

# **Real Estate Price Prediction Based On Its Features Using Machine Learning**

Submitted in partial fulfillment of the requirements of the degree of

## **Bachelor OF ENGINEERING**

In

### **Information Technology**

By

Smith Dabreo (Roll No. 8382)

Shaleel Rodrigues (Roll No. 8423)

Valiant Rodrigues (Roll No. 8424)

Under the guidance of

**Prof. Parshvi Shah**



**Fr. Conceicao Rodrigues College of Engineering,  
Father Agnel Ashram, Bandstand, Bandra (West),  
Mumbai – 400050.**

**Department of Information Technology  
University of Mumbai**

November - 2020

# CERTIFICATE

This is to certify that the project entitled “ Real Estate Price Prediction Based On Its Features Using Machine Learning” is a bonafide work of Smith Dabreo (Roll No. 8382), Shaleel Rodrigues (Roll No. 8423), Valiant Rodrigues (Roll No. 8424) submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology.

(Name & Sign)  
Supervisor / Guide

Dr. Jagruti Save  
Head of Department

Dr. Srija Unnikrishnan  
Principal



## **Project Report Approval for Bachelor of Engineering**

This project report entitled Real Estate Price Prediction Based On Its Features Using Machine Learning by Smith Dabreo (Roll No. 8382), Shaleel Rodrigues (Roll No. 8423), Valiant Rodrigues (Roll No. 8424) is approved for the degree of Bachelor of Engineering in Information Technology.

Examiners

1.-----

2.-----

Date: 07/11/2020

Place: Bandra, Mumbai.

## DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

1. Smith Dabreo (8382) -----
2. Shaleel Rodrigues (8423)-----
3. Valiant Rodrigues (8424)-----

Date: 07/11/2020

## **ABSTRACT**

People and real estate agencies buy or sell houses, people buy to live in or as an investment and the agencies buy to run a business. Either way, we believe everyone should get exactly what they pay for. detecting over-valuation/under-valuation in housing markets is still more of an art than a science. Broad measures, such as house price to rent ratios, provide a first pass. But detailed analysis and judgment are needed to make a call about this issue. Here's where machine learning comes in, by training a ML model with hundreds and thousands of data a solution can be developed which will be powerful enough to predict prices accurately and can cater to everyone's needs.

## Acknowledgement

### Acknowledgement

It gives us great pleasure in expressing our gratitude to all those people who have supported us and had their contributions in making this project report possible. First and foremost, we express our profound sense of reverence to our guide **Prof. Parshvi Shah**, Assistant Professor, Department of Information Technology, Fr. Conceicao Rodrigues College of Engineering, Bandra West, Mumbai, for his constant guidance, support, motivation and untiring help during the course of our Bachelor of Engineering.

We are also thankful to **Dr. Jagruti Save**, Head of Department and **Dr. Srija Uniikrishnan**, Principal of Fr. Conceicao Rodrigues College of Engineering for their support and valuable suggestions. We are also thankful to all staff members of the Department of Information Technology, without whom the completion of this project report would have been impossible.

1. Smith Dabreo (8382)
2. Shaleel Rodrigues (8423)
3. Valiant Rodrigues (8424)

# TABLE OF CONTENTS

	ABSTRACT	i
	ACKNOWLEDGEMENT	ii
	TABLE OF CONTENTS	iii
	LIST OF FIGURES	iv
	LIST OF TABLES	v
	ABBREVIATIONS	vi
1	INTRODUCTION	13
	1.1. Background	13
	1.2. Limitations of Prevailing Methodologies	13
	1.3. Objectives	14
	1.4. Contribution of the Proposed System	15
2	LITERATURE REVIEW	16
3	PROPOSED WORK	20

	3.1.	Block Diagram of the System	20
	3.2.	System Design (UML Diagram, Activity Diagram, Sequence Diagram, Deployment Diagram, State chart Diagram, DFDs)	22
	3.3.	Dataset Source	29
4		SYSTEM IMPLEMENTATION	32
	4.1.	Algorithms Used	32
	4.2.	Modules Implemented	33
	4.3.	Parameters Estimated	41
5		RESULTS AND DISCUSSION	42
	5.1.	Data Extraction	42
	5.2.	Data Visualization	43
	5.3.	Plotting of the independent variable with the dependent	45
	5.4.	Correlation between all attributes and the label	52
	5.5.	Machine Learning Models	54
6		CONCLUSION AND FUTURE WORK	61



REFERENCES	62
------------	----

iii

## LIST OF FIGURES

3.1.	Block Diagram of the System	20
3.1.1	Workflow Diagram	21
3.2.1	UML Diagram	22
3.2.2.	Activity Diagram	23
3.2.3.	Sequence Diagram	25
3.2.4.	Deployment Diagram	26
3.2.5	State Chart Diagram	27
3.2.6	Data Flow Diagram Levels	28
4.1	Columns in the dataset	33
4.2	Simple Imputer Code Snippet	34
4.3	Dropping Null Price rows from Melbourne dataset	35

4.4	Label encoding	35
4.5	Stratified Shuffle Split in Boston dataset	36
4.6	Stratified Shuffle Split in Melbourne dataset	36
5.1	Extraction Of Boston Data	42
5.2	Extraction Of Melbourne Data	42
5.3	Data Visualization	43
5.4	Independent Variables with the Dependent	45
5.5	Relationship between Label and one random Attribute	46
5.6	Relationship between two Attributes	48
5.7	Factors Affecting the Price	50
5.8	Correlation between all attributes and the label	52
5.9	Predictions done by XGBoost Model	54
5.10	Evaluation of XGBoost Model	55
5.11	Predictions done by Decision Tree Model	56

5.12	Predictions done by Linear Regression Model	57
5.13	Predictions done by Random Forest Model	58
5.14	Predictions done by Gradient Boosting Model	59
5.15	Website (Undeveloped)	60

iv

## LIST OF TABLES

2	Comparison Chart	18
---	------------------	----

v

## **ABBREVIATIONS**

SVM	Support Vector Machine
LSVM	Least Squares Support Vector Machine
XGBoost	eXtreme Gradient Boosting
PLS	Partial Least Squares
RMSE	Root Mean Squared Error

# Chapter 1

## INTRODUCTION

### 1.1 Background

Machine Learning (ML) is an important aspect of modern business and research. It uses algorithms and neural network models to assist computer systems in progressively improving their performance. Machine Learning algorithms automatically build a mathematical model using sample data – also known as “training data” – to make decisions without being specifically programmed to make those decisions.

The primary aim of our project is to use these Machine Learning Techniques and curate them into models which can then serve the users. The main objective of a Buyer is to search for their dream house which has all the amenities they need and they look for these houses/Real estate with a price in mind and there is no guarantee that they will get the product for a proper price and will not be overpriced. Similarly, A seller looks for a Certain number which they can put on the estate as a price tag and this cannot be just a wild guess, lots of research needs to be put to come to a conclusion over a valuation of a home. Additionally, there exists a possibility of underpricing the product. If the price is predicted for these users this might help them get estates for their deserving prices not more not less.

### 1.2 Limitations of Prevailing Methodologies

There is a notable amount of research done in the house price prediction department but this research has not come to any real life solutions. There is little to no evidence of a working price predictor set up by a company. For now very less digital solutions exist for such a huge market and most of the methods used by people and companies are as follows

#### Buyers/Customers:

1. When people first think of buying a house/Real estate they tend to go online nowadays and try to study trends and other stuff so they can look for a house which contains everything and while doing this people make note of the price which goes with these. However, Average person doesn't

have in depth knowledge about what the actual price should be and these leads to misinformation as they believe the prices mentioned on the internet to be true.

2. The second thing that comes to mind while searching for a property is to contact various Estate agents. The problem with this is these agents need to be paid a fraction of the amount just for searching a house and setting a price tag for you. This price tag is blindly believed by normal people because they have no other choice. There might be cases that the agents and sellers may have a secret dealing and the customer might be sold an overpriced house.

#### Seller/Agencies:

1. When an individual thinks of selling his/her property they compare their property with hundreds and thousands of other properties which are posted all around the world. Determining the price by comparing with numerous amounts of estate is highly time consuming and has a huge risk of incorrect pricing

2. Huge Real estate companies have various products they need to sell and they have to assign people for each of the products to handle these products. This again bases the prediction of a price tag to a human where there is room for human error. Additionally these assigned individuals need to be paid. However having a computer do this work for you by crunching the heavy numbers can save a lot of time money and provide accuracy which a human cannot achieve.

These are just few of the limitations that exist in the current system.

### **1.3 Objectives**

The main goal of our project model are as follows:

- To analyse various features of numerous properties by feeding into the algorithm and thus predicting the appropriate price for a house based on the feature entered by the user.
- Further fine tuning the model such that maximum accuracy is achieved
- Making use of various ML algorithms and comparing and contrasting them with each other to determine which performs the best and which can help the user more in the long run by predicting the appropriate price for their house.

- To design a system which would help companies determine prices load of data within minutes which they can trust to be accurate
- To use various types of dataset for truly learning what features affect what and how the market can change the price.

## **1.4 Proposed System**

The purpose of this system is to determine the price of a house by looking at the various features which are given as input by the user. Giving these features to the ML model and returning the predictions.

This will be done by first searching for an appropriate dataset which can suit the needs of the developer as well as the user. Furthermore, after finalizing the dataset the dataset will go through the process known as data cleaning where all the data which is not needed will be eliminated and the raw data will be turned into a .csv file. Moreover, The data will go through data preprocessing where missing data will be handled then thil will go through data transformation where it will be converted into a numpy array so that it can finally be sent for training the model. While training various machine learning algorithms will be used so traint the model their error rate will be extracted and consequently an algorithm and model will be finalized which can yield accurate predictions.

Users and companies will be able to login and then fill a form about various attributes about their property which they want to predict the price of and after through selection of attributes the form will be submitted this data entered by the user will then go to the model and within seconds the user will be able to view the predicted price of the property that they put in.

## Chapter 2

# Literature Survey

Real Estate has become more than a necessity in this 21st century, It represents something much more nowadays. Not only for people looking into buying Real Estate but also the companies that sell these Estates. According to [4] Real Estate Property is not only the basic need of a man but today it also represents the riches and prestige of a person. Investment in real estate generally seems to be profitable because their property values do not decline rapidly. Changes in the real estate price can affect various household investors, bankers, policymakers, and many. Investment in the real estate sector seems to be an attractive choice for investments. Thus, predicting the real estate value is an important economic index. [3] suggests that Every single organization in today's real estate business is operating fruitfully to achieve a competitive edge over alternative competitors. There is a need to simplify the process for a normal human being while providing the best results. [6] proposed to use machine learning and artificial intelligence techniques to develop an algorithm that can predict housing prices based on certain input features. The business application of this algorithm is that classified websites can directly use this algorithm to predict prices of new properties that are going to be listed by taking some input variables and predicting the correct and justified price i.e. avoid taking price inputs from customers and thus not letting any error creeping in the system. [12] used Google Colab/Jupyter IDE. Jupyter IDE is an open-source web app that helps us to share as well create documents that have LiveCode, visualizations, equations, and text that narrates. It contains tools for data cleaning, data transformation, simulation of numeric values, modeling using statistics, visualization of data, and machine learning tools. [10] designed a system that will help people to know close to the precise price of real estate. User can give their requirements according to which they will get the prices of the desired houses User can also get the sample plan of the house to get a reference for houses. In [5] Housing value of the Boston suburb is analyzed and forecast by SVM, LSSVM, and PLS methods and the corresponding characteristics. After getting rid of the missing samples from the original data set, 400 samples are treated as training data and 52 samples are treated as test data. Housing value of the training data. As per [1]'s findings, the best accuracy was provided by the Random Forest Regressor followed by the Decision Tree Regressor. A similar result is generated by the Ridge and Linear Regression with a very slight reduction in Lasso. Across all groups of feature selections, there is no extreme difference between all regardless of strong or weak groups. It gives a good sign that the buying prices can be solely used for predicting the selling prices without considering other features to disseminate model over-fitting. Additionally, a reduction in accuracy is apparent in the very weak features group. The same pattern of results is visible on the Root Square Mean Error (RMSE) for all feature selection. [2] observed that their data set took more than one day to prepare. As opposed to performing the computations sequentially, we might utilize various processors and parallel the computations involved, which might possibly decrease the preparation time Furthermore prediction period. Include All the more functionalities under the model, we can give choices for clients with select a



district alternately locale should produce those high-temperature maps, as opposed to entering in the list. [7] used a data set of 100 houses with several parameters. We have used 50 percent of the data set to train the machine and 50 percent to test the machine. The results are truly accurate. And we have tested it with different parameters also. Not using PSO makes it easier to train machines with complex problems and hence regression is used. [13] experimented with the most fundamental machine learning algorithms like decision tree classifier, decision tree regression, and multiple linear regression. Work is implemented using the Scikit-Learn machine learning tool. This work helps the users to predict the availability of houses in the city and also to predict the prices of the houses. [8] used machine learning algorithms to predict house prices. We have mentioned the step by step procedure to analyze the dataset. These feature sets were then given as an input to four algorithms and a CSV file was generated consisting of predicted house prices. [9] expressed that There is a need to use a mix of these models a linear model gives a high bias (underfit) whereas a high model complexity-based model gives a high variance (overfit). The outcome of this study can be used in the annual revision of the guideline value of land which may add more revenue to the State Government while this transaction is made. [11] concludes that by conducting this experiment with various machine learning algorithms its been clear that random for-set and gradient boosted trees are performing better with more accuracy percentage and with fewer error values. when this experiment is compared with the and to the result achieved these algorithms predict well.

