

INDUSTRIAL TRAINING

*Submitted in partial fulfilment of the
Requirements of the degree*

Of

Bachelor of technology

In

Information Technology & Engineering

By:

Aadika Bhatia(00213203120/IT-1/2020-24)

Under the guidance of:

Dr. Amandeep Kaur

Ms. Bhavneet Kaur



Department of Information Technology & Engineering

Guru Tegh Bahadur Institute of Technology

Guru Gobind Singh Indraprastha University

Dwarka, New Delhi

Year 2020-2024

Real Estate Price Prediction Using Machine Learning

Duration

18th August, 2022 – 18th September, 2022

By:

Aadika Bhatia(00213203120/IT-1/2020-24)

At:

UDEMY

By Star tech academy

DEVTOWN

By Mr. Shaurya Sinha

DECLARATION

I hereby declare that all the work presented in this Industrial Training Report for the partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in **Information Technology & Engineering**, Guru Tegh Bahadur Institute of Technology, Guru Govind Singh Indraprastha University, New Delhi is an authentic record of our own work carried out at UdeMy and Devtown by Star Tech Academy and Mr. Shaurya Sinha respectively from 18th August , 2022 to 18th September, 2022.

Date:

Aadika Bhatia(00213203120/IT-1/2020-24)



Certificate no: UC-3308c41b-93cb-485a-9ee3-596744f8ef111
Certificate url: ude.my/UC-3308c41b-93cb-485a-9ee3-596744f8ef111
Reference Number: 0006

CERTIFICATE OF COMPLETION

Machine Learning using Python

Instructor **Start-Tech Academy**

Aadika Bhatia

Date **Sept. 12, 2022**

Length **18.5 total hours**



Certificate of Completion



devtown

Presented to

AADIKA BHATIA

for completing 7-days Bootcamp on
Real Estate Price Prediction



SHAURYA SINHA
CO-FOUNDER

Scan the QR code to verify



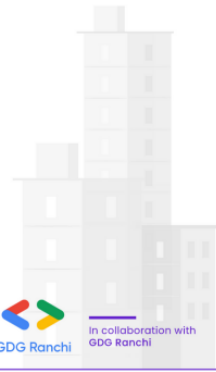
or visit <https://cert.devtown.in/tech/verify/710948>



In collaboration with
Microsoft Learn



In collaboration with
GDG Ranchi



Microsoft Learn Student Ambassador

This certificate is presented to:

AADIKA BHATIA

In recognition of your attendance and completion of the
Microsoft Student Ambassadors

**Real Estate Price Prediction
Bootcamp**

Event Hosted By
Microsoft Learn Student Ambassador

Scan the QR code to verify



or visit <https://cert.devtown.in/tech/verify/929908>



ACKNOWLEDGEMENT

I would like to express our great gratitude towards our supervisor, Dr. Amandeep Kaur and Ms. Bhavneet Kaur who has given us support and suggestions. Without their help we could not have presented this dissertation up to the present standard. We also take this opportunity to give thanks to all others who gave us support for the project or in other aspects of our study at Guru Tegh Bahadur Institute Of Technology.

Date: 18/01/2023

Aadika Bhatia

(00213203120/IT-1/2024)

aadikabhatia30@gmail.com

ABSTRACT

House price prediction is an important aspect for real estate. It all revolves around the accurate prediction of the prices so as to sell or buy the property at an accurate rate without major losses. Machine learning techniques play an important role in the prediction using the datasets and applying various algorithms to determine the price of the real estate at affordable prices. Moreover, based on the features in the dataset the prices will be predicted using linear regression and supervised learning.

LIST OF FIGURES AND TABLES

Fig no.	Figure Name	Page
1.	Machine learning block diagram	xii
2.	Basic plots	xxix
3.	Arrays and fields plots	xxx
4.	Statistics plot	xxxix
5.	Unstructured plot	xxxix
6.	Seaborn pair plot	xxxiv
7.	Supervised learning	xxxvii
8.	Types of supervised learning	xxxviii
9.	Unsupervised learning	xl
10.	Types of Unsupervised learning	xlxi
11.	Steps in ML	xlxi
12.	Data collection	xliv
13.	Data preparation	xlvi
14.	Train a model	xlvi
15.	Analyse the data	xlvi
16.	Use case	lii
17.	DFD	liii
18.	ER	liv
19.	Functions	lxii
20.	Functions	lxiii
21.	Functions	lxiii
22.	Functions	lxiv
23.	Functions	lxiv
24.	Functions	lxv
25.	Heatwave	lxvi
26.	Pair plot	lxvii
27.	Result	lxviii

CONTENTS

Chapter	Page no.
Title page	ii
Declaration	iii
Certificate	iv
Acknowledgement	vi
Abstract	vii
Tables and Figures	viii
1. Introduction	xi
1.1. Introduction	xii
1.2. What system consists of?	xiii
1.3. Need of system	xiv
2. Hardware and Software Requirement	xv
2.1. Introduction	xvi
2.2. System requirement	xvii
2.3. Hardware and software specification	xviii
3. System Analysis	xix
3.1. Purpose	xx
3.2. Project scope	xx
3.3. Existing system	xxi
3.4. Proposed system	xxi
3.5. System overview	xxi
4. Tools and Technologies	xxii
4.1. NumPy	xxiii
4.2. Pandas	xxvi
4.3. Matplotlib	xxviii
4.4. Seaborn	xxxiii
4.5. Sklearn	xxxvi
4.6. Machine Learning	xxxvii
5. System Design	li
5.1. Introduction	lii

5.2. Use case	liii
5.3. DFD	liv
5.4. ER	lv
5.5. System Testing	lvi
5.5.1 Unit Testing	lvi
5.5.2 Integration Testing	lvii
6. Conclusion	lviii
6.1. Conclusion	lix
6.2. Future scope	lix
References	lx
Appendix A	lxii
Appendix B	lxx

Introduction

INTRODUCTION TO THE PROJECT:

Thousands of houses are sold everyday. There are some questions every buyer asks himself like: What is the actual price that this house deserves? Am I paying a fair price? In this project, a machine learning model is proposed to predict a house price based on data related to the house (its size, the year it was built in, no. of rooms etc.). In earlier time people used to seek guidance from experts of real estates for prediction of house prices, but as we know human estimation is not much reliable. Using machine learning model we could increase the accuracy of the price prediction and give a fair result to the customer and the real estate companies. Here linear regression and supervised learning will be used.

This report focuses on Real Estate price prediction model that I've made under a bootcamp provided by Devtown.

This bootcamp focused on python, it's different libraries and machine learning algorithms and where they all can be used.

WHAT THE SYSTEM MAINLY CONSISTS OF:

Machine Learning

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of Machine Learning.

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machines build the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:

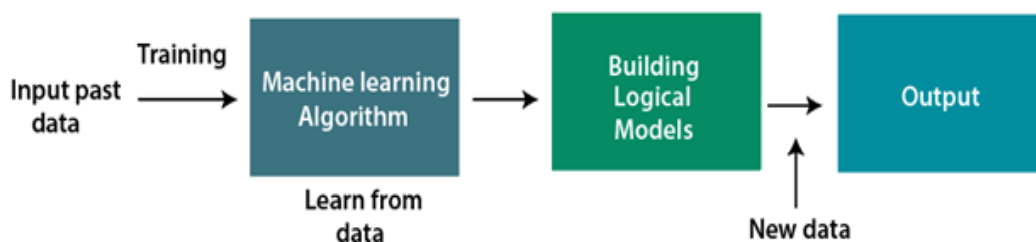


Figure 1

Over the last couple of decades, the technological advances in storage and processing power have enabled some innovative products based on machine learning, such as Netflix's recommendation engine and self-driving cars.

Need for Machine Learning

As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy for us. We can train machine learning algorithms by providing them the huge amount of data and let them explore the data, construct the models, and predict the required output automatically.

Following are some key points which show the importance of Machine Learning:

- Rapid increment in the production of data
- Solving complex problems, which are difficult for a human
- Decision making in various sector including finance
- Finding hidden patterns and extracting useful information from data.

HARDWARE AND SOFTWARE REQUIREMENTS

INTRODUCTION

In this chapter we mentioned the software and hardware requirements, which are necessary for successfully running this system. The major element in building systems is selecting compatible hardware and software. The system analyst has to determine what software package is best for “The Real Estate Price Prediction Model” and, where software is not an issue, the kind of hardware and peripherals needed for the final conversion.

SYSTEM ENVIRONMENT:

After analysis, some resources are required to convert the abstract system into the real one. The hardware and software selection begins with requirement analysis, followed by a request for proposal and vendor evaluation.

Software and real system are identified. According to the provided functional specification all the technologies and its capacities are identified.

