Hence we will use that.

Test the model for few properties

In [68]:
```
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 [69]:
```
predict_price('1st Phase JP Nagar',1000, 2, 2)
```

Out[69]: 83.86570258312179

In [70]:
```
predict_price('1st Phase JP Nagar',1000, 3, 3)
```

Out[70]: 86.0806228498696

In [71]:
```
predict_price('Indira Nagar',1000, 2, 2)
```

Out[71]: 193.31197733179872

In [72]:
```
predict_price('Indira Nagar',1000, 3, 3)
```

Out[72]: 195.52680759854655

Export the tested model to a pickle file

In [73]:
```
import pickle
with open('banglore_home_prices_model.pickle','wb') as f:
    pickle.dump(lr_clf,f)
```

Export location and column information to a file that will be useful later on in our prediction application

In [74]:
```
import json
columns = {
    'data_columns' : [col.lower() for col in X.columns]
}
with open("columns.json","w") as f:
    f.write(json.dumps(columns))
```

In [ ]:
```

```

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/>
                <label for="radio-bath-
2">2</label>
                <input type="radio" id="radio-
bath-3" name="uiBathrooms" value="3"/>
                <label for="radio-bath-
3">3</label>
                <input type="radio" id="radio-
bath-4" name="uiBathrooms" value="4"/>
                <label for="radio-bath-
4">4</label>
```

```
                    <input type="radio" id="radio-bath-5" name="uiBathrooms" value="5"/>
                    <label for="radio-bath-5">5</label>
            </div>
                    <h2>Location</h2>
            <div>
  <select class="location" name="" id="uiLocations">
    <option value="" disabled="disabled" selected="selected">Choose a Location</option>
                    <option>Electronic City</option>
       <option>Rajaji Nagar</option>
  </select>
</div>
            <button class="submit" onclick="onClickedEstimatePrice()" type="button">Estimate Price</button>
            <div id="uiEstimatedPrice" class="result">          <h2></h2> </div>
</body>
                                                                    </html>
```

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;
}

.switch-field input:checked + label {
        background-color: #a5dc86;
        box-shadow: none;
}

.switch-field label:first-of-type {
        border-radius: 4px 0 0 4px;
}

.switch-field label:last-of-type {
        border-radius: 0 4px 4px 0;
}

.form {
        max-width: 270px;
        font-family: "Lucida
Grande", Tahoma, Verdana, sans-
serif;
        font-weight: normal;
        line-height: 1.625;
        margin: 8px auto;
```

```
padding-left: 16px;
        z-index: 2;
}

h2 {
        font-size: 18px;
        margin-bottom: 8px;
}
.area {
        font-family: "Roboto", sans-
serif;
        outline: 0;
        background: #f2f2f2;
        width: 76%;
        border: 0;
        margin: 0 0 10px;
        padding: 10px;
        box-sizing: border-box;
        font-size: 15px;
        height: 35px;
        border-radius: 5px;
}

.location {
        font-family: "Roboto", sans-
serif;
        outline: 0;
        background: #f2f2f2;
        width: 76%;
```

```
  border: 0;
        margin: 0 0 10px;
        padding: 10px;
        box-sizing: border-box;
        font-size: 15px;
        height: 40px;
        border-radius: 5px;
  }

  .submit {
        background: #a5dc86;
        width: 76%;
        border: 0;
        margin: 25px 0 10px;
        box-sizing: border-box;
        font-size: 15px;
        height: 35px;
        text-align: center;
        border-radius: 5px;
  }

  .result {
        background: #dcd686;
        width: 76%;
        border: 0;
        margin: 25px 0 10px;
        box-sizing: border-box;
        font-size: 15px;
        height: 35px;
```

```
        text-align: center;
}

.img {
        background:
url("https://images.unsplash.com/ph
oto-1564013799919-
ab600027ffc6?ixlib=rb-
1.2.1&auto=format&fit=crop&w=1
350&q=80");
        background-repeat: no-
repeat;
        background-size: auto;
        background-size: 100%
100%;
        -webkit-filter: blur(5px);
        -moz-filter: blur(5px);
        -o-filter: blur(5px);
        -ms-filter: blur(5px);
        filter: blur(15px);
        position: fixed;
        width: 100%;
        height: 100%;
        top: 0;
        left: 0;
        z-index: -1;
}

body,
html {
        height: 100%;
```

```
}
```

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),
            bhk: bhk,
            bath: bathrooms,
            location: location.value,
        },
        function (data, status) {
            console.log(data.estimated_price);
            estPrice.innerHTML =
                "<h2>" + data.estimated_price.toString() + " Lakh</h2>";
            console.log(status);
        }
    );
}


function onPageLoad() {
    console.log("document loaded");
    // var url = "http://127.0.0.1:5000/get_location_names"; // Use this if you are NOT using nginx which is first 7 tutorials
    var url = "/api/get_location_names"; // Use this if  you are using nginx. i.e tutorial 8 and onwards
```

```
        $.get(url, function (data, status) {
                console.log("got response for get_location_names request");
                if (data) {
                        var locations = data.locations;
                        var uiLocations = document.getElementById("uiLocations");
                        $("#uiLocations").empty();
                        for (var i in locations) {
                                var opt = new Option(locations[i]);
                                $("#uiLocations").append(opt);
                        }
                }
        });
}

window.onload = onPageLoad;
```

server.py

```
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()
```

util.py

```
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
```

References

- https://jupyter.org
- www.w3schools.com
- www.geeksforgeeks.com
- www.youtube.com