## Comparison Chart:

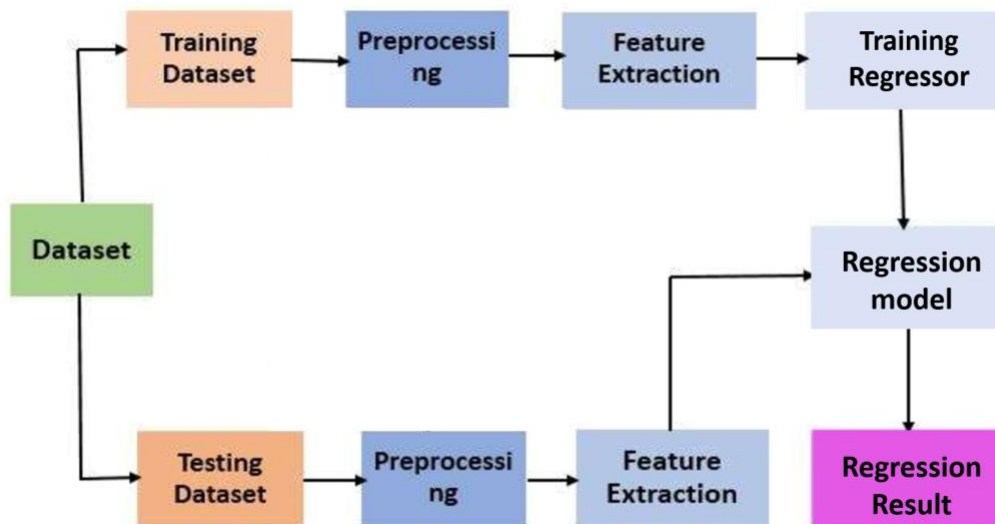
Sr. No	TitleT	Authors	Publication Year	Technique and Algorithm
1	Machine Learning Housing Price Prediction in etaling Jaya, Selangor, Malaysia	Thuraiya Mohd, Suraya Masrom, Noraini Johari	2019	Random forest regressor, linear regression, lasso, Ridge and decision tree regressor.
2	House Price Prediction Using Machine Learning	G. Naga Satish, Ch. V. Raghavendran, M.D.Sugnana Rao, Ch.Srinivasulu	2019	Linear regression algorithm
3	House Price ForeCasting Using Machine Learning	Alisha Kuvalekar , Shivani Manchewar , Sidhika Mahadik	2018	Decision tree Machine Learning Algorithm
4	Valuation Of House Prices Using Predictive Techniques	Neelam Shinde, Kiran Gawande	2018	Decision tree, lasso,logistic regression algorithms
5	Housing Value Forecasting Based on Machine Learning Methods	Jingyi Mu, Fang Wu,and Aihua Zhang	2014	SVM, LSSVM, and PLS algorithm
6	PropTech for Proactive Pricing of Houses in Classified Advertisements in the Indian Real Estate Market	Sayan Putatunda		Random Forest, Gradient boosting, ANN1, ANN2, ANN3 algorithms
7	House Price Prediction Using Machine Learning	Atharva Chouthai, Mohammed Athar Rangila , Sanved Amate, Prayag Adhikari, Vijay Kukre	2019	Gradient boosting algorithm
8	Predicting Housing Prices using Machine Learning Techniques	B.Balakumar, P.Raviraj, S.Essakkiammal		Decision tree overfit best algorithm

9	Literature Review on Real Estate Value Prediction Using Machine Learning	Akshay Babu, Dr. Anjana S Chandran	2019	Support vector Regression, Random Forest, KNN, linear regression algorithms
10	House Planning and Price Prediction System using Machine Learning	Mr. Rushikesh Naikare, Mr. Girish Gahandule, Mr. Akash Dumbre, Mr. Kaushal Agrawal, Prof. Chaitanya Manka	2019	Regression Algorithm
11	Real Estate Price Prediction Using Machine Learning	Aswin Sivam Ravikumar	2016	Support vector, Gradient boosting, multiple regression, Random Forest algorithms
12	House Price Prediction	Bindu Sivasankar, Arun P. Ashok, Gouri Madhu, Fousiya S	2020	Ridge, Lasso and XG boost algorithm
13	House Price Prediction Modeling Using Machine Learning	Dr. M. Thamarai ,Dr. S P. Malarvizhi	2020	Decision tree classifier, Decision tree Regression, Multiple Linear Regression

## Chapter 3

### PROPOSED WORK

#### 3.1. Block Diagram of the System



The above block diagram is the traditional Machine Learning Approach. It consists of two sections: the training and the testing. The training has the following components: the label, input, feature extractor and the machine machine learning algorithm. The testing section has the following components in it: the input, feature extractor, the classifier model and the output label.

**Input :** The input consists of data collected from various sources.

**Feature Extractor:** Only important features which affect the prediction results are kept other all unnecessary attributes are discarded like id or name.

**Features :** After feature extraction only some inputs are considered which largely contribute in the prediction of the model.

**Machine Learning Algorithm:** Machine Learning Algorithm is the method by which the AI system conducts its task, generally predicting output values from given input data. The two main processes of machine learning algorithms are classification and regression.

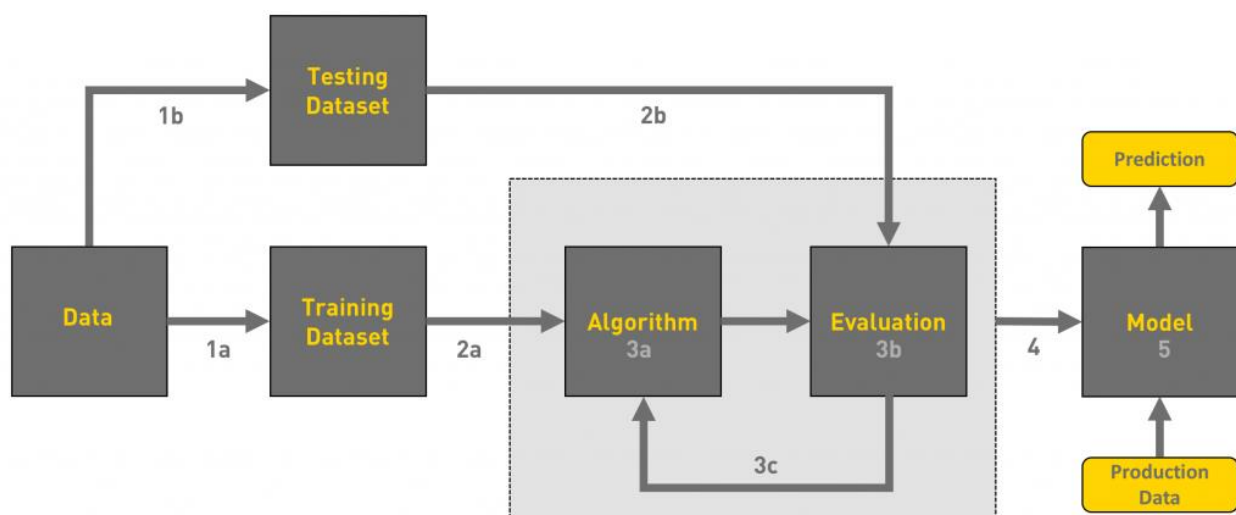
**Regression Model:** Regression model consists of a set of machine learning methods that allow us to predict a continuous outcome variable (y) based on the value of one or multiple predictor variables (x).

Briefly, the goal of a regression model is to build a mathematical equation that defines y as a function of the x variables.

**Label:** The label is the output obtained from the model after training.

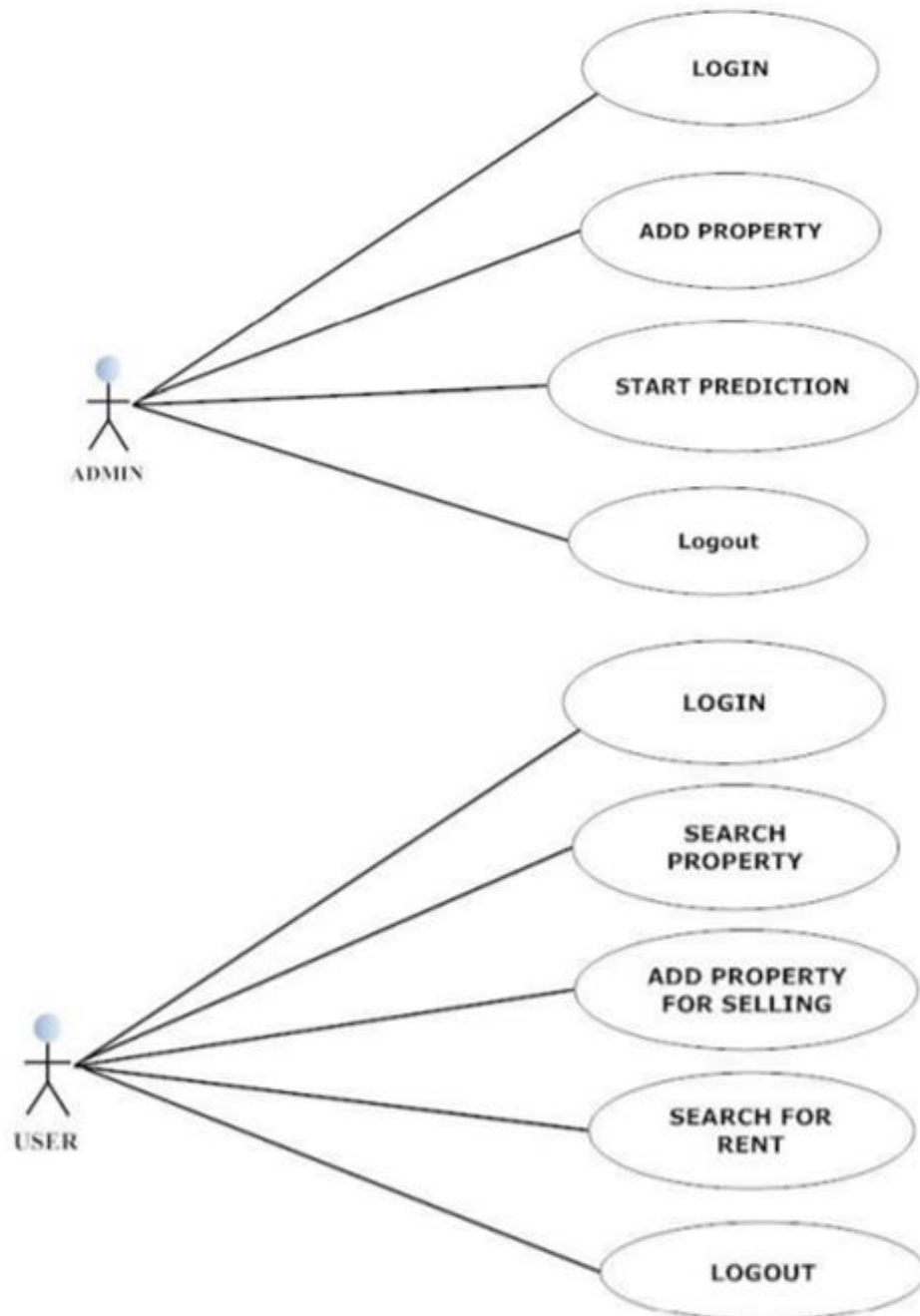
The data obtained from the dataset is given as a training input first and the relevant training features are extracted. These training features are preprocessed in order to get a normalised dataset and labelling of the data row is done. The result from the training dataset is fed to the Machine learning algorithm. The result from the Machine Learning Algorithm is fed to the Regression model thus producing a trained model or trained regressor that can take the new data that is the extracted feature from the test as an input and predicts its output label.