Basic functions and procedures and methodologies are prepared to implement.

Some of the Basic requirements such as hardware and software are described as follows: -

HARDWARE AND SOFTWARE SPECIFICATION:

SOFTWARE REQUIREMENTS:

- Operating systems : Windows 10
- Coding Language : Python 3.8
- Web Framework : Google colab, Jupyter

HARDWARE REQUIREMENTS:

- System : Pentium i3 Processor
- Hard Disk : 500 GB
- Monitor : 15” LED
- Input Devices : Keyboard, Mouse
- Ram : 4 GB

SYSTEM ANALYSIS

PURPOSE:

The purpose of developing the real estate price prediction model is to determine the actual prices of the houses accurately for their purchase. It helps the seller and buyer get a reasonable price for their house.

PROJECT SCOPE:

There are many people around the world investing in real estate. Buying and selling of properties has always been a trend , a kind of investment or a income to humans. Everyone at times want to buy the estates at lower prices and sell at higher prices for money. But how can they do it. Sometimes they think the worth of their property is less and sell it in lower prices at a certain loss or sometimes they set it at such a high price that the people seem less interested in buying it from them.

What if we had a system to determine the actual price of the houses without any kind of cloudy assumptions and based on real life truth. Using Machine Learning algorithms it is possible in a much easier way. It analyse the data and gives a factual result on which everyone could agree on without any assumption or dispute.

The project has a wide scope, as it is not intended to a particular organization. This project is going to develop generic software, which can be applied by any businesses organization. Moreover it provides facility to its users. Also the software is going to provide a huge amount of summary data.

EXISTING SYSTEM:

At present the real estate is a market of estimation. Everything works only on human assumptions which is at times riskier as it may lead them to higher losses or no selling of their properties. As we know it depends on various factor like no. of rooms, sq. foot, locality etc. keeping all these in mind and applying machine learning algorithms on them gives a actual price which may help them to get accurate prices.

PROPOSED SYSTEM:

In this machine learning project we will train the model and provide it with the required dataset and labels. Using supervised learning and linear regression it will predict the reasonable prices.

SYSTEM OVERVIEW:

In this system the main component is the dataset. A fully labelled dataset will be given to train and using linear regression and algorithms the prices will be detected on the basis of parameters such at locality, nitric oxide in soil, no. of rooms, square feet etc.

A heat map and graph will be produced consisting of the results.

TOOLS AND TECHNOLOGIES

1. NumPy



NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the `ndarray` object. This encapsulates n -dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an `ndarray` will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python,

including NumPy) objects, thereby allowing for arrays of different sized elements.

- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behaviour is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also it is optimized to work with the latest CPU architectures.
- NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays.

In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays. The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with the corresponding element in another sequence of the same length. If the data are stored in two Python lists, *a* and *b*, we could iterate over each element:

```
c = []
for i in range(len(a)): c.append(a[i]*b[i])
```

This produces the correct answer, but if *a* and *b* each contain millions of numbers, we will pay the price for the inefficiencies of looping in Python. We could accomplish the same task much more quickly in C by writing (for clarity we neglect variable declarations and initializations, memory allocation, etc.)

```
for (i = 0; i < rows; i++) {
```



```
c[i] = a[i]*b[i];  
}
```

This saves all the overhead involved in interpreting the Python code and manipulating Python objects, but at the expense of the benefits gained from coding in Python.

Furthermore, the coding work required increases with the dimensionality of our data. In the case of a 2-D array, for example, the C code (abridged as before) expands to

```
for (i = 0; i < rows; i++) {  
  for (j = 0; j < columns; j++) {  
    c[i][j] = a[i][j]*b[i][j]; }  
}
```

2. PANDAS



Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis/manipulation tool available in any language. It is already well on its way toward this goal. Pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet.
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels.
- Any other form of observational / statistical data sets. The data need not be labelled.

at all to be placed into a pandas data structure.

The two primary data structures of pandas, **Series** (1-dimensional) and **DataFrame** (2-

dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, Data Frame provides everything that R's data frame provides and much more. Pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries. Here are just a few of the things that pandas does well:

- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data.
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects.
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations.
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data.
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects.
- Time series-specific functionality: date range generation and frequency conversion, moving window statistics, date shifting, and lagging.

3. MATPLOTLIB



Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications. A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot. The matplotlib scripting layer overlays two APIs:

- The **pyplot** API is a hierarchy of Python code objects topped by **matplotlib.pyplot**
- An OO (Object-Oriented) API collection of objects that can be assembled with greater flexibility than pyplot. This API provides direct access to Matplotlib's backend layers.

Matplotlib makes hard things possible and things easy .

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in Jupyter Lab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

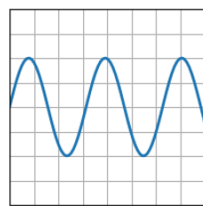
Matplotlib and Pyplot in Python

The pyplot API has a convenient MATLAB-style stateful interface. In fact, matplotlib was originally written as an open source alternative for MATLAB. The OO API and its interface is more customizable and powerful than pyplot, but considered more difficult to use. As a result, the pyplot interface is more commonly used.

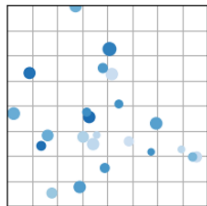
Now let's look at the various types of plots available in matplotlib :

Basic plots

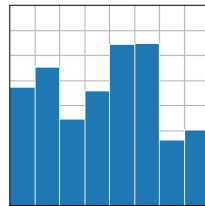
Basic plot types, usually y versus x.



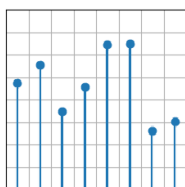
`plot(x, y)`



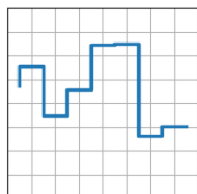
`scatter(x, y)`



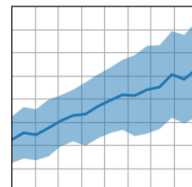
`bar(x, height)`



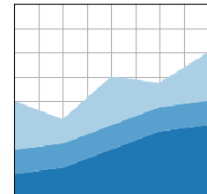
`stem(x, y)`



`step(x, y)`



`fill_between(x, y1, y2)`

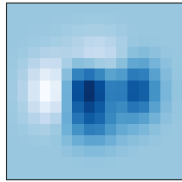


`stackplot(x, y)`

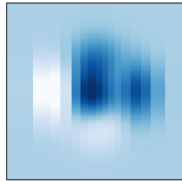
Figure 2

Plots of arrays and fields

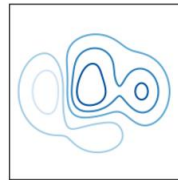
Plotting for arrays of data $Z(x, y)$ and fields $U(x, y)$, $V(x, y)$.



`imshow(Z)`



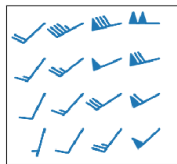
`pcolormesh(X, Y, Z)`



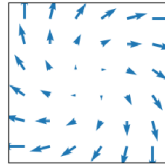
`contour(X, Y, Z)`



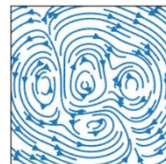
`contourf(X, Y, Z)`



`barbs(X, Y, U, V)`



`quiver(X, Y, U, V)`

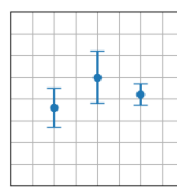
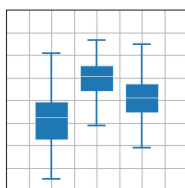
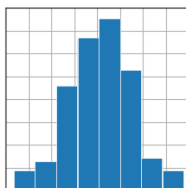


`streamplot(X, Y, U, V)`

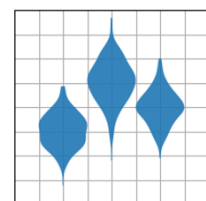
Figure 3

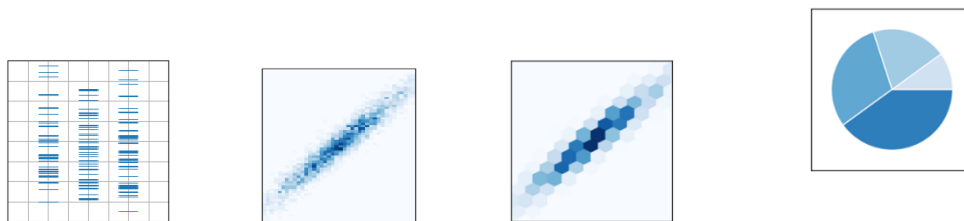
Statistics plots

Plots for statistical analysis.