### 3.1.1 Workflow Diagram



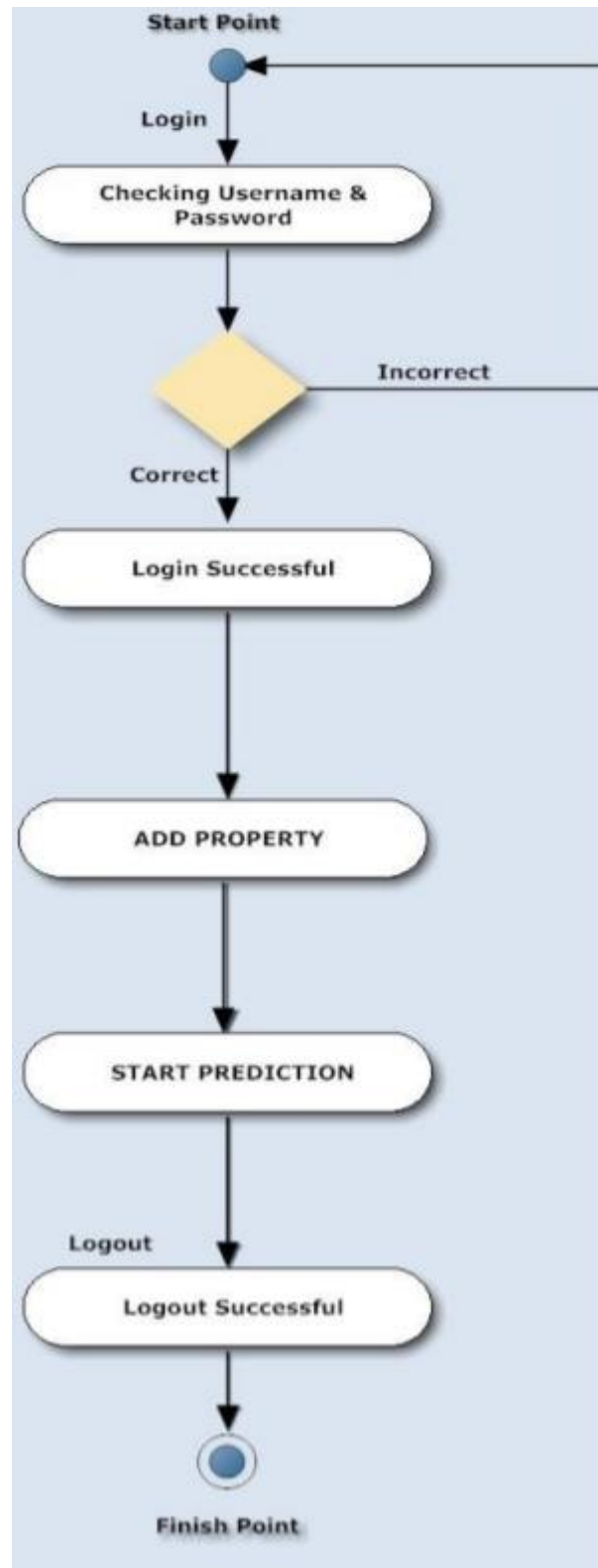
## 3.2. System Design

### 3.2.1 UML diagram

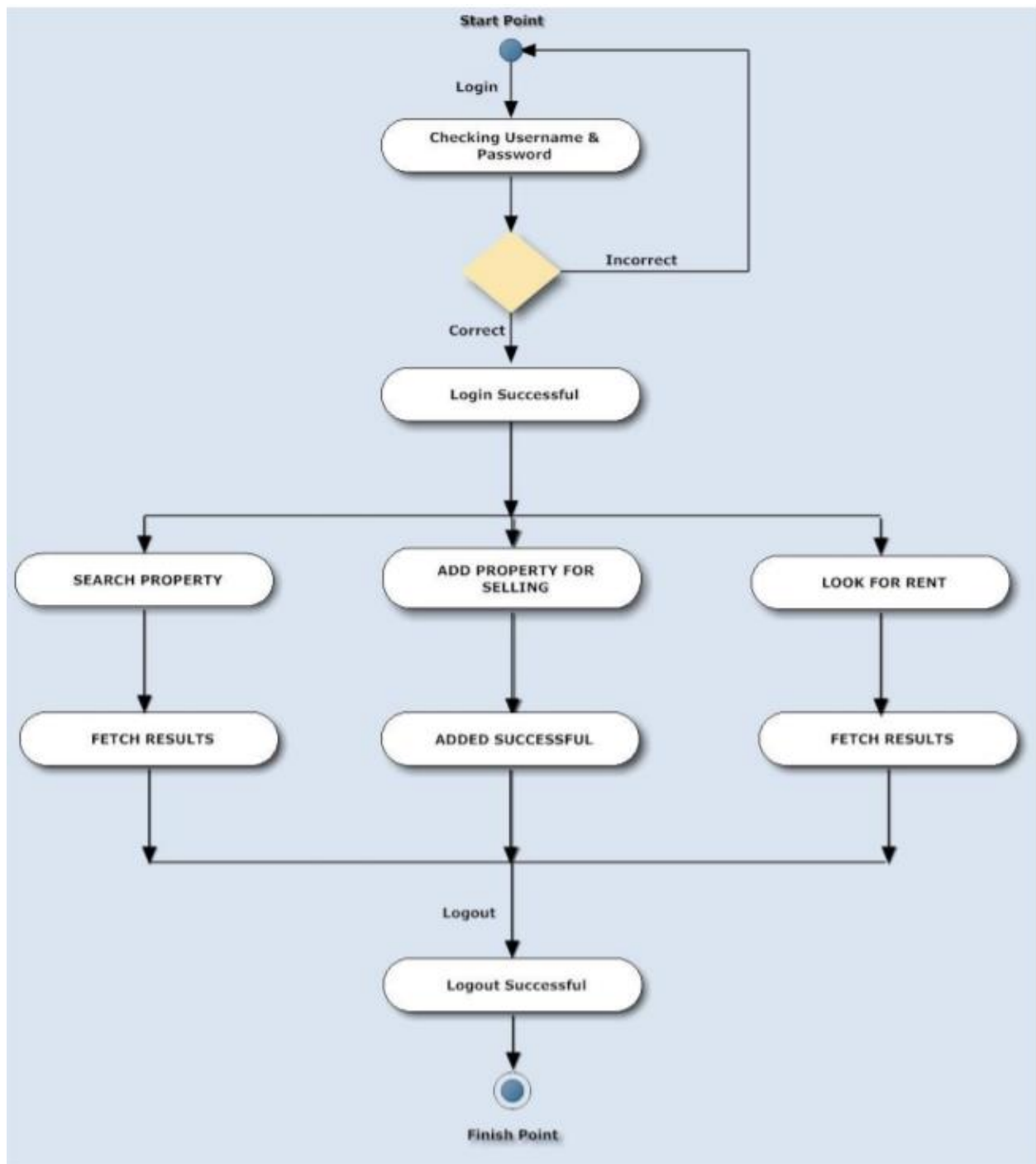


### 3.2.3 Activity diagram

#### 1. Admin

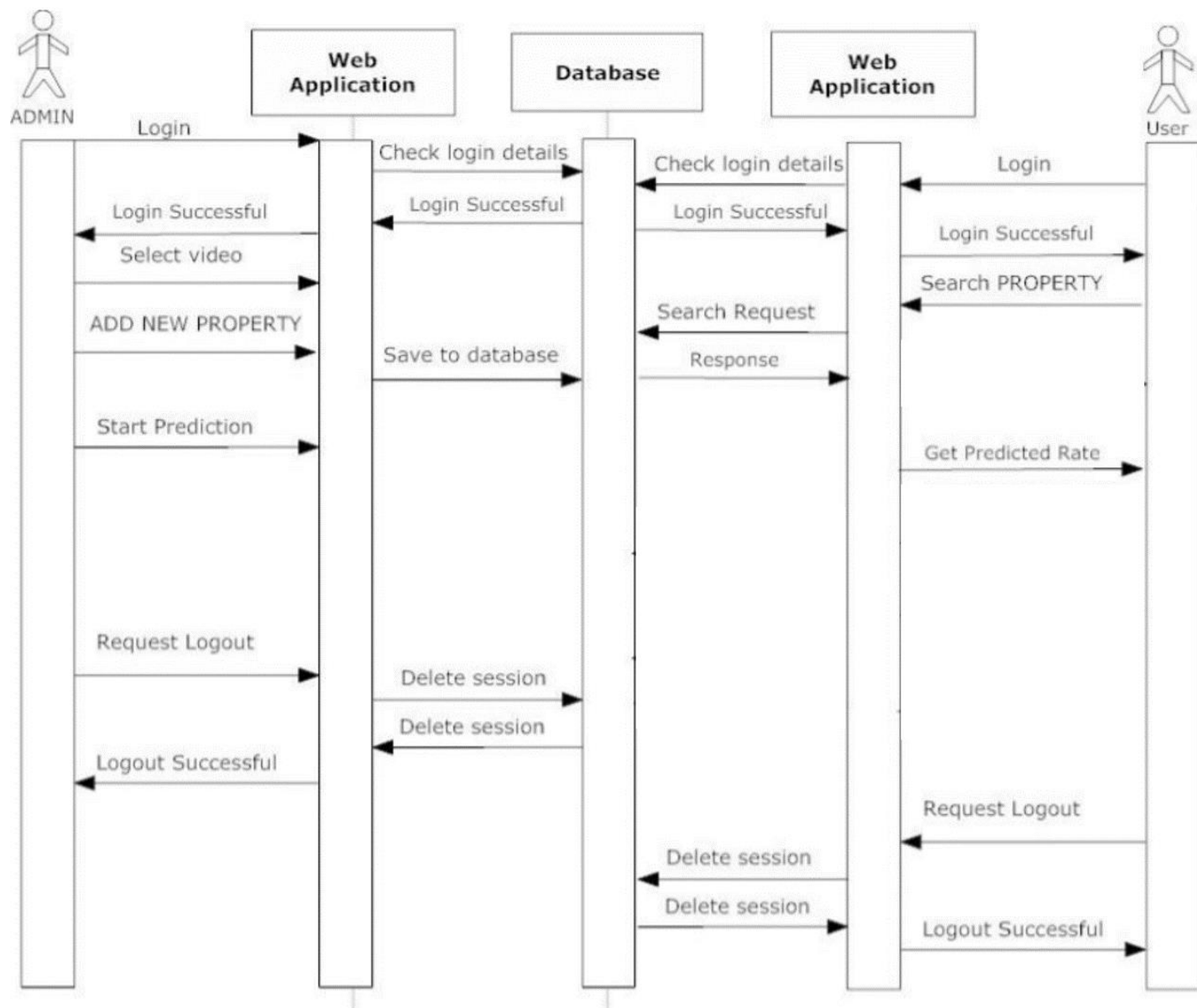


## 2. User

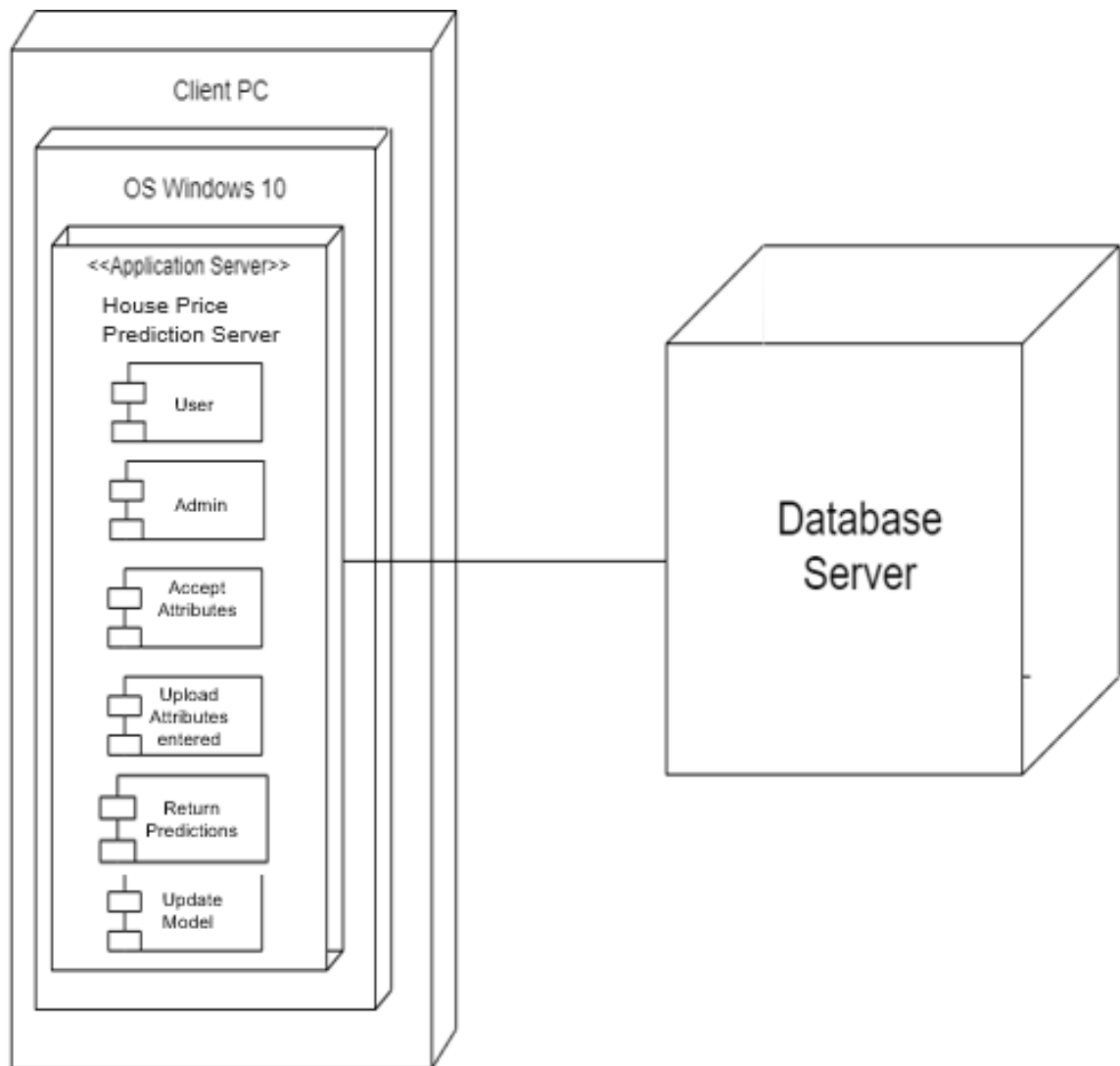




### 3.2.4 Sequence Diagram

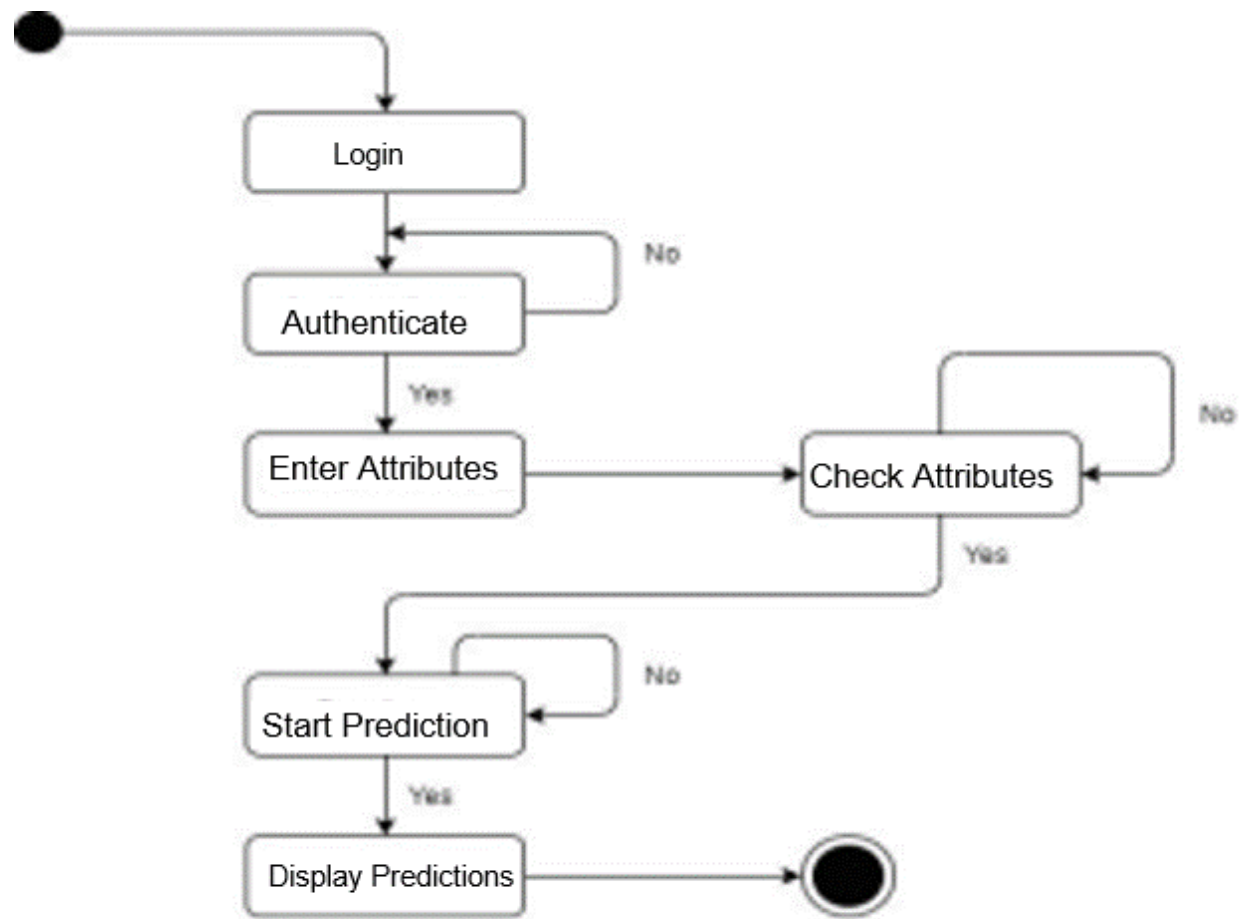


### 3.2.5 Deployment diagram



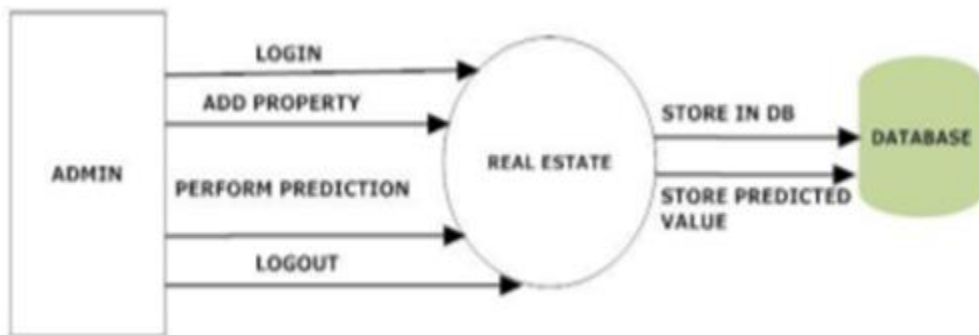
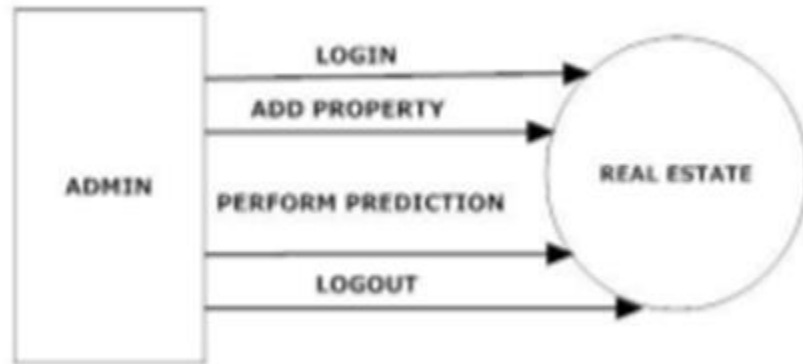
**Deployment Diagram**