`hist(x)`
`boxplot(X)`
`errorbar(x, y, yerr,`
`xerr)` `violinplot(D)`





eventplot(D)

hist2d(x, y)

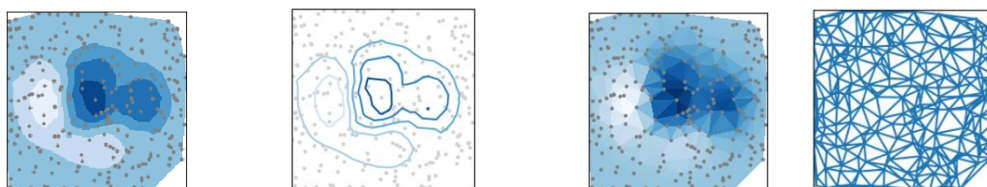
hexbin(x, y, C)

pie(x)

Figure 4

Unstructured coordinates

Sometimes we collect data z at coordinates (x,y) and want to visualize as a contour. Instead of gridding the data and then using contour, we can use a triangulation algorithm and fill the triangles.



Tricontour(x, y, z)

tricontourf(x, y, z)

tripcolor(x, y, z)

triplot(x,y)

Figure 5

4. SEABORN



Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for the same variables for better understanding of the dataset. Different categories of plot in Seaborn: Plots are basically used for visualizing the relationship between variables.

Those variables can either be completely numerical or a category like a group, class or division. Seaborn divides plot into the below categories –

- **Relational plots:** This plot is used to understand the relation between two variables.
- **Categorical plots:** This plot deals with categorical variables and how they can be visualized.
- **Distribution plots:** This plot is used for examining univariate and bivariate distributions
- **Regression plots:** The regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analysis.
- **Matrix plots:** A matrix plot is an array of scatterplots.
- **Multi-plot grids:** It is a useful approach to draw multiple instances of the same plot on different subsets of the dataset.
- **Pair Plot :**

The Seaborn Pairplot function allows the users to create an axis grid via which each numerical variable stored in data is shared across the X- and Y-axis in the structure of columns and rows. We can create the Scatter plots in order to display the pairwise relationships in addition to the distribution plot displaying the data distribution in the column diagonally. The pairplot() function can also be used to showcase the subset of variables, or we can plot different types of variables on rows and columns.

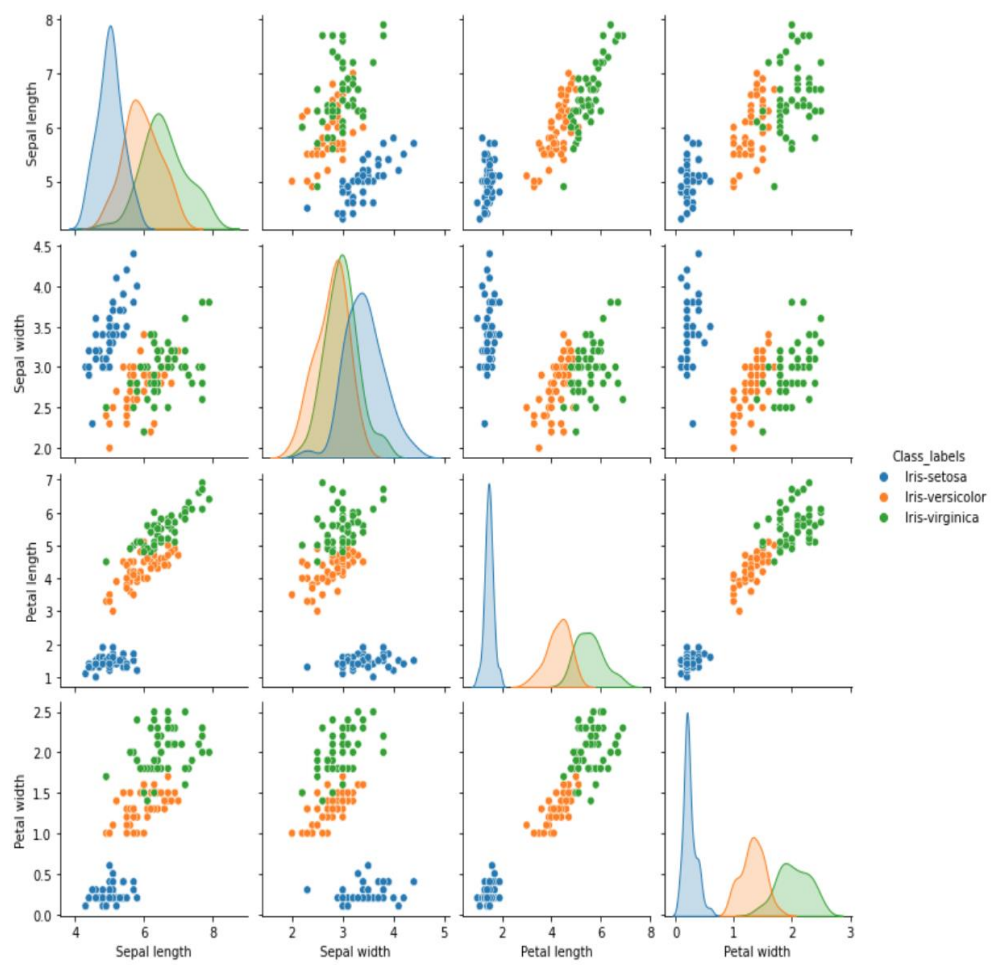


Figure 6

5. SKLEARN (SCIKIT LEARN)



It is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Here we imported the data set using pandas and explored the data using methods like `head()`, `tail()`, `dtypes()` and then try our hand at using plotting techniques from Seaborn to visualize our data.

Machine Learning:

SUPERVISED LEARNING:

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

How Supervised Learning Works?

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:

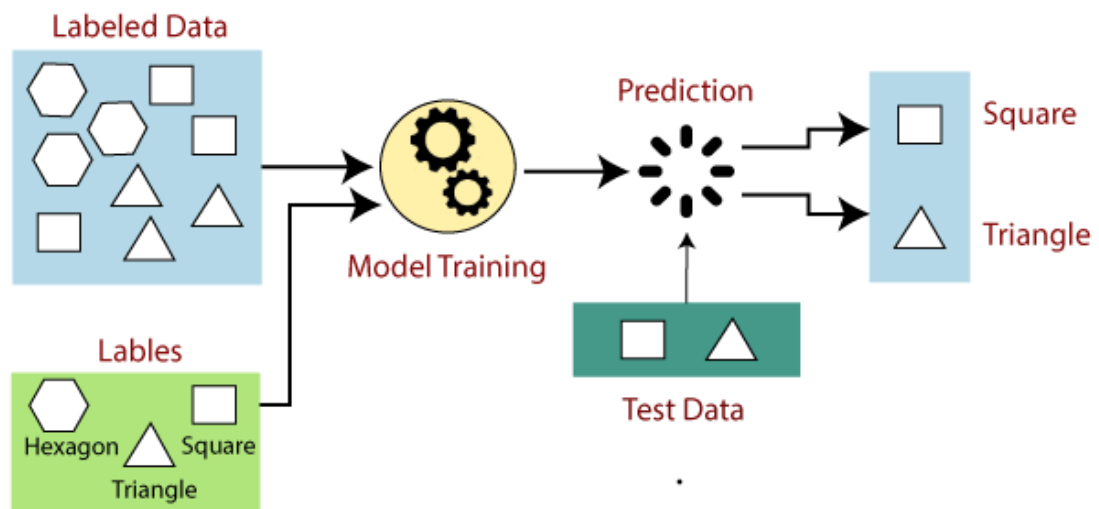


Figure 7

Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:

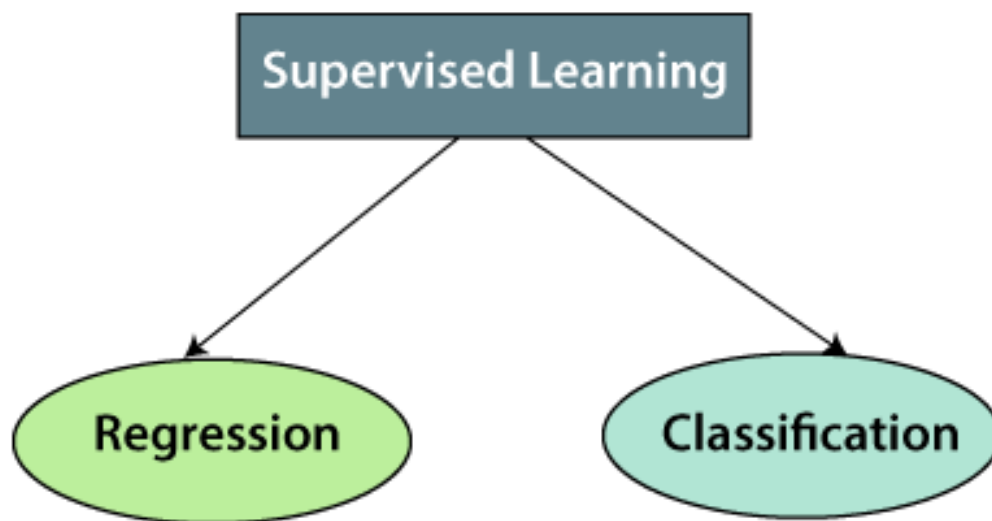


Figure 8

1. Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

2. Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Spam Filtering,
- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

UNSUPERVISED LEARNING:

Unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things.

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:

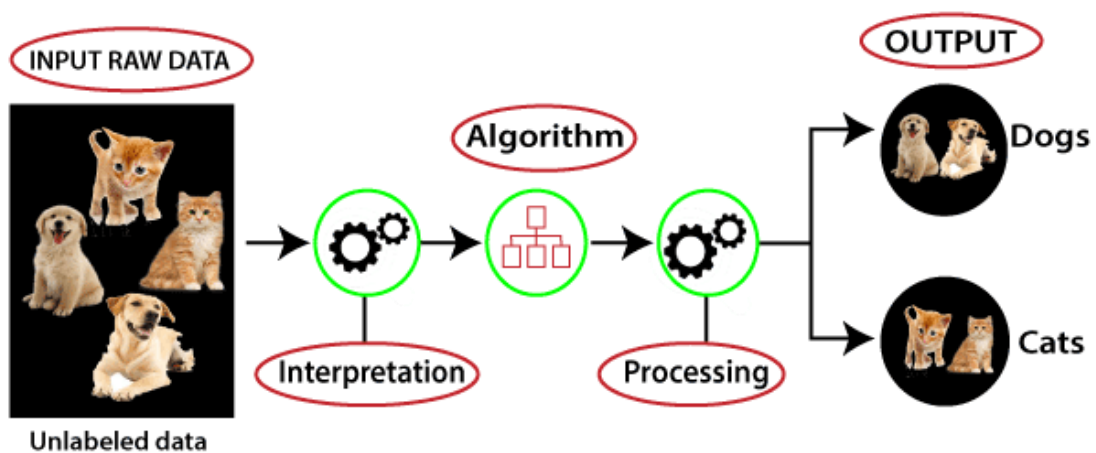


Figure 9

Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:

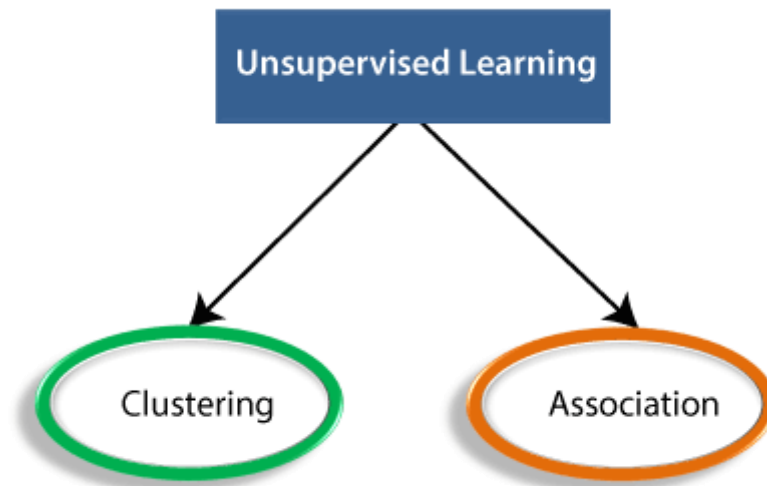


Figure 10

Clustering: Clustering is a method of grouping the objects into clusters such that objects with most similarities remain in a group and have less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

Association: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Steps involved in Machine Learning

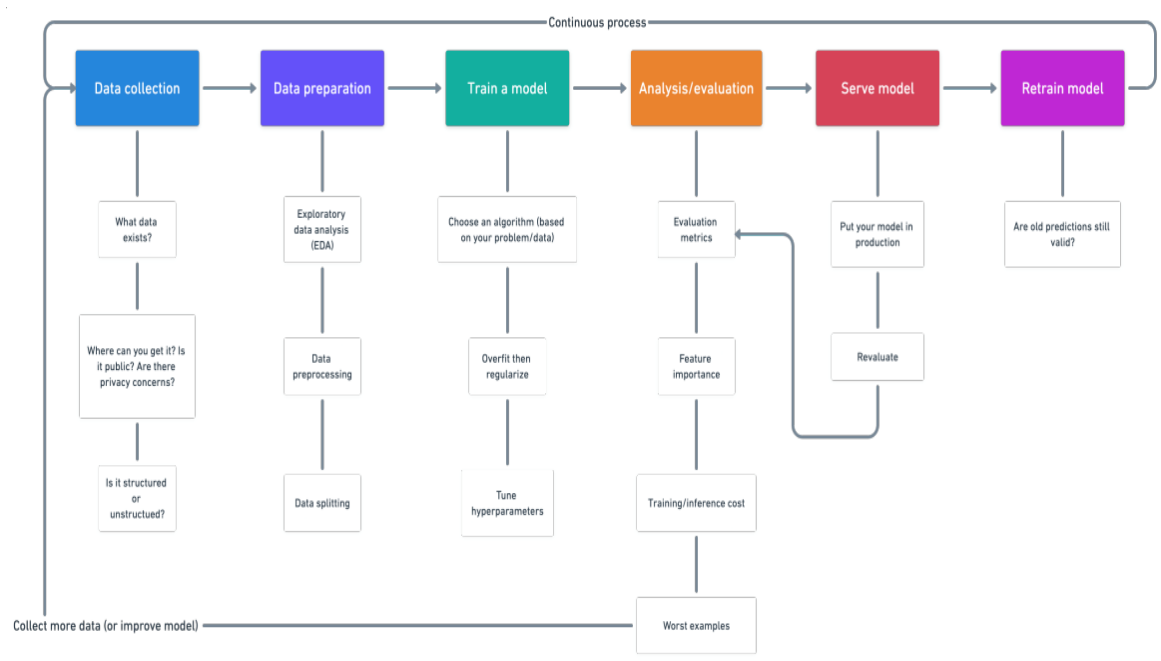


Figure 11

1. Data Collection

The real estate price prediction project that I have made is using Boston housing data for learning purpose only.

The boston dataset comes pre loaded with the pandas dataframe so we are using:

```
boston = load_boston()  
df = pd.DataFrame(boston.data)
```

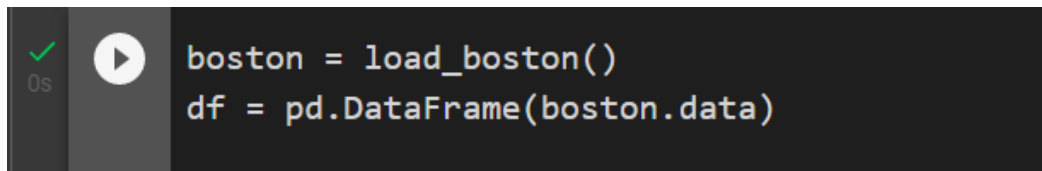


Figure 12

- 1.1 The quantity & quality of your data dictate how accurate our model is.
- 1.2 The outcome of this step is generally a representation of data which we will use for training.
- 1.3 Using pre-collected data, by way of datasets from Kaggle, GitHub, etc., still fits into this step.
- 1.4 What type of data it is- Structured or Unstructured.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978. Used in Belsley, Kuh & Welsch, 'Regression diagnostics ...', Wiley, 1980. N.B. Various transformations are used in the table on pages 244-261 of the latter.

Variables in order:

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

0.00632	18.00	2.310	0	0.5380	6.5750	65.20	4.0900	1	296.0	15.30
396.90	4.98	24.00								
0.02731	0.00	7.070	0	0.4690	6.4210	78.90	4.9671	2	242.0	17.80
396.90	9.14	21.60								
0.02729	0.00	7.070	0	0.4690	7.1850	61.10	4.9671	2	242.0	17.80
392.83	4.03	34.70								
0.03237	0.00	2.180	0	0.4580	6.9980	45.80	6.0622	3	222.0	18.70
394.63	2.94	33.40								
0.06905	0.00	2.180	0	0.4580	7.1470	54.20	6.0622	3	222.0	18.70
396.90	5.33	36.20								
0.02985	0.00	2.180	0	0.4580	6.4300	58.70	6.0622	3	222.0	18.70
394.12	5.21	28.70								
0.08829	12.50	7.870	0	0.5240	6.0120	66.60	5.5605	5	311.0	15.20
395.60	12.43	22.90								

2. Data Preparation

2.1 Wrangle data and prepare it for training.

2.2 Clean the data which may require(remove duplicates, correct errors, deal with missing values, normalization, data type conversions etc.)