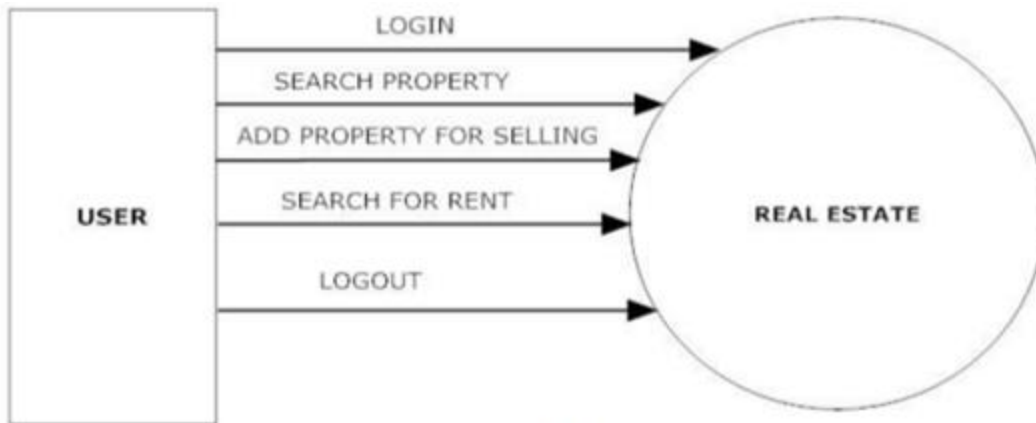
### 3.2.6 State Chart diagram



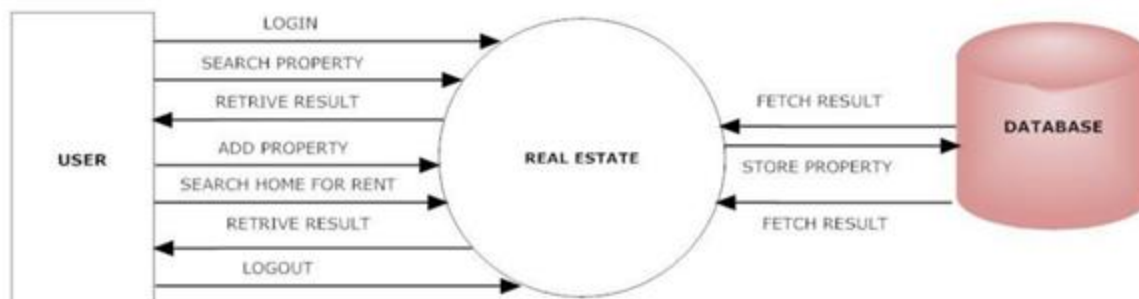
**State Chart Diagram**

### 3.2.7 Data Flow Diagram Levels





**LEVEL 0(User)**



**LEVEL 1(User)**

### 3.3. Dataset Source

The dataset sources are UCI Repository(now moved to kaggle) and Kaggle respectively.

1. Boston Dataset:

<https://www.kaggle.com/heptapod/uci-ml-datasets>

2. Melbourne Dataset:

<https://www.kaggle.com/dansbecker/melbourne-housing-snapshot>

Description:

1. Boston Dataset:

Total number of values : 506

Variable Description:

1. CRIM per capita crime rate by town
2. ZN proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS proportion of non-retail business acres per town
4. CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX nitric oxides concentration (parts per 10 million)
6. RM average number of rooms per dwelling
7. AGE proportion of owner-occupied units built prior to 1940
8. DIS weighted distances to five Boston employment centres
9. RAD index of accessibility to radial highways
10. TAX full-value property-tax rate per \$10,000
11. PTRATIO pupil-teacher ratio by town
12. B  $1000(B_k - 0.63)^2$  where  $B_k$  is the proportion of blacks by town
13. LSTAT % lower status of the population
14. MEDV Median value of owner-occupied homes in \$10,000's

This data set is an ideal data set for price prediction because it has all numeric values which were carefully gathered and recorded and processed so that developers have less problems with the dataset. However, the data set is small and hence does not apply in real-life scenarios.

2. Melbourne dataset:

Total number of values : 31420

Variable Description:

1. CouncilArea: Governing council for the area

## 2. Method:

1 - property sold; 2 - property sold prior; 3 - property passed in; 4 - sold prior not disclosed; 5 - sold not disclosed; 6 - no bid; 7 - vendor bid; 8 - withdrawn prior to auction; 9 - sold after auction; 10 - sold after auction price not disclosed; 11 - price or highest bid not available.

3. Regionname: General Region (West, North West, North, North east ...etc)

4. Rooms: Number of rooms:

5. Type: 1 - house,cottage,villa, semi,terrace; 2 - unit, duplex; 3 - townhouse.

6. Distance: Distance from CBD in Kilometres

7. Bedroom2 : Scraped # of Bedrooms (from different source)

8. Bathroom: Number of Bathrooms

9. Car: Number of carspots

10. Landsize: Land Size in Metres

11. BuildingArea: Building Size in Metres

12. YearBuilt: Year the house was built

13. Propertycount: Number of properties that exist in the suburb.

14. Price: Price in Australian dollars

This dataset prevails over the boston dataset because it has a massive amount of data with more attributes contributing towards the price prediction. However, it has a lot of missing values which may cause a problem but are fixable in python.

- For training the model we have used the above datasets.
- We have trained all the models at least once.
- Here we have done an exploratory analysis and a visualization analysis of the Recruitment dataset.

## Chapter 4

### SYSTEM IMPLEMENTATION

#### 4.1 Algorithms Used

We have currently used Five algorithms namely Decision Tree, Linear Regression, Random Forest, XGBoost and Gradient Boosting(The first 4 for boston dataset and all of them for Melbourne dataset).

**Decision Tree** is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs and utility. Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

**Linear Regression** is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

**Random Forest** is an ensemble learning method for classification and regression. It is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

**XGBoost** is an implementation of Gradient Boosted decision trees. This library was written in C++. It is a type of Software library that was designed basically to improve speed and model performance. It has recently been dominating in applied machine learning. XGBoost models majorly dominate in many Kaggle Competitions. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. Weight of variables predicted wrong by the tree is increased and these the variables are then fed to the second decision tree. These



individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

**Gradient Boosting** is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

## 4.2 Modules Implemented

### 4.2.1 Data Cleaning:

#### Handling Missing Values

The boston dataset only had 5 missing values. we can't drop the null values in the RM column since it may destroy some potential information from our dataset. These null values were handled by replacing them with the median value. This was done by implementing the simple imputer in the pipeline itself so that any miss value in the future will be handled automatically whenever it passed through the pipeline.

```
housing.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype  
---  -
0   CRIM        506 non-null   float64
1   ZN          506 non-null   float64
2   INDUS       506 non-null   float64
3   CHAS        506 non-null   int64  
4   NOX         506 non-null   float64
5   RM          501 non-null   float64
6   AGE         506 non-null   float64
7   DIS         506 non-null   float64
8   RAD         506 non-null   int64  
9   TAX         506 non-null   int64  
10  PTRATIO     506 non-null   float64
11  B           506 non-null   float64
12  LSTAT       506 non-null   float64
13  MEDV        506 non-null   float64
dtypes: float64(11), int64(3)
memory usage: 55.4 KB
```

(a) Columns in the Boston dataset

The Melbourne dataset had a lot of missing values( in thousands). It even had missing label values. Similar to the Boston dataset we implemented the simple imputer and all the values were filled with

their respective medians. However for the labels all the null rows were dropped since it was negatively affecting the model.

```
[5] housing.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31420 entries, 0 to 31419
Data columns (total 14 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   CouncilArea     31420 non-null  int64
1   Method          31420 non-null  int64
2   Regionname      31420 non-null  int64
3   Rooms           31420 non-null  int64
4   Type            31420 non-null  int64
5   Distance        31420 non-null  float64
6   Bedroom2        24211 non-null  float64
7   Bathroom        24205 non-null  float64
8   Car             23821 non-null  float64
9   Landsize        20621 non-null  float64
10  BuildingArea    12569 non-null  float64
11  YearBuilt       14294 non-null  float64
12  Propertycount   31420 non-null  int64
13  Price           24589 non-null  float64
dtypes: float64(8), int64(6)
memory usage: 3.4 MB
```

(b) Columns in the Melbourne dataset

Figure 4.1 Columns in the dataset

## Creating a pipeline

```
In [56]: from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler # FOR SCALING
my_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy="median")),
    # We can add as many as we want in our pipeline
    ('std_scaler', StandardScaler()),
])
```

Figure 4.2 Simple Imputer Code Snippet

We have dropped a few columns like ID or seller\_info since they didn't have any affect on the final predictions.

## Dropping Null Price Rows

```
[27] strat_test_set = strat_test_set.dropna(subset=["Price"]) #Option 1
      strat_test_set.shape
```

```
(4929, 14)
```

```
[28] strat_train_set = strat_train_set.dropna(subset=["Price"]) #Option 1
      strat_train_set.shape
```

```
(19660, 14)
```

Figure 4.3 Dropping Null Price rows from Melbourne dataset

### 4.2.2 Data Preprocessing:

#### a) Label Encoding(only needed for Melbourne dataset)

We have used a excel formula for encoding our columns which had words as classes and were of less than 15 types. Some of the columns which were label encoded were Type and Method

CouncilArea	Method	Region	Rooms	Type
32	10	1	2	1
32	1	1	2	1
32	1	1	2	1
32	7	1	3	2
32	2	1	3	1
32	3	1	3	1
32	7	1	4	1
32	5	1	4	1
32	1	1	2	1
32	1	1	2	1
32	1	1	2	1
32	1	1	3	1
32	3	1	2	2
32	8	1	4	1
32	1	1	2	1
32	1	1	3	1

Figure 4.4 Label encoding

## b) Train and Test Split

We have split the dataset into two sets i.e Training set and Testing set. Training set consists of 80% of the dataset and the testing set has 20% of the dataset.

We had columns with only two distinct values and wanted to make sure that the splitting should split these values in equal proportions. Therefore, we used stratified shuffle split for train test splitting for better results

```
In [24]: from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
print(f"Rows in train set: {len(train_set)}\nRows in test set: {len(test_set)}\n")

Rows in train set: 404
Rows in test set: 102
```

### Stratified Shuffle Split of CHAS

```
In [25]: from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(housing, housing['CHAS']):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]
```

```
In [26]: strat_test_set["CHAS"].value_counts()
```

```
Out[26]: 0    95
         1     7
         Name: CHAS, dtype: int64
```

```
In [27]: strat_train_set["CHAS"].value_counts()
```

```
Out[27]: 0    376
         1    28
         Name: CHAS, dtype: int64
```

Figure 4.5 Stratified Shuffle Split in Boston dataset

```
[25] from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
print(f"Rows in train set: {len(train_set)}\nRows in test set: {len(test_set)}\n")

Rows in train set: 25136
Rows in test set: 6284
```

### Stratified Shuffle Split of "Type"

```
[26] from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(housing, housing['Type']):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]
```

Figure 4.6 Stratified Shuffle Split in Melbourne dataset

### 4.2.3 Decision Tree Regression

#### Code:

```
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
print(model)
model.fit(housing_num_tr, housing_labels)
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, housing_num_tr, housing_labels,
scoring="neg_mean_squared_error", cv=10)
rmse_scores = np.sqrt(-scores)
def print_scores(scores):
    print("Scores:", scores)
    print("Mean: ", scores.mean())
    print("Standard deviation: ", scores.std())
print_scores(rmse_scores)
scores = cross_val_score(model, housing_num_tr, housing_labels, cv=10)
print("Mean cross-validation score: %.2f" % scores.mean())
```

#### Results:

##### 1. Melbourne

Mean: 421026.9

Standard deviation: 15907.3

Mean cross-validation score: 0.56

##### 2. Boston

Mean: 4.189

Standard deviation: 0.8480

Mean cross-validation score: 0.76

### 4.2.4 Linear Regression

#### Code:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

```

print(model)
model.fit(housing_num_tr, housing_labels)
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, housing_num_tr, housing_labels,
scoring="neg_mean_squared_error", cv=10)
rmse_scores = np.sqrt(-scores)
def print_scores(scores):
    print("Scores:", scores)
    print("Mean: ", scores.mean())
    print("Standard deviation: ", scores.std())
print_scores(rmse_scores)
scores = cross_val_score(model, housing_num_tr, housing_labels, cv=10)
print("Mean cross-validation score: %.2f" % scores.mean())

```

#### Results:

##### 1. Melbourne

Mean: 519029.1

Standard deviation: 97954.6

Mean cross-validation score: 0.31

##### 2. Boston

Mean: 4.22

Standard deviation: 0.75

Mean cross-validation score: 0.69

### **4.2.5 Random Forest**

#### Code:

```

from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()

print(model)

```

```

model.fit(housing_num_tr, housing_labels)

from sklearn.model_selection import cross_val_score

scores = cross_val_score(model, housing_num_tr, housing_labels,
                          scoring="neg_mean_squared_error", cv=10)

rmse_scores = np.sqrt(-scores)

def print_scores(scores):

    print("Scores:", scores)

    print("Mean: ", scores.mean())

    print("Standard deviation: ", scores.std())

print_scores(rmse_scores)

scores = cross_val_score(model, housing_num_tr, housing_labels, cv=10)

print("Mean cross-validation score: %.2f" % scores.mean())

```

#### Results:

##### 1. Melbourne

Mean: 312347.6

Standard deviation: 19880.3

Mean cross-validation score: 0.76

##### 2. Boston

Mean: 3.38

Standard deviation: 0.76

Mean cross-validation score: 0.85

## **4.2.5 XGBoost**

#### Code:

```

import xgboost as xgb
model = xgb.XGBRegressor(n_estimators=100, learning_rate=0.08, gamma=0, subsample=0.75,
colsample_bytree=1, max_depth=7)
print(model)
model.fit(housing_num_tr, housing_labels)
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, housing_num_tr, housing_labels,
scoring="neg_mean_squared_error", cv=10)
rmse_scores = np.sqrt(-scores)
def print_scores(scores):
    print("Scores:", scores)
    print("Mean: ", scores.mean())
    print("Standard deviation: ", scores.std())
print_scores(rmse_scores)
scores = cross_val_score(model, housing_num_tr, housing_labels, cv=10)
print("Mean cross-validation score: %.2f" % scores.mean())

```

#### Results:

##### 1. Melbourne

Mean: 299880.4

Standard deviation: 23019.5

Mean cross-validation score: 0.78

##### 2. Boston

Mean: 3.06

Standard deviation: 0.75

Mean cross-validation score: 0.88

## **4.2.5 Gradient Boosting**

#### Code:

```

from sklearn import ensemble

```



```

from sklearn.ensemble import GradientBoostingRegressor
params = {'n_estimators': 500, 'max_depth': 4, 'min_samples_split': 2, 'learning_rate': 0.01, 'loss':
'ls'}
model = ensemble.GradientBoostingRegressor(**params)print(model)
model.fit(housing_num_tr, housing_labels)
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, housing_num_tr, housing_labels,
scoring="neg_mean_squared_error", cv=10)
rmse_scores = np.sqrt(-scores)
def print_scores(scores):
    print("Scores:", scores)
    print("Mean: ", scores.mean())
    print("Standard deviation: ", scores.std())
print_scores(rmse_scores)
scores = cross_val_score(model,housing_num_tr, housing_labels,cv=10)
print("Mean cross-validation score: %.2f" % scores.mean())

```

### Results:

#### 1. Melbourne

Mean: 342911.4

Standard deviation: 26544.5

Mean cross-validation score: 0.71

## **4.3 Parameters Estimated**

- All the parameters above were used while training the data
- Any null values present were imputed with their respective median instantly
- In Melbourne Dataset, only The null Price rows were dropped (no. of rows dropped = 6831)

## Chapter 5

### RESULTS AND DISCUSSION

#### 5.1 Data Extraction:

The below screenshot is the set of datasets that we have extracted and have provided as an input into our model which are segregated into columns.

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

In [2]: housing = pd.read_csv("data.csv") #imported daata

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

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

Figure 5.1 Extraction Of Boston Data

```
[1] import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

[2] housing = pd.read_csv("houseprice.csv") #imported daata

[3] housing.head()
```

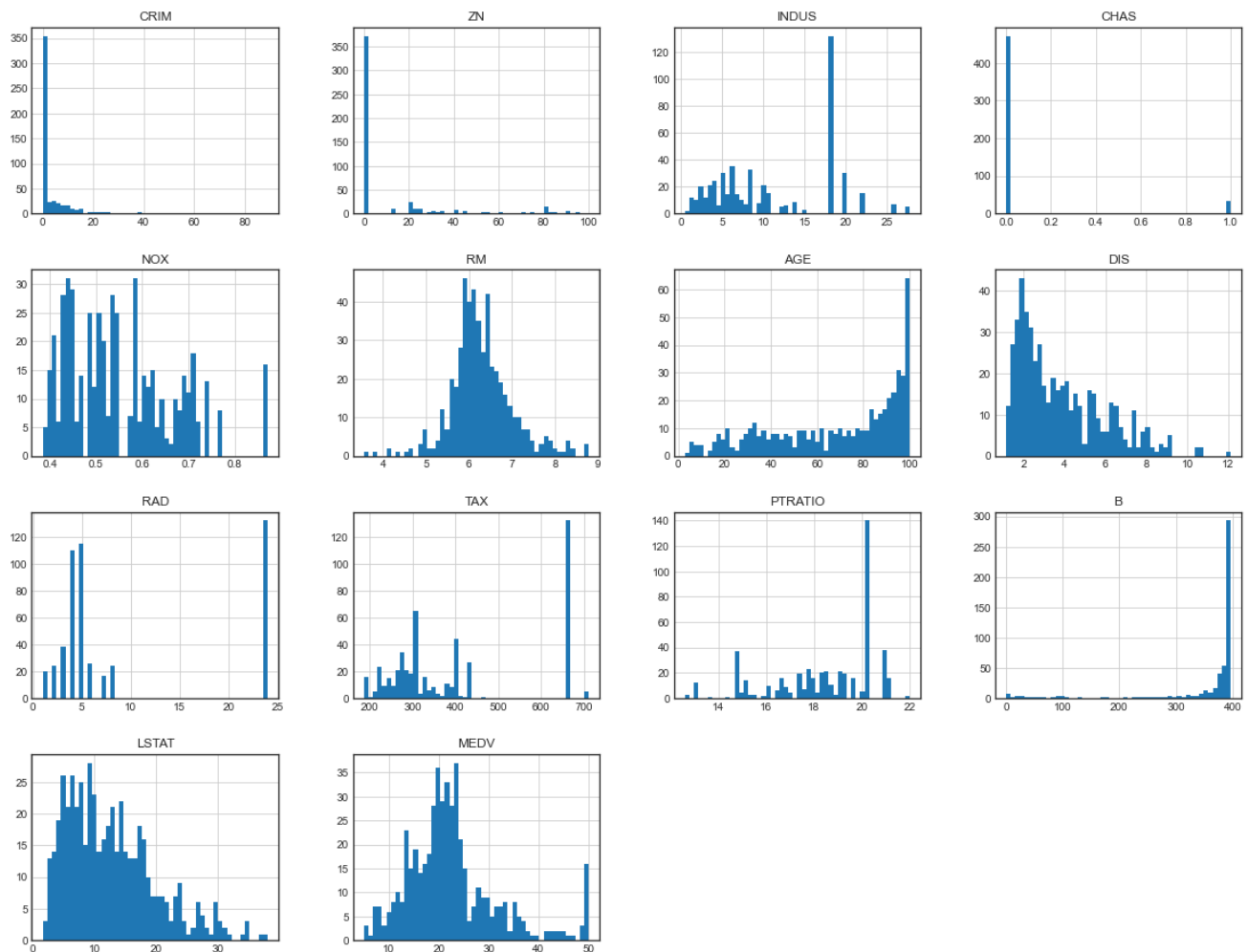
	CouncilArea	Method	Regionname	Rooms	Type	Distance	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Propertycount	Price
0	32	10	1	2	1	2.5	2.0	1.0	1.0	126.0	NaN	NaN	4019	NaN
1	32	1	1	2	1	2.5	2.0	1.0	1.0	202.0	NaN	NaN	4019	1480000.0
2	32	1	1	2	1	2.5	2.0	1.0	0.0	156.0	79.0	1900.0	4019	1035000.0
3	32	7	1	3	2	2.5	3.0	2.0	1.0	0.0	NaN	NaN	4019	NaN
4	32	2	1	3	1	2.5	3.0	2.0	0.0	134.0	150.0	1900.0	4019	1465000.0

Figure 5.2 Extraction Of Melbourne Data

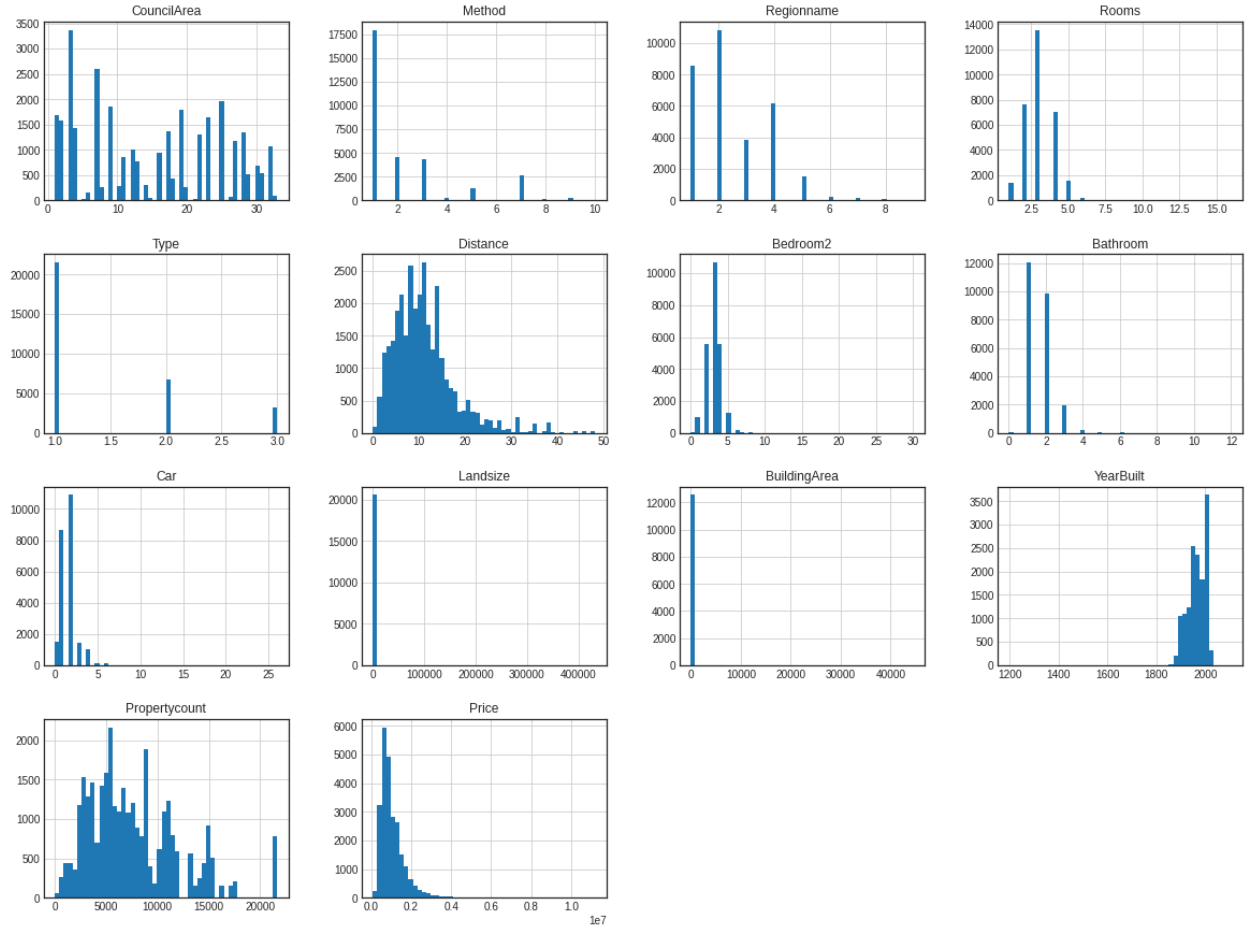
After the extraction of the datasets, They were examined. We speculated that the provided dataset has several criteria like CRIM, RM, MEDV and Rooms, Method, Price . We hinged that our main objective is to Determine the accurate Price of the House which would be our dependent variable and the rest of the attributes would be independent variables.

## 5.2 Data Visualisation:

Here a = Boston dataset & b = Melbourne Dataset



(a)



(b)

Figure 5.3(a & b) Data Visualisation

After checking and handling the null data set the data has been visualised in the form of a bar graph. From the bar graph we have inferred that for (a) most of the house prices lie between 15 to 30 also CHAS has almost 8 times more values as 1 than 0. Additionally, most of the houses in the dataset come equipped with 5 to 8 rooms. For (b) the median price was 870000 however the mean was 1048342 this means there are few values in the dataset with extremely high or extremely low prices. Also, the Distance from CBD was between 2 to 18 kilometers. Furthermore, The type of houses were almost 20000 of these entries were of type 1(house,cottage,villa, semi,terrace) and had around 2 to 5 rooms.

### 5.3 Plotting Of The Independent Variables with the Dependent :

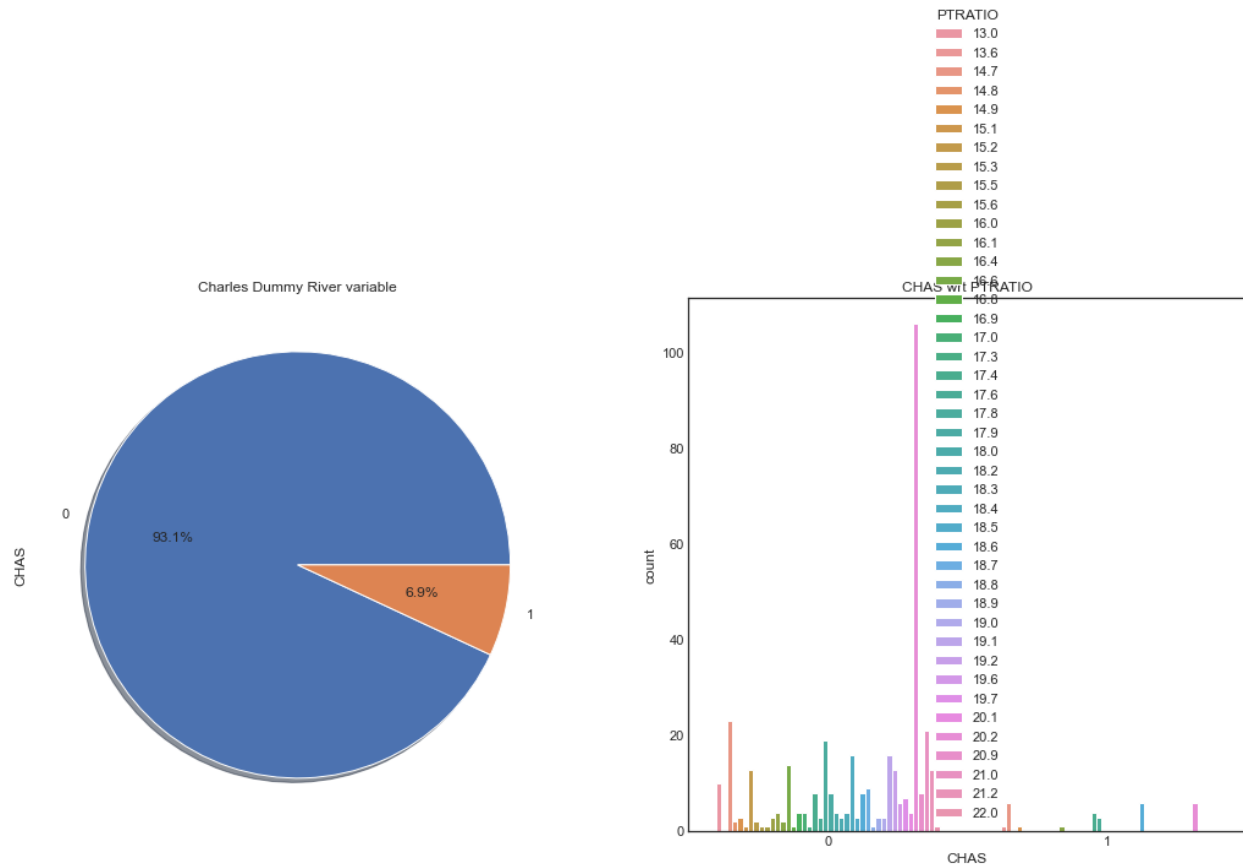


Figure 5.4 (a) CHAS vs PTRATIO

The CHAS attribute which has only two values was plotted against PTRATIO. It was observed that most of the CHAS had PTRATIO between 20 to 21 and also most of these CHAS were 0.