2.3 Randomize data, which erases the effects of particular order in which we collected and/or otherwise prepared our data.

```
[ ] df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

Figure 13

```
✓ 0s df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
#   Column  Non-Null Count  Dtype
---  -
0    0      506 non-null    float64
1    1      506 non-null    float64
2    2      506 non-null    float64
3    3      506 non-null    float64
4    4      506 non-null    float64
5    5      506 non-null    float64
6    6      506 non-null    float64
7    7      506 non-null    float64
8    8      506 non-null    float64
9    9      506 non-null    float64
10   10     506 non-null    float64
11   11     506 non-null    float64
12   12     506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

I have used `info` and `describe` functions for preparing the data. The `info()` function prints information about the `DataFrame`. The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values) . The `describe()` function is used for calculating some statistical data like percentile, mean and std of the numerical values of the `Series` or `DataFrame`. It analyses both numeric and object series and also the `DataFrame` column sets of mixed data types.

3. Train a model

```
from sklearn.linear_model import LinearRegression
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

```
✓ [49] X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
0s
```

```
✓ [31] len(X_train)
0s
```

```
404
```

```
✓ [32] len(y_train)
0s
```

```
404
```

```
✓ [33] len(X_test)
0s
```

```
102
```

```
✓ [34] len(y_test)
0s
```

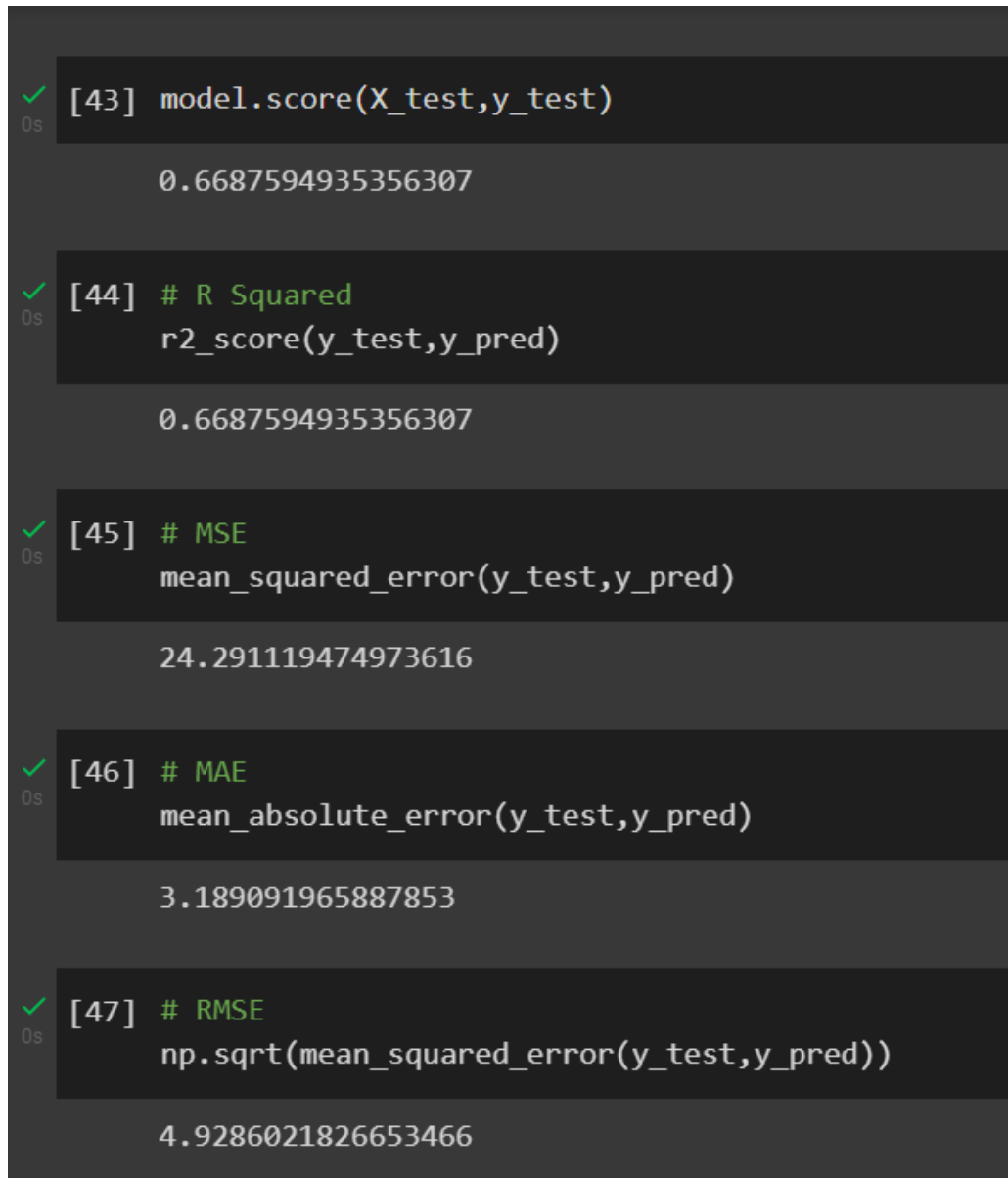
```
102
```

```
✓ [37] model.fit(X_train,y_train)
0s
```

```
LinearRegression()
```

Figure 14

4. Analyse and Evaluation



The image shows a Jupyter Notebook interface with five code cells, each containing a metric calculation and its output. Each cell is preceded by a green checkmark and '0s' indicating successful execution. The metrics calculated are model.score, r2_score, mean_squared_error, mean_absolute_error, and np.sqrt(mean_squared_error).