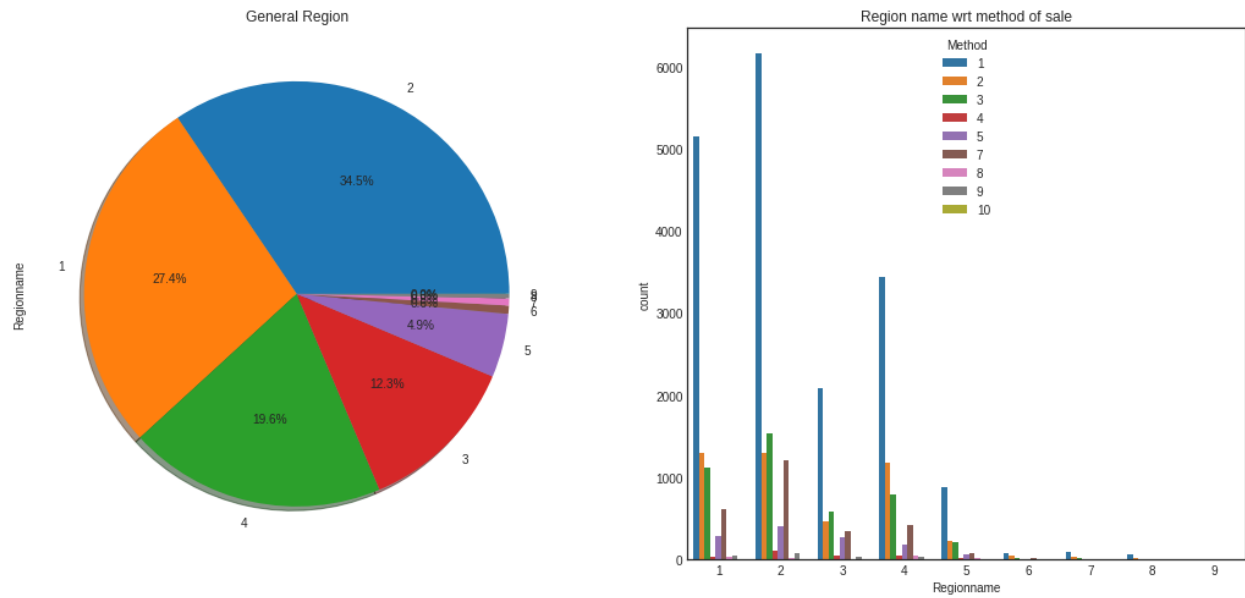
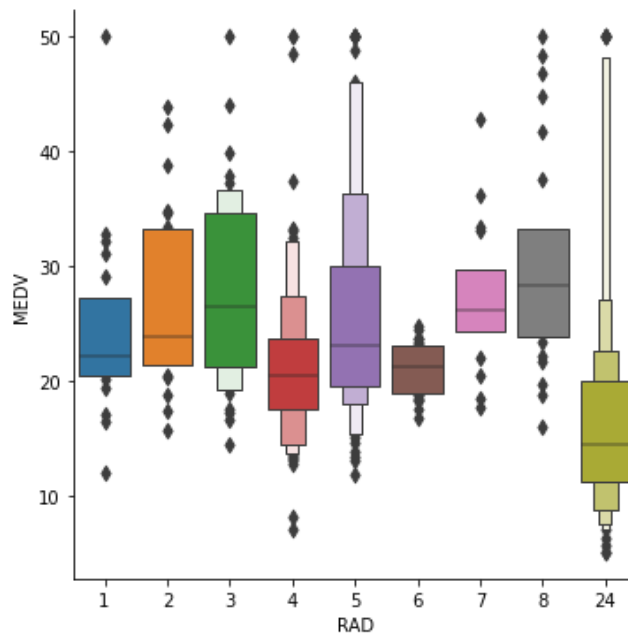


Figure 5.4 (b) Region vs Method of sale

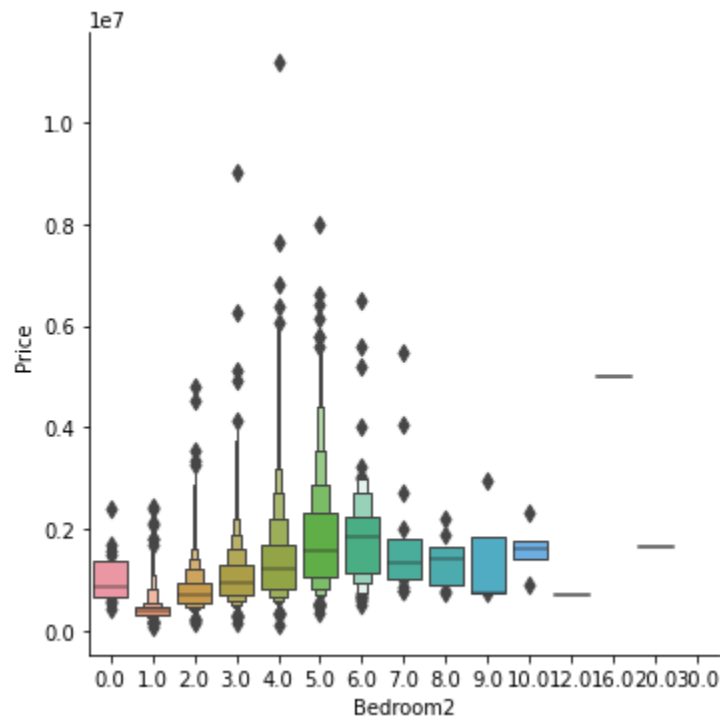
Various Regions were compared with the method of sale to determine which method of sale was used most in all these regions. It was evident that Method 1 was mostly preferred method in all the Regions. Method 7-10 were extremely rare.

### Determining the relationship between Label and one random Attribute :



(a) MEDV(price) wrt RAD

For maximum of the Distance's the inner 50% of the price was between 15 to 35 and median hitting mostly at 20. However for RAD 21 the price experienced a downfall as the distance from the highway increased the price got lower. Median staying at just 15.

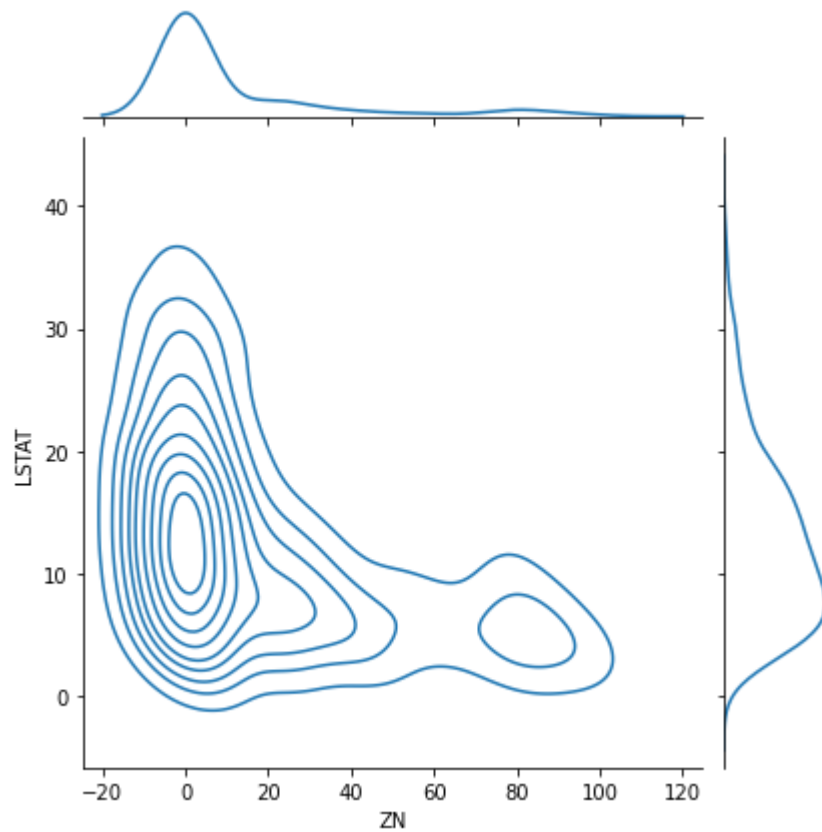


(b) Price wrt Bedroom count

Figure 5.5 (a & b) Label vs Attribute

It was observed that increasing bedroom count did not affect the price that much and surprisingly houses with a number of bedrooms between 3 to 6 topped the chart rather than the houses which had 8 to 30 bedrooms. And a house with 4 bedrooms had the highest price as compared to others.

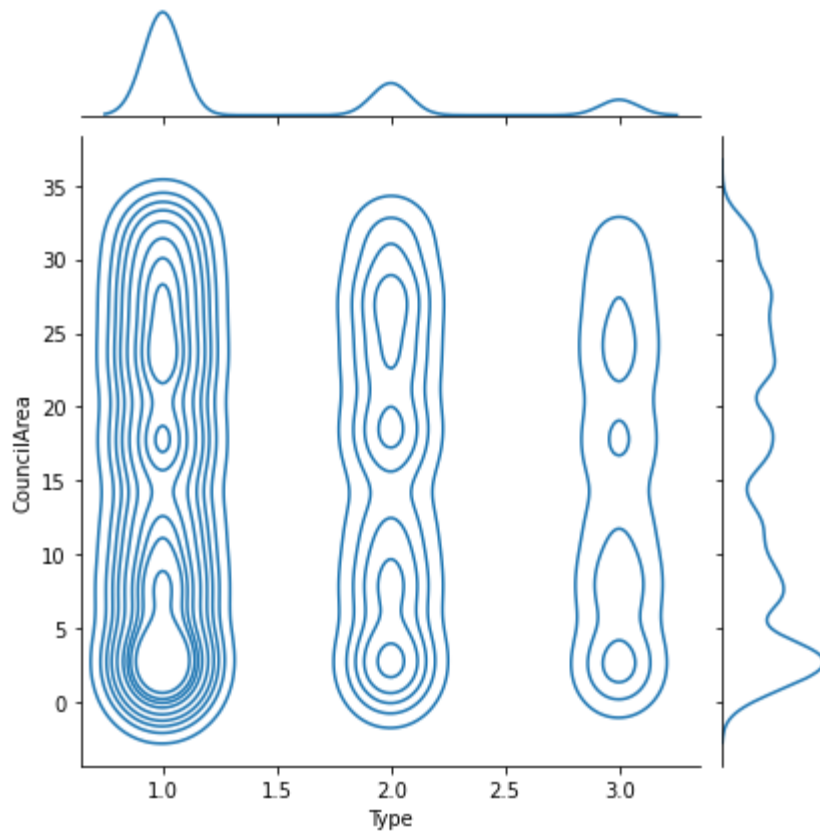
### Determining the relationship between two Attributes :



(a) ZN vs LSTAT

While comparing the proportion of residential land zoned for lots with % lower status of the population we noticed that places with 5% to 15% of lower status had -10 to 10 of the proportion of land zoned.



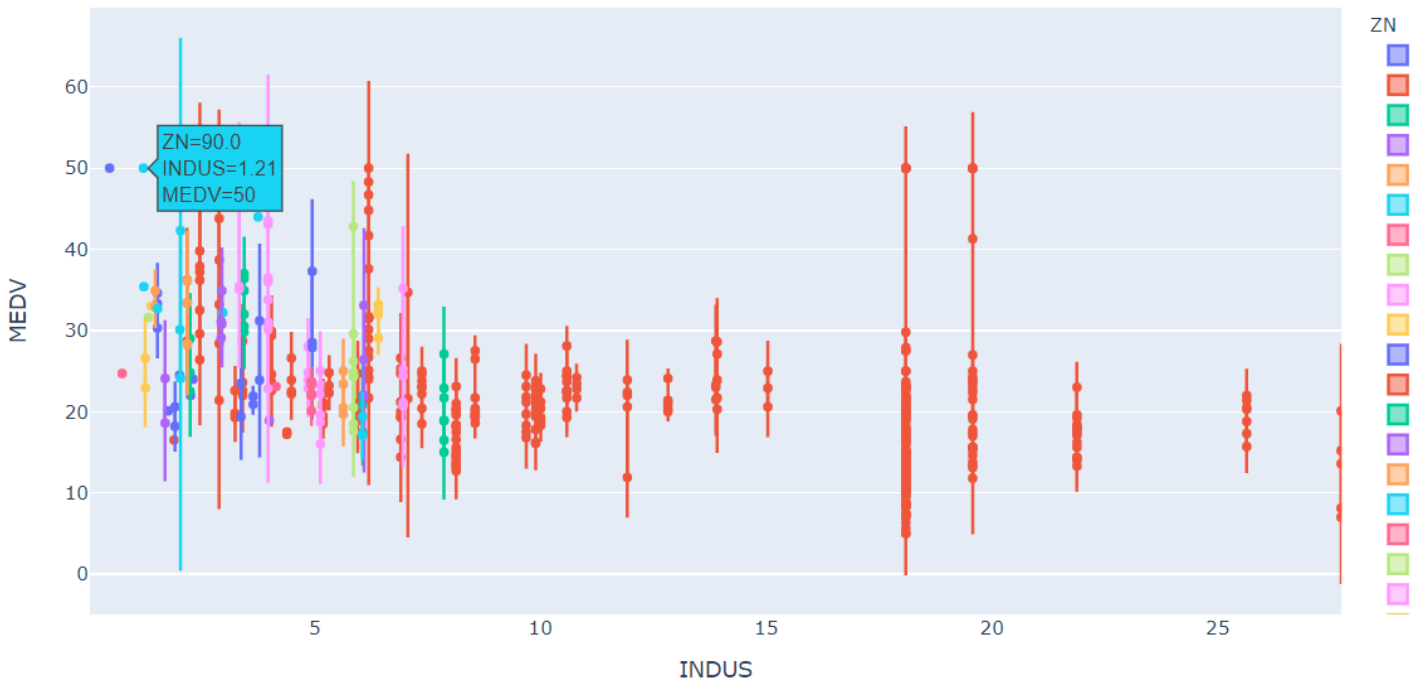


(b) Type of house vs Governing council for the area

Figure 5.6 (a & b) Attribute vs Attribute

It was observed that for type 1 most of the councils were 1st council(Banyule City Council) and the 2nd council(Bayside City Council) governing the houses and this was quite similar for type 2 and type 3.

## Few of the Factors Affecting the Price :



(a) Effect of ZN and INDUS on the Price(MEDV)

After a deep study of this violin chart it was noted that whenever there was a surge in ZN and a decline in INDUS the price skyrocketed also surprisingly if there was a surge in INDUS and ZN went near 0 the Price Increased again. This graph showed that both of these attributes were important and were not to be dropped. Also concluded that ZN was inversely Proportional to INDUS



(b) Effect of Landsize and Type on the Price

Figure 5.7 (a & b) Effect on label

It was observed that not the Landsize but the type of the house mattered the most while pricing the house and this shows how important it is to keep the type of the house in the dataset for yielding correct results.

## 5.4 Correlation between all attributes and the label :

### # Looking For Correlations

```
In [31]: #For making pearson correlation coefficient
# 1= strong positive correlation -1= strong negative correlation
# coeff> 0 with increase medv price increase eg: RM
# coeff< 0 with increase medv price decrease eg: CRIM
corr_matrix = housing.corr()
corr_matrix['MEDV'].sort_values(ascending=False)
```

```
Out[31]: MEDV      1.000000
RM       0.680857
B        0.361761
ZN       0.339741
DIS      0.240451
CHAS     0.205066
AGE     -0.364596
RAD     -0.374693
CRIM    -0.393715
NOX     -0.422873
TAX     -0.456657
INDUS   -0.473516
PTRATIO -0.493534
LSTAT   -0.740494
Name: MEDV, dtype: float64
```



(a) Correlation heat map

From above images we can see that RM(No of rooms) was highly positively correlated followed by B and LSTAT and PTRATIO were the two most negatively correlated attributes. This means that if the value of RM or B were increased the Price will increase and if PTRATIO or LSTAT were to increase the price would decrease.

## ▼ Looking For Correlations

```
[35] #For making pearson correlation coefficient
# 1= strong positive correlation -1= strong negative
# coeff> 0 with increase medv price increase eg: RM
# coeff< 0 with increase medv price decrease eg: CRIM
corr_matrix = housing.corr()
corr_matrix['Price'].sort_values(ascending=False)
```

```
Price          1.000000
Rooms          0.471397
Bedroom2       0.441021
Bathroom       0.440653
Car            0.210017
BuildingArea   0.092212
Method         0.053989
Landsize       0.043875
CouncilArea    -0.051692
Propertycount  -0.052155
Regionname     -0.108198
Distance       -0.204589
Type           -0.276615
YearBuilt      -0.330253
Name: Price, dtype: float64
```



(b) Correlation heat map

Figure 5.8 (a & b) Correlation between all attributes and the label

From the above images we can see that the attribute Rooms was highly positively correlated followed by Bedrooms and Type and YearBuilt were the two most negatively correlated attributes. This means that if the value of Rooms or Bedrooms were increased then the Price will increase and if Type or YearBuilt were to increase the price would decrease.

Now the data has been welcomed into our model and Preprocessing of the data has been done into our Regression model. On label encoding, training and testing on the data, We have arrived at the main goal of our project to predict if the student would be placed or not.

## 5.5 Machine Learning Models:

A set of algorithms were applied on the model to see which algorithm shows the best result and is more suitable and accurate.

1. **XGBoost**: The model is evaluated and the RMSE and Mean Cross Validation Scores by performing regression on the model is found.

### Testing The Model On Test Data

```
In [99]: X_test = strat_test_set.drop("MEDV", axis=1)
Y_test = strat_test_set["MEDV"].copy()
X_test_prepared = my_pipeline.transform(X_test) # x_test will be transformed by going through pipeline
final_predictions = model.predict(X_test_prepared) # will be predicted using random forest regressor(best option)
# final_predictions = rf_random.predict(X_test_prepared) # will be predicted using random forest regressor(best option)

final_mse = mean_squared_error(Y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
print(list(final_predictions))
print(list(Y_test))

[15.462904, 15.440298, 25.23592, 24.76033, 15.315966, 15.125524, 14.126881, 15.399217, 23.982582, 31.327154, 19.104845, 14.7681
13, 18.254274, 15.701168, 14.975743, 14.713916, 27.934967, 14.9854965, 36.57302, 16.244534, 16.928692, 20.239973, 16.672352, 2
5.118439, 18.11777, 34.71853, 16.90684, 47.2216, 14.768113, 31.08938, 10.536156, 15.595368, 27.383821, 15.236251, 16.062883, 1
4.673799, 29.95511, 19.737072, 17.702452, 45.86369, 27.12071, 22.074667, 16.23933, 18.85028, 24.044462, 32.41839, 34.62992, 15.
717798, 15.315994, 16.81451, 17.5214, 15.392921, 14.408202, 15.41171, 22.641308, 31.568277, 29.80175, 21.92621, 13.987278, 10.6
03747, 46.59596, 15.236251, 11.130643, 27.70081, 15.341451, 36.645096, 15.879411, 15.234028, 15.322627, 37.149498, 41.426525, 2
7.057344, 15.643321, 13.672273, 39.602627, 18.646193, 15.457759, 18.0272, 15.44423, 12.070372, 28.951632, 26.327797, 15.069501,
18.274244, 16.241203, 16.05394, 15.339492, 13.733832, 25.724903, 15.702539, 15.113274, 26.738127, 14.713916, 28.65097, 15.31599
4, 23.780384, 19.723635, 38.323563, 14.967384, 23.994383, 21.801645, 23.093866]
[16.5, 10.2, 30.1, 23.0, 14.4, 15.6, 19.4, 14.1, 30.3, 35.2, 23.1, 13.8, 25.0, 27.9, 19.5, 12.3, 32.2, 13.5, 23.8, 21.7, 19.2,
19.5, 10.4, 23.2, 18.6, 28.5, 15.2, 32.0, 7.2, 34.6, 20.1, 20.6, 23.6, 13.1, 23.8, 12.7, 43.1, 24.7, 22.2, 44.0, 28.1, 31.0, 2
1.7, 23.4, 19.5, 33.1, 41.7, 18.7, 19.9, 20.6, 21.2, 13.6, 20.3, 17.8, 27.1, 31.5, 50.0, 29.1, 18.9, 20.4, 50.0, 7.2, 17.2, 36.
2, 14.6, 33.2, 23.8, 19.9, 21.5, 37.3, 27.0, 22.0, 24.3, 19.8, 33.3, 7.0, 19.4, 20.9, 21.1, 20.4, 22.2, 11.9, 11.7, 21.6, 19.7,
23.0, 16.7, 21.7, 20.6, 23.3, 19.6, 28.0, 5.0, 24.4, 20.8, 24.8, 21.8, 23.6, 19.0, 25.0, 20.3, 21.5]
```

### Testing The Model On Test Data

```
X_test = strat_test_set.drop("Price", axis=1)
Y_test = strat_test_set["Price"].copy()
X_test_prepared = my_pipeline.transform(X_test) # x_test will be transformed by going through pipeline

final_predictions = model.predict(X_test_prepared) # will be predicted using random forest regressor(best option)
# final_predictions = classifier.predict(X_test_prepared) # will be predicted using random forest regressor(best option)

# final_predictions = rf_random.predict(X_test_prepared)

final_mse = mean_squared_error(Y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
print(list(final_predictions))
print(list(Y_test))

[942074.6, 1031283.56, 1380110.1, 457159.6, 792580.94, 1043985.5, 813438.9, 696936.1, 759577.94, 3007194.2, 864882.25, 418541.75, 1529303.5,
[975000.0, 850000.0, 1570000.0, 470000.0, 1175000.0, 1250000.0, 588000.0, 700000.0, 651500.0, 2940000.0, 986000.0, 350000.0, 1420000.0, 31000
4
```

Figure 5.9 (a & b) Predictions done by XGBoost Model

It was noted that XGBoost performed the best for both the datasets and yielded the Mean Cross Validation Scores of 0.88 for Boston(a) Dataset and 0.78 for Melbourne(b) Dataset.

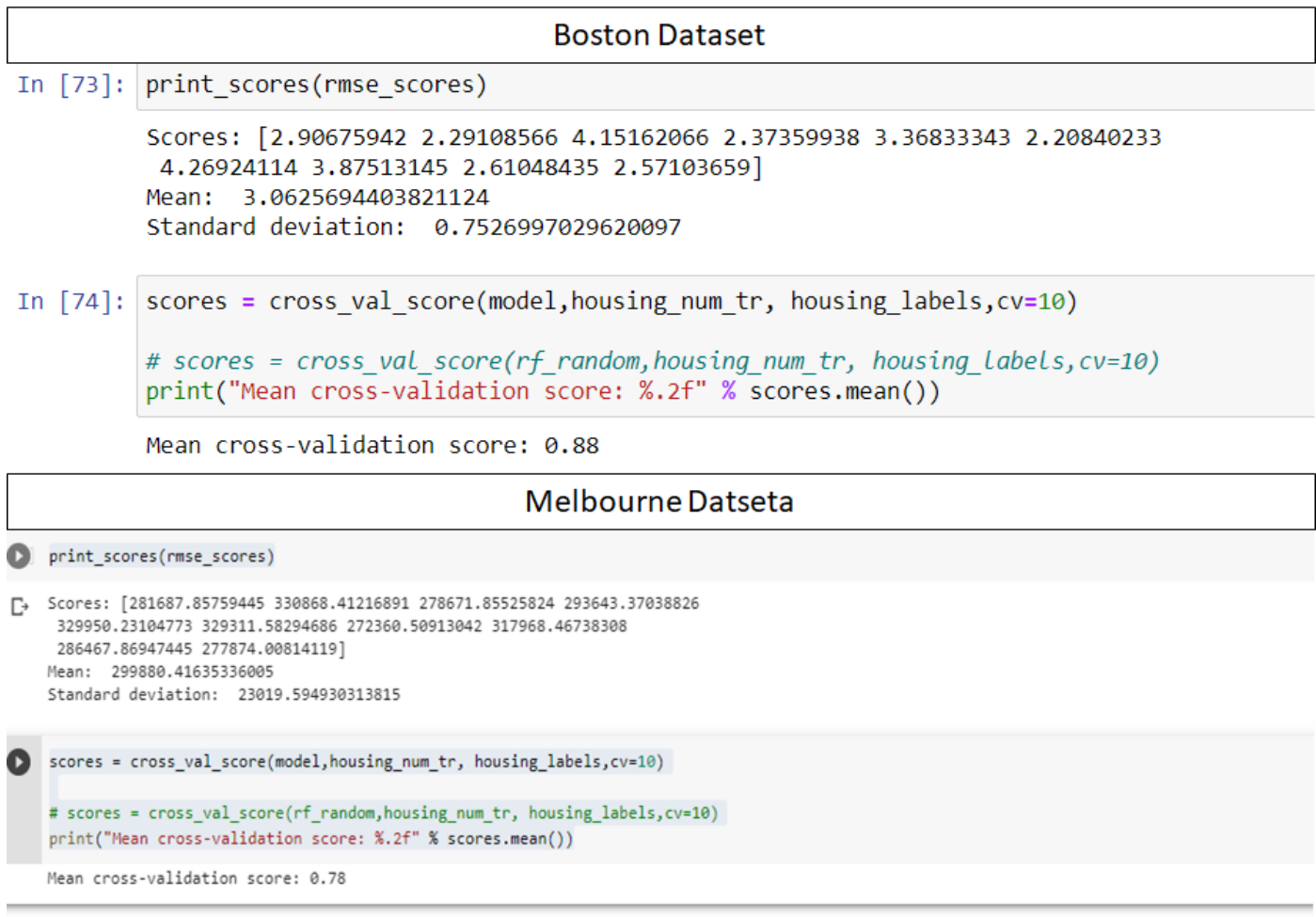


Figure 5.10 Evaluation of XGBoost Model

## 2. Decision Tree:

### Boston Dataset

#### Testing The Model On Test Data

```
X_test = strat_test_set.drop("MEDV", axis=1)
Y_test = strat_test_set["MEDV"].copy()
X_test_prepared = my_pipeline.transform(X_test) # x_test will be transformed by going through pipeline
final_predictions = model.predict(X_test_prepared) # will be predicted using random forest regressor(best option)
# final_predictions = rf_random.predict(X_test_prepared) # will be predicted using random forest regressor(best option)

final_mse = mean_squared_error(Y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
print(list(final_predictions))
print(list(Y_test))
```

[28.4, 10.9, 29.0, 30.7, 20.0, 13.8, 22.4, 11.5, 32.7, 43.5, 21.7, 10.9, 21.6, 23.1, 19.0, 11.3, 32.7, 15.2, 25.0, 17.3, 19.6, 15.2, 10.9, 21.7, 16.1, 33.1, 17.1, 33.1, 10.5, 32.7, 20.0, 20.6, 23.9, 10.9, 22.3, 8.3, 48.8, 25.3, 20.7, 46.0, 25.0, 30.7, 19.5, 19.4, 14.3, 33.2, 48.3, 19.0, 20.2, 23.1, 22.7, 13.1, 22.4, 17.1, 28.6, 31.7, 37.6, 30.8, 19.4, 19.0, 50.0, 5.0, 17.5, 23.9, 18.1, 32.7, 17.4, 13.4, 19.9, 36.4, 24.3, 22.6, 18.5, 22.6, 32.7, 13.3, 15.6, 19.4, 19.0, 21.0, 23.9, 19.0, 13.4, 23.9, 22.8, 21.2, 8.7, 22.0, 18.9, 22.2, 17.5, 22.0, 5.6, 26.6, 23.1, 31.1, 24.5, 28.7, 8.5, 28.6, 22.0, 18.8]

[16.5, 10.2, 30.1, 23.0, 14.4, 15.6, 19.4, 14.1, 30.3, 35.2, 23.1, 13.8, 25.0, 27.9, 19.5, 12.3, 32.2, 13.5, 23.8, 21.7, 19.2, 19.5, 10.4, 23.2, 18.6, 28.5, 15.2, 32.0, 7.2, 34.6, 20.1, 20.6, 23.6, 13.1, 23.8, 12.7, 43.1, 24.7, 22.2, 44.0, 28.1, 31.0, 21.7, 23.4, 19.5, 33.1, 41.7, 18.7, 19.9, 20.6, 21.2, 13.6, 20.3, 17.8, 27.1, 31.5, 50.0, 29.1, 18.9, 20.4, 50.0, 7.2, 17.2, 36.

### Melbourne Dataset

#### Testing The Model On Test Data

```
[80] X_test = strat_test_set.drop("Price", axis=1)
Y_test = strat_test_set["Price"].copy()
X_test_prepared = my_pipeline.transform(X_test) # x_test will be transformed by going through pipeline

final_predictions = model.predict(X_test_prepared) # will be predicted using random forest regressor(best option)
# final_predictions = classifier.predict(X_test_prepared) # will be predicted using random forest regressor(best option)

# final_predictions = rf_random.predict(X_test_prepared)

final_mse = mean_squared_error(Y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
print(list(final_predictions))
print(list(Y_test))
```

[810000.0, 931000.0, 1405000.0, 506000.0, 700000.0, 1050000.0, 650000.0, 547000.0, 959000.0, 3500000.0, 700000.0, 360000.0, 1890000.0, 2005000.0, 496388.8888888889, 688000.0, 570000.0, 975000.0, 850000.0, 1570000.0, 470000.0, 1175000.0, 1250000.0, 588000.0, 700000.0, 651500.0, 2940000.0, 986000.0, 350000.0, 1420000.0, 3100000.0, 605000.0, 599000.0, 450000.0, 1710000.0]

Figure 5.11 (a & b) Predictions done by Decision Tree Model



### 3. Linear Regression:

#### Boston Dataset

##### Testing The Model On Test Data

```
X_test = strat_test_set.drop("MEDV", axis=1)
Y_test = strat_test_set["MEDV"].copy()
X_test_prepared = my_pipeline.transform(X_test) # x_test will be transformed by going through pipeline
final_predictions = model.predict(X_test_prepared) # will be predicted using random forest regressor(best option)
# final_predictions = rf_random.predict(X_test_prepared) # will be predicted using random forest regressor(best option)

final_mse = mean_squared_error(Y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
print(final_predictions)
print(list(Y_test))
```