```
[43] model.score(X_test,y_test)
0.6687594935356307

[44] # R Squared
r2_score(y_test,y_pred)
0.6687594935356307

[45] # MSE
mean_squared_error(y_test,y_pred)
24.291119474973616

[46] # MAE
mean_absolute_error(y_test,y_pred)
3.189091965887853

[47] # RMSE
np.sqrt(mean_squared_error(y_test,y_pred))
4.9286021826653466
```

Figure 15

SYSTEM DESIGN

USE CASE DIAGRAM:

Use case diagram consists of use cases and actors and shows the interaction between them.

The key points are:

1. The main purpose is to show the interaction between the use cases and the actor.
2. To represent the system requirement from user's perspective.
3. The use cases are the functions that are to be performed in the module.

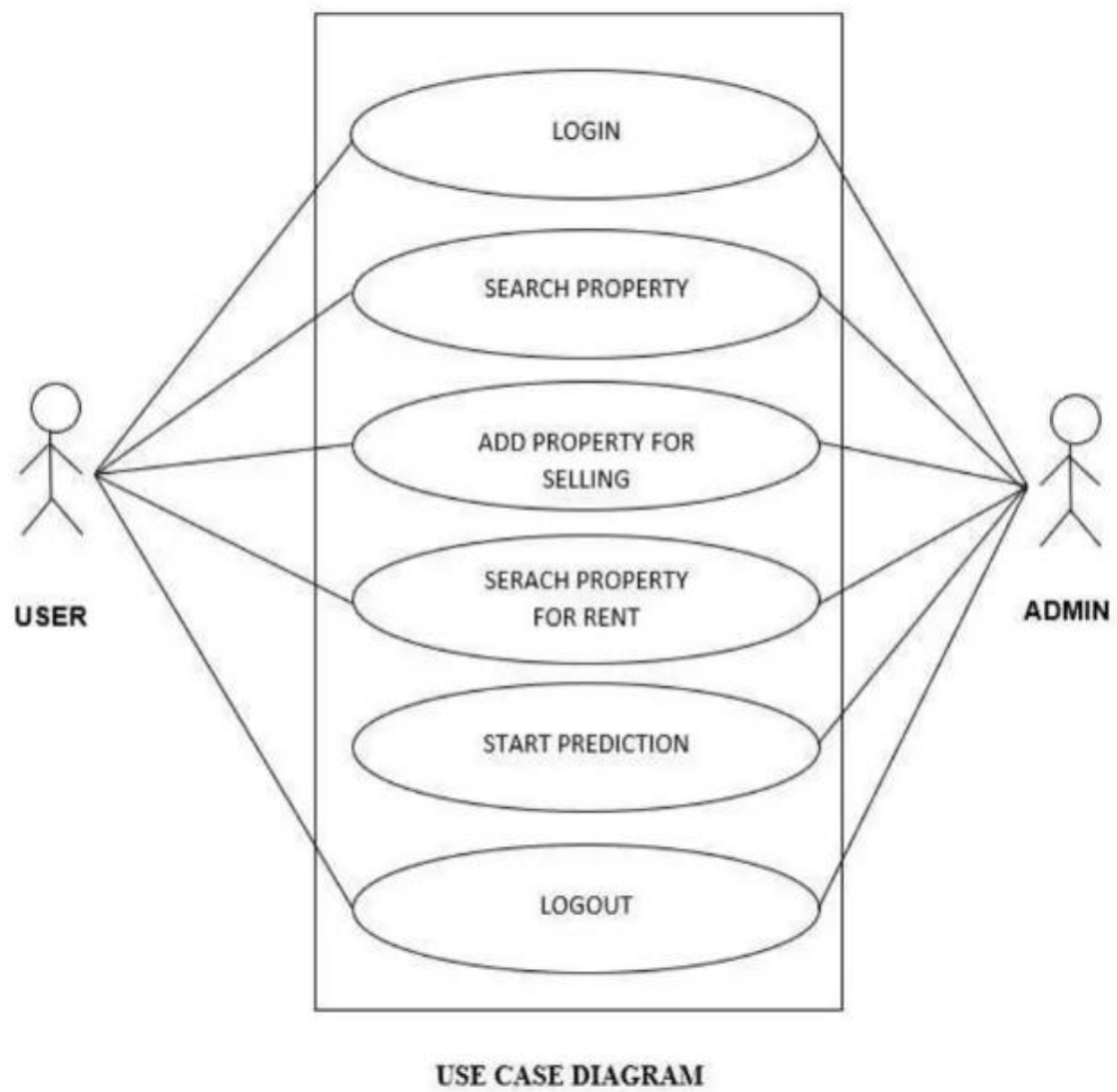


Figure 16

DATA FLOW DIAGRAM(DFD):

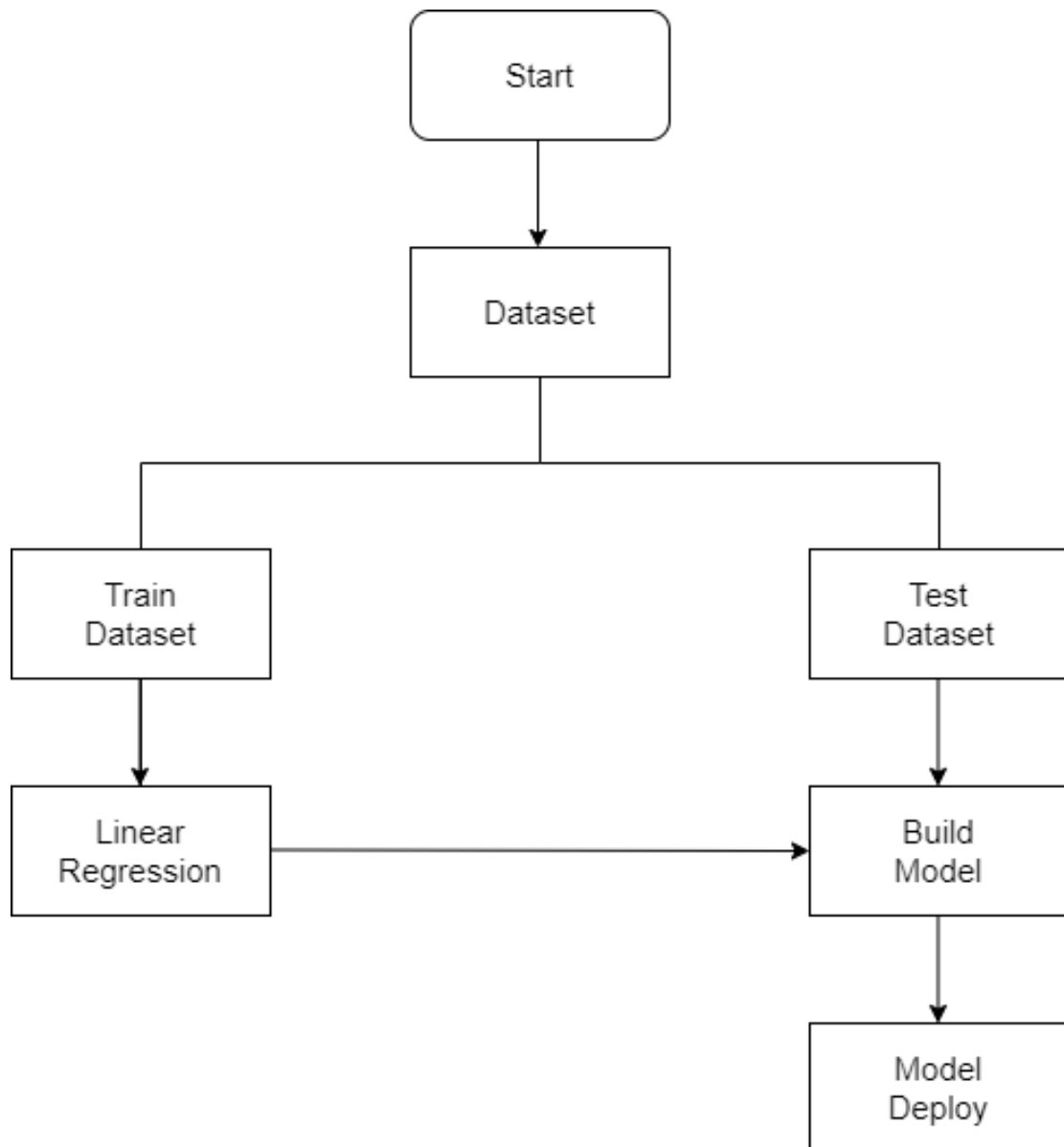


Figure 17

ER DIAGRAM:

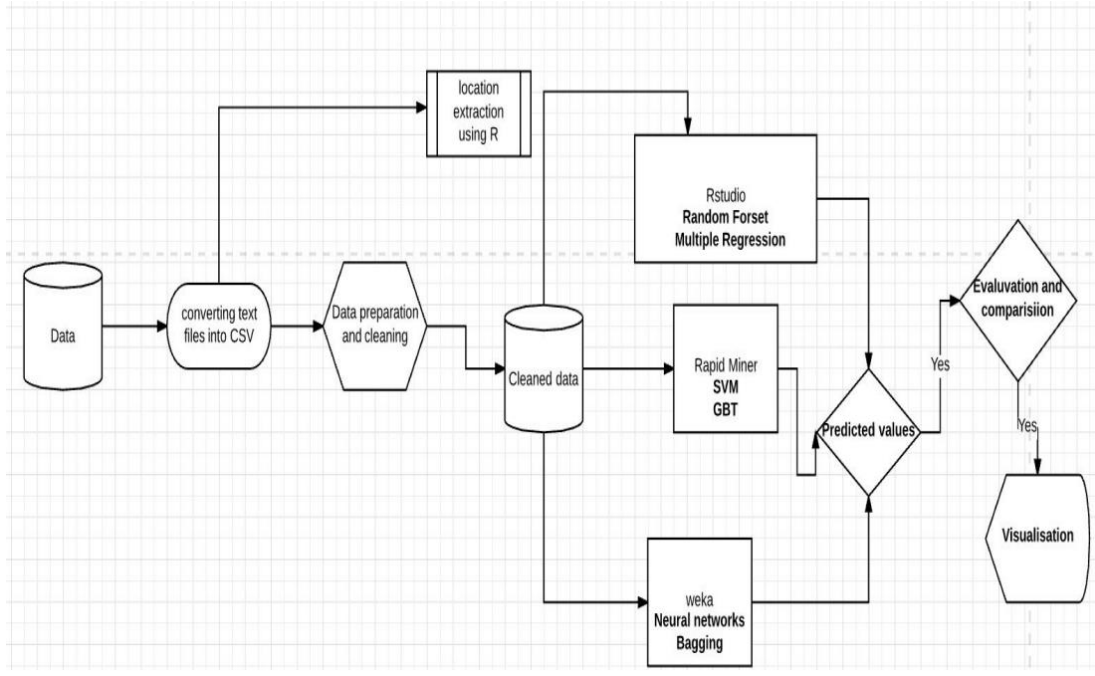


Figure 18

IMPLEMENTATION AND SYSTEM TESTING:

After all phase have been perfectly done, the system will be implemented to the server and the system can be used.

SYSTEM TESTING:

The goal of the system testing process was to determine all faults in our project .The program was subjected to a set of test inputs and many explanations were made and based on these explanations it will be decided whether the program behaves as expected or not. Our Project went through two levels of testing

1. Unit testing
2. Integration testing

UNIT TESTING:

Unit testing is commenced when a unit has been created and effectively reviewed .In order to test a single module we need to provide a complete environment i.e. besides the section we would require

- The procedures belonging to other units that the unit under test calls
- Non local data structures that module accesses
- A procedure to call the functions of the unit under test with appropriate parameters

1. Test for the updating the post

- **Testing update form** -This form is used for taking the id of the post to the backend and then finding the post with that id and updating the post with that id. And as soon as the update button is clicked the website asks for the admin id and pass and if that matches the post is updated and if it does not match the home page is displayed. And the post is updated successfully.

2. Test for deletion of the post

- **Testing delete form** -This form is used for taking the id of the post to the backend and then finding the post with that id and deleting the post with that id. And as soon as the delete button is clicked, first it checks if the keyword “delete or Delete” is written and then the website asks for the admin id and pass and if that matches, the post is deleted and if it does not match the home page is displayed. And the post is deleted successfully.

INTEGRATION TESTING:

In the Integration testing we test various combination of the project module by providing the input. The primary objective is to test the module interfaces in order to confirm that no errors are occurring when one module invokes the other module.

CONCLUSION

CONCLUSION:

In this project, we used Linear Regression and Supervised Learning to train our data. Processing of data is also important to acquire the best result and as we can see the above results, they are very satisfactory. The accuracy scores of the above model is good and can be used to predict the prices of real estate. As we can conclude that in future with appropriate data of features of any house it is possible to classify the price of the house.

Through this project, I learned how to train our own supervised machine learning model using Real Estate Price Prediction Project with Machine Learning. Through this project, we learned about machine learning, data analysis, data visualization, model creation, etc.

While, making this project I also faced certain roadblocks and obstacles, but they were eventually overcome. I created and learned what I aimed to, getting introduced to the exciting world of Machine Learning Algorithms.

FUTURE SCOPE:

The price prediction case study above shows that the Machine Learning algorithm works well in this linear regression method. The speed of computing is fast and the result is acceptable. However, it is just one of the algorithm in supervised learning. There are more algorithms for different work objectives in different scientific fields. Using these algorithms might give us better accuracy too.

Moreover, we can also add more features to our dataset like the texture of the petals, or use better machinery to get better, clearer and faster results.

REFERENCES

- 1) <https://medium.com/pursuitnotes/support-vector-regression-in-6-steps-with-python-c4569acd062d>
- 2) <https://www.youtube.com/watch?v=2v430er9hkI>
- 3) <https://numpy.org/doc/stable/>
- 4) <https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/>
- 5) https://www.w3schools.com/python/matplotlib_pie_charts.asp
- 6) <https://gist.github.com/netj/8836201>
- 7) <https://seaborn.pydata.org/generated/seaborn.PairGrid.html>
- 8) <https://www.sofcontraining.com/mission-vision.htm>
- 9) <https://muthu.co/understanding-the-classification-report-in-sklearn/#:~:text=A%20Classification%20report%20is%20used,classification%20report%20as%20shown%20below.>
- 10) <https://www.ibm.com/cloud/learn/machine-learning>
- 11) https://www.researchgate.net/profile/J-Winston/publication/327349604_A_comprehensive_review_on_iris_image-based_biometric_system/links/5be987bda6fdcc3a8dd0da3b/A-comprehensive-review-on-iris-image-based-biometric-system.pdf
- 12) <https://www.javatpoint.com/supervised-machine-learning>

APPENDIX A

SCREENSHOTS

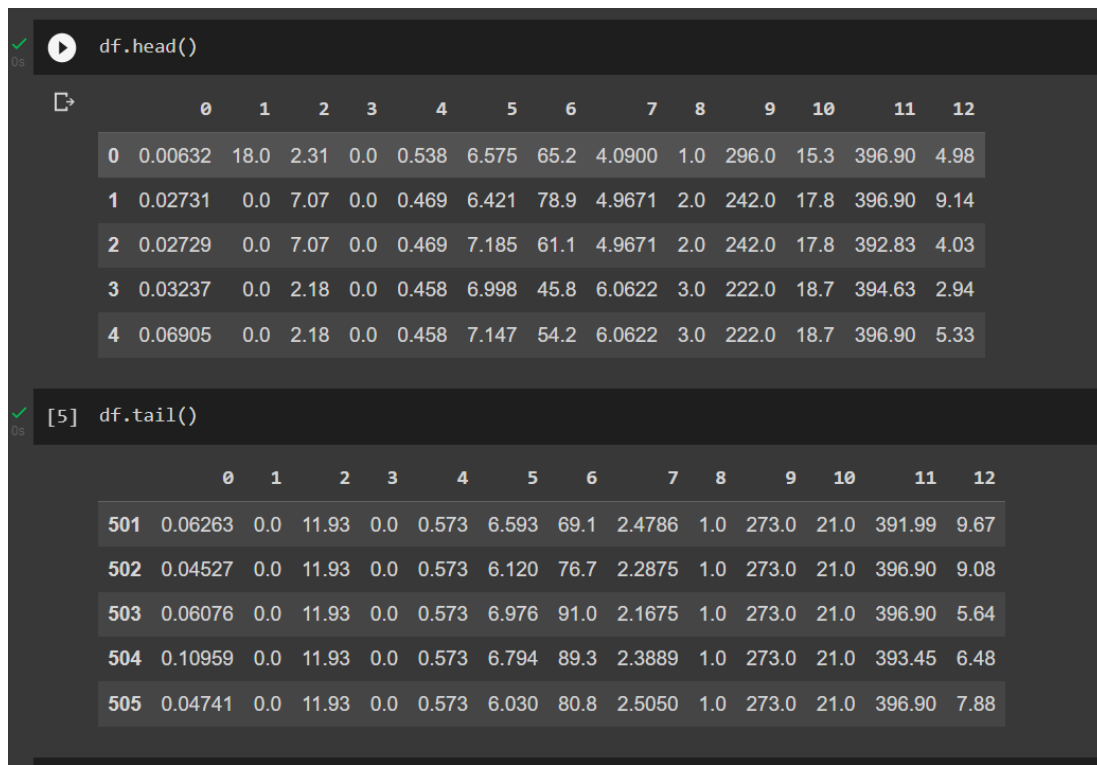


Figure 19