[22.69968907 17.21809545 30.02249279 30.74778916 8.89512081 13.32142051  
17.33667356 17.75884279 32.49829759 36.03953505 16.35542057 0.56736309  
23.00036857 20.45406323 20.07195268 12.94763771 31.15259787 13.42929671  
25.03377998 24.16484366 20.41376724 17.03371972 17.78872127 25.59396216  
19.48925131 32.82684685 19.43986145 33.71211148 8.03165673 34.67822079  
19.55672166 21.44813056 29.29990156 16.34255866 26.9868503 18.36287062  
37.29070365 24.57516862 22.25473001 37.13556596 25.15387036 34.46727484  
23.46332441 24.04296023 18.50218667 32.68878654 38.45633313 21.41849048  
17.66121404 16.30911908 21.22360232 12.41720203 19.93411198 20.41550492  
27.9774864 33.08886348 40.09861375 31.36242042 14.94917745 19.75561251  
40.46388017 18.11130418 15.18135521 27.6464993 19.49913588 32.4746605  
23.41651711 20.31958551 21.17791079 33.77535007 34.10374558 27.61946367  
24.52197711 21.88353054 36.17789977 8.56108814 17.44322967 21.51365153  
20.57689904 23.01502065 25.9813584 22.54880065 14.17517708 25.4719191  
21.27124207 23.81881373 19.89012048 21.73051343 22.47067585 21.74229837

#### Melbourne Dataset

##### Testing The Model On Test Data

```
[80] X_test = strat_test_set.drop("Price", axis=1)
Y_test = strat_test_set["Price"].copy()
X_test_prepared = my_pipeline.transform(X_test) # x_test will be transformed by going through pipeline

final_predictions = model.predict(X_test_prepared) # will be predicted using random forest regressor(best option)
# final_predictions = classifier.predict(X_test_prepared) # will be predicted using random forest regressor(best option)

# final_predictions = rf_random.predict(X_test_prepared)

final_mse = mean_squared_error(Y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
print(list(final_predictions))
print(list(Y_test))
```

[1260495.4378664233, 1042223.9254275924, 1305690.2432319922, 722121.2241895171, 802428.2999665327, 986376.4544497868, 958673.2029083064, 698  
[975000.0, 850000.0, 1570000.0, 470000.0, 1175000.0, 1250000.0, 588000.0, 700000.0, 651500.0, 2940000.0, 986000.0, 350000.0, 1420000.0, 3100

Figure 5.12 (a & b) Predictions done by Linear Regression Model

#### 4. Random Forest Regression:

### Boston Dataset

#### Testing The Model On Test Data

```
X_test = strat_test_set.drop("MEDV", axis=1)
Y_test = strat_test_set["MEDV"].copy()
X_test_prepared = my_pipeline.transform(X_test) # x_test will be transformed by going through pipeline
final_predictions = model.predict(X_test_prepared) # will be predicted using random forest regressor(best option)
# final_predictions = rf_random.predict(X_test_prepared) # will be predicted using random forest regressor(best option)

final_mse = mean_squared_error(Y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
print(final_predictions)
print(list(Y_test))
```

[24.778 11.63 25.885 22.151 18.513 15.287 19.876 14.267 32.725 42.526  
19.365 11.633 24.281 25.352 19.515 11.097 31.777 14.229 23.615 19.558  
20.133 17.812 16.781 22.226 18.734 31.877 16.18 33.234 8.524 33.535  
23.716 21.46 22.563 10.706 20.564 11.239 43.859 24.442 23.645 41.804  
24.163 29.72 20.896 20.646 18.758 32.443 44.825 20.432 19.797 21.343  
21.121 14.741 21.171 15.021 25.355 32.8 41.257 28.741 19.695 20.859  
46.673 9.894 19.253 25.224 14.911 33.362 18.908 18.552 18.813 33.564  
25.971 22.696 21.76 22.677 34.144 12.964 15.539 19.972 20.627 21.538  
22.711 21.513 14.146 22.948 20.532 21.145 13.973 21.197 21.892 23.834  
18.732 27.192 7.369 27.025 18.342 29.339 19.456 31.463 14.639 26.428  
21.367 20.385]  
[16.5, 10.2, 30.1, 23.0, 14.4, 15.6, 19.4, 14.1, 30.3, 35.2, 23.1, 13.8, 25.0, 27.9, 19.5, 12.3, 32.2, 13.5, 23.8, 21.7, 19.2,  
19.5, 10.4, 23.2, 18.6, 28.5, 15.2, 32.0, 7.2, 34.6, 20.1, 20.6, 23.6, 13.1, 23.8, 12.7, 43.1, 24.7, 22.2, 44.0, 28.1, 31.0, 2  
1.7, 23.4, 19.5, 33.1, 41.7, 18.7, 19.9, 20.6, 21.2, 13.6, 20.3, 17.8, 27.1, 31.5, 50.0, 29.1, 18.9, 20.4, 50.0, 7.2, 17.2, 36.  
2, 14.6, 33.2, 23.8, 19.9, 21.5, 37.3, 27.0, 22.0, 24.3, 19.8, 33.3, 7.0, 19.4, 20.9, 21.1, 20.4, 22.2, 11.9, 11.7, 21.6, 19.7,  
23.0, 16.7, 21.7, 20.6, 23.3, 19.6, 28.0, 5.0, 24.4, 20.8, 24.8, 21.8, 23.6, 19.0, 25.0, 20.3, 21.5]

### Melbourne Dataset

#### Testing The Model On Test Data

```
[80] X_test = strat_test_set.drop("Price", axis=1)
Y_test = strat_test_set["Price"].copy()
X_test_prepared = my_pipeline.transform(X_test) # x_test will be transformed by going through pipeline

final_predictions = model.predict(X_test_prepared) # will be predicted using random forest regressor(best option)
# final_predictions = classifier.predict(X_test_prepared) # will be predicted using random forest regressor(best option)

# final_predictions = rf_random.predict(X_test_prepared)

final_mse = mean_squared_error(Y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
print(list(final_predictions))
print(list(Y_test))
```

[874175.0, 939900.0, 1490810.0, 565715.0, 751771.6666666665, 1228365.0, 785848.3333333333, 540716.1547619047, 730971.5, 3275610.0, 88352  
[975000.0, 850000.0, 1570000.0, 470000.0, 1175000.0, 1250000.0, 588000.0, 700000.0, 651500.0, 2940000.0, 986000.0, 350000.0, 1420000.0,

Figure 5.13 (a & b) Predictions done by Random Forest Regression Model

## 5. Gradient Boosting:

### Melbourne Dataset

#### • Testing The Model On Test Data

```
[80] X_test = strat_test_set.drop("Price", axis=1)
     Y_test = strat_test_set["Price"].copy()
     X_test_prepared = my_pipeline.transform(X_test) # x_test will be transformed by going through pipeline

     final_predictions = model.predict(X_test_prepared) # will be predicted using random forest regressor(best option)
     # final_predictions = classifier.predict(X_test_prepared) # will be predicted using random forest regressor(best option)

     # final_predictions = rf_random.predict(X_test_prepared)

     final_mse = mean_squared_error(Y_test, final_predictions)
     final_rmse = np.sqrt(final_mse)
     print(list(final_predictions))
     print(list(Y_test))

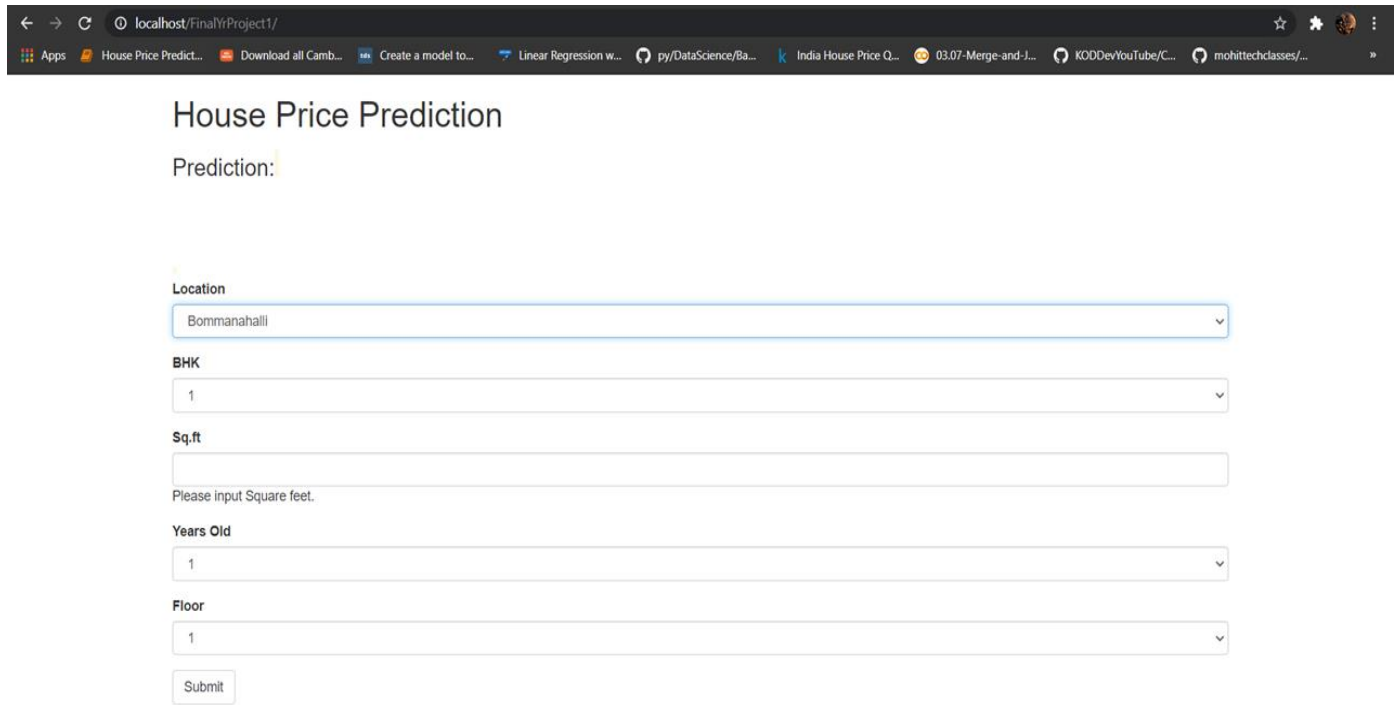
[1098823.3402773298, 1027005.9824405565, 1384815.8155619227, 496885.3484099893, 834783.362830733, 838874.9701549627, 814945.398781418, 824845.7444511683, 661166.4394087
[975000.0, 850000.0, 1570000.0, 470000.0, 1175000.0, 1250000.0, 588000.0, 700000.0, 651500.0, 2940000.0, 986000.0, 350000.0, 1420000.0, 3100000.0, 605000.0, 599000.0, 4
```

Figure 5.14 (a) Predictions done by Gradient Boosting Model

All Predictions CSV: <https://drive.google.com/drive/folders/11uXlwlhQZ40yjT-MTLupM7S-bspEjFer?usp=sharing>

## 5.5 Developing a User Interface(work in progress):

Currently the UI is still in development and is not yet complete here's a screenshot of an extremely rough UI. This is nowhere near the real web application we plan to build. However, as of now the work for the web app is done this much



The screenshot shows a web browser window with the address bar displaying 'localhost/FinalYrProject1/'. The browser's tab bar shows several open tabs, including 'House Price Predict...', 'Download all Camb...', 'Create a model to...', 'Linear Regression w...', 'py/DataScience/Ba...', 'India House Price Q...', '03.07-Merge-and-J...', 'KODDevYouTube/C...', and 'mohittechclasses/'. The main content area of the browser displays a form titled 'House Price Prediction'. Below the title is a label 'Prediction:' followed by a yellow cursor. The form contains several input fields: a dropdown menu for 'Location' with 'Bommanahalli' selected, a dropdown menu for 'BHK' with '1' selected, a text input field for 'Sq.ft' with the placeholder text 'Please input Square feet.', a dropdown menu for 'Years Old' with '1' selected, and a dropdown menu for 'Floor' with '1' selected. At the bottom of the form is a 'Submit' button.

Figure 5.15 Website(Undeveloped)

## **Chapter 6**

### **CONCLUSION AND FUTURE WORK**

Buying your own house is a dream each and every human wishes for . Using this proposed model we want people to buy houses and real estates at their right prices and want to ensure that the don't get tricked by sketchy agents who just are after their money. Additionally, this model will also help Big companies by giving accurate predictions for them to set the pricing and save them from a lot of hassle and save a lot of precious time and money. Correct real estate prices are the essence of the market and we want to ensure that by using this model.

The system is apt enough in training itself and in predicting the future from the raw data provided to it in a layman's term. After going through several literature surveys and numerous blogs and articles a set of algorithms were selected that was found suitable in applying on both the datasets of the model. After several testings and training it was determined that the XGBoost Algorithm showed the best result amongst the rest of the algorithms. The system was potent enough for Predicting the prices of different houses with various features and was able to handle large sums of data. The system is quite user-friendly and time saving.

The supplementary features that can be added in our proposed system is to avail users with an user interface so they can enter all the attributes of their houses and just with one click get the prediction for their house. Also, an Amazon EC2 connection will take the system even further and increase the ease of use. Lastly, developing a well integrated web application which can predict prices whenever users want it to will complete the project.

## REFERENCES

1. Thuraiya Mohd, Suraya Masrom, Noraini Johari , "Machine Learning Housing Price Prediction in Petaling Jaya, Selangor, Malaysia " , 2019.
2. G. Naga Satish, Ch. V. Raghavendran, M.D.Sugnana Rao, Ch.Srinivasulu , "House Price Prediction Using Machine Learning" , 2019.
3. Alisha Kuvalekar , Shivani Manchewar , Sidhika Mahadik, "House Price Fore Casting Using Machine Learning" , 2018.
4. Neelam Shinde, Kiran Gawande , "Valuation Of House Prices Using Predictive Techniques" , 2018.
5. Jingyi Mu, Fang Wu,and Aihua Zhang , " Housing Value Forecasting Based on Machine Learning Methods" , 2014.
6. Sayan Putatunda , "PropTech for Proactive Pricing of Houses in Classified Advertisements in the Indian Real Estate Market" ,
7. Atharva Chouthai, Mohammed Athar Rangila , Sanved Amate, Prayag Adhikari, Vijay Kukre , "House Price Prediction Using Machine Learning" , 2019.
8. B.Balakumar, P.Raviraj, S.Essakkiammal , "Predicting Housing Prices using Machine Learning Techniques" ,
9. Akshay Babu, Dr. Anjana S Chandran , "Literature Review on Real Estate Value Prediction Using Machine Learning" , 2019.
10. Mr. Rushikesh Naikare, Mr. Girish Gahandule, Mr. Akash Dumbre, Mr. Kaushal Agrawal, Prof. Chaitanya Manka , "House Planning and Price Prediction System using Machine Learning" , 2019.
11. Aswin Sivam Ravikumar , "Real Estate Price Prediction Using Machine Learning" , 2016.

12. Bindu Sivasankar, Arun P. Ashok, Gouri Madhu, Fousiya S , "House Price Prediction" , 2020.
13. Dr. M. Thamarai ,Dr. S P. Malarvizhi , "House Price Prediction Modeling Using Machine Learning" , 2020.
14. <https://medium.com/topic/data-science>
15. <https://www.geeksforgeeks.org/machine-learning/>
16. <https://stackoverflow.com/questions/tagged/machine-learning>
17. <https://towardsdatascience.com/>
18. [https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html)
19. <https://jupyter-notebook.readthedocs.io/en/stable/>
20. <https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/01.01-Help-And-Documentation.ipynb>