```
[12] df.columns

Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
      'PTRATIO', 'B', 'LSTAT'],
      dtype='object')

[6] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
#   Column  Non-Null Count  Dtype  
---  -
0    0      506 non-null     float64
1    1      506 non-null     float64
2    2      506 non-null     float64
3    3      506 non-null     float64
4    4      506 non-null     float64
5    5      506 non-null     float64
6    6      506 non-null     float64
7    7      506 non-null     float64
8    8      506 non-null     float64
9    9      506 non-null     float64
10   10     506 non-null     float64
11   11     506 non-null     float64
12   12     506 non-null     float64
dtypes: float64(13)
memory usage: 51.5 KB
```

Figure 20

```
[14] df.head()

   CRIM  ZN  INDUS  CHAS  NOX   RM  AGE  DIS  RAD  TAX  PTRATIO   B  LSTAT  Price
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0   15.3  396.90  4.98  24.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0   17.8  396.90  9.14  21.6
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0   17.8  392.83  4.03  34.7
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0   18.7  394.63  2.94  33.4
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0   18.7  396.90  5.33  36.2

[15] df.nunique()

CRIM      504
ZN         26
INDUS      76
CHAS        2
NOX        81
RM        446
AGE        356
DIS        412
RAD         9
TAX         66
PTRATIO    46
B          357
LSTAT     455
Price     229
dtype: int64
```

Figure 21

✓ [16] df.isnull()

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...
501	False	False	False	False	False	False	False	False	False	False	False	False	False	False
502	False	False	False	False	False	False	False	False	False	False	False	False	False	False
503	False	False	False	False	False	False	False	False	False	False	False	False	False	False
504	False	False	False	False	False	False	False	False	False	False	False	False	False	False
505	False	False	False	False	False	False	False	False	False	False	False	False	False	False

506 rows × 14 columns

Figure 22

✓ [17] df.isnull().sum()

CRIM	0
ZN	0
INDUS	0
CHAS	0
NOX	0
RM	0
AGE	0
DIS	0
RAD	0
TAX	0
PTRATIO	0
B	0
LSTAT	0
Price	0
dtype:	int64

Figure 23

```

[19] df.corr

<bound method DataFrame.corr of
0      0.00632  18.0  2.31  0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1      0.02731   0.0  7.07  0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2      0.02729   0.0  7.07  0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3      0.03237   0.0  2.18  0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4      0.06905   0.0  2.18  0.0  0.458  7.147  54.2  6.0622  3.0  222.0
..      ...     ...     ...     ...     ...     ...     ...     ...     ...
501     0.06263   0.0 11.93  0.0  0.573  6.593  69.1  2.4786  1.0  273.0
502     0.04527   0.0 11.93  0.0  0.573  6.120  76.7  2.2875  1.0  273.0
503     0.06076   0.0 11.93  0.0  0.573  6.976  91.0  2.1675  1.0  273.0
504     0.10959   0.0 11.93  0.0  0.573  6.794  89.3  2.3889  1.0  273.0
505     0.04741   0.0 11.93  0.0  0.573  6.030  80.8  2.5050  1.0  273.0

PTRATIO      B  LSTAT  Price
0      15.3  396.90   4.98   24.0
1      17.8  396.90   9.14   21.6
2      17.8  392.83   4.03   34.7
3      18.7  394.63   2.94   33.4
4      18.7  396.90   5.33   36.2
..      ...     ...     ...     ...
501     21.0  391.99   9.67   22.4
502     21.0  396.90   9.08   20.6
503     21.0  396.90   5.64   23.9
504     21.0  393.45   6.48   22.0
505     21.0  396.90   7.88   11.9

[506 rows x 14 columns]>

```

Figure 24

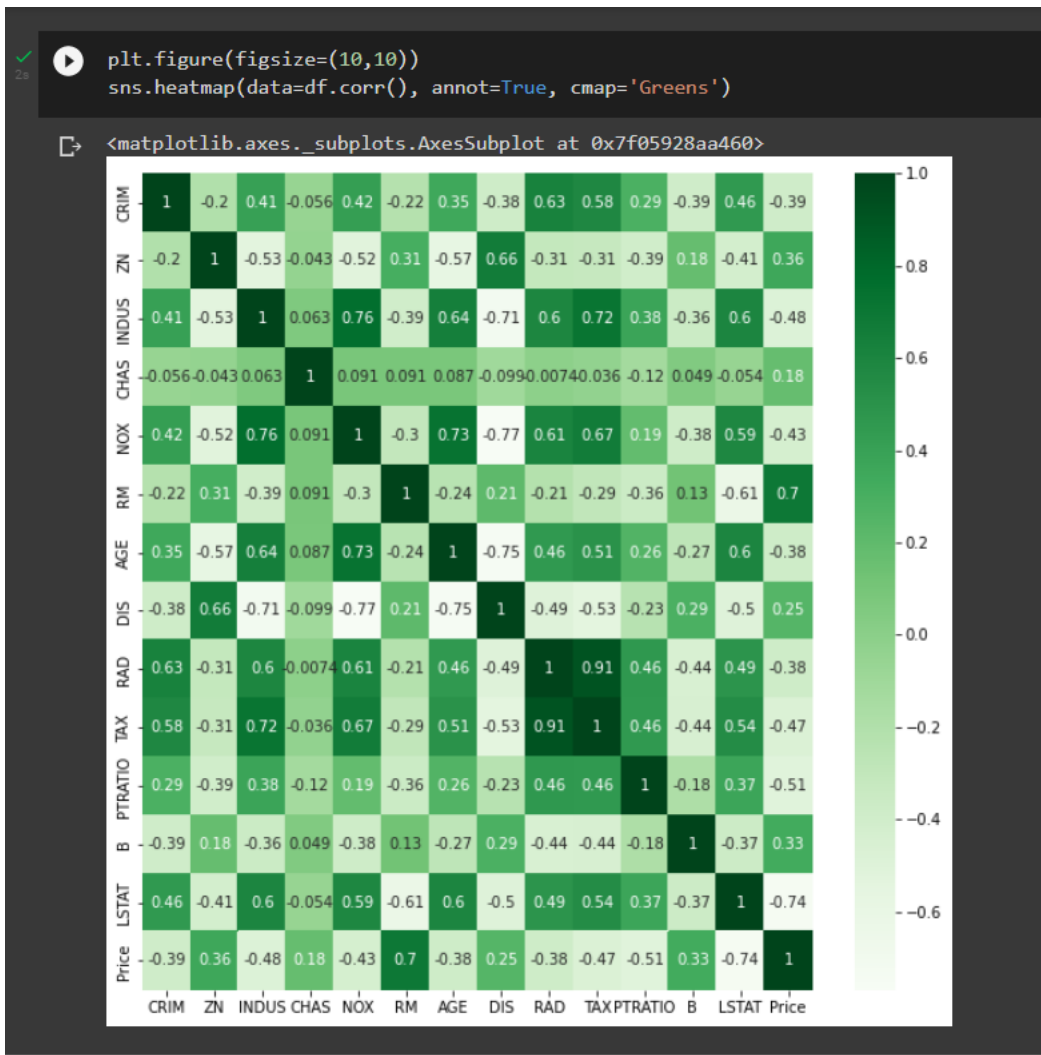


Figure 25

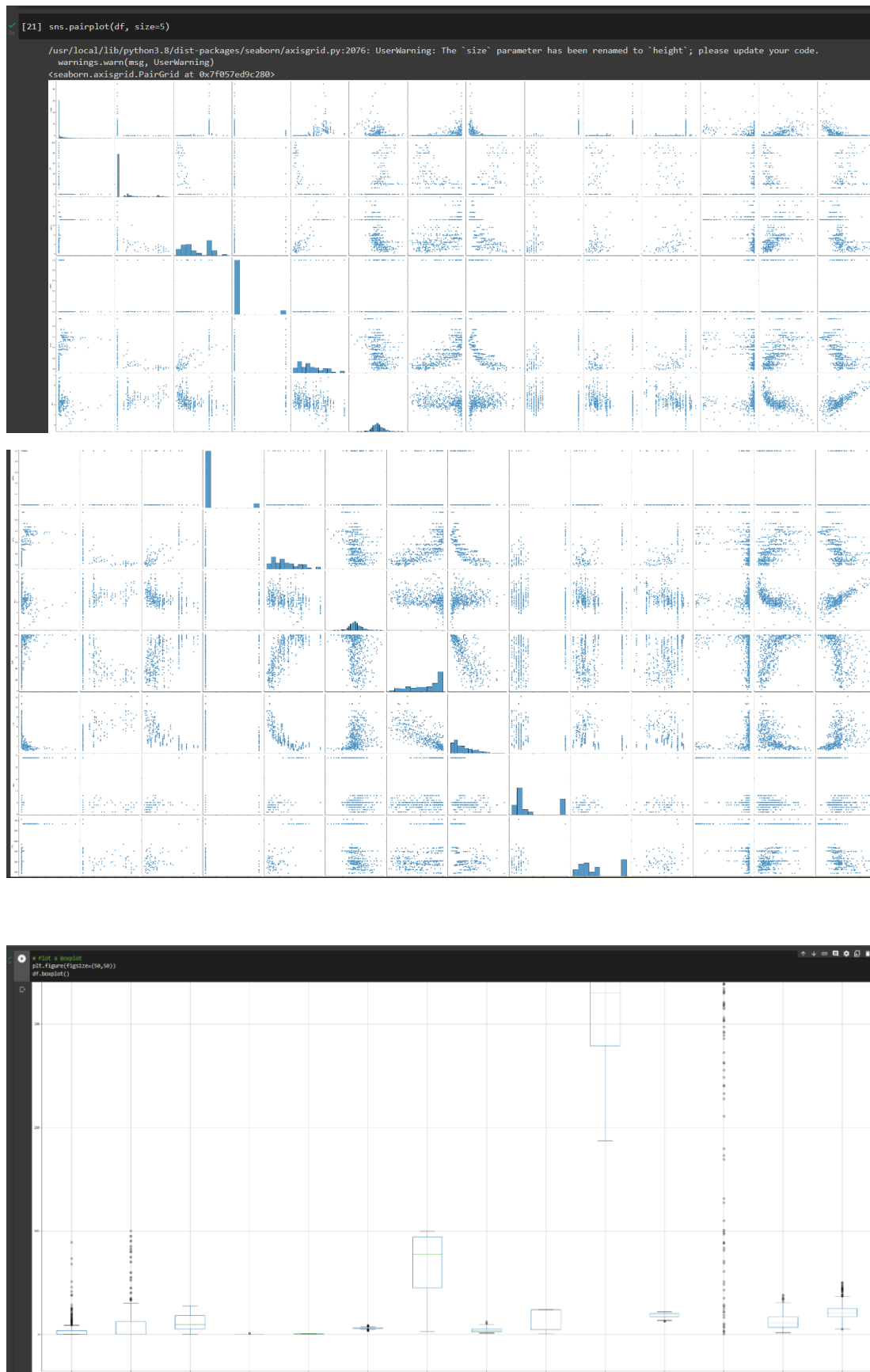


Figure 26

```

✓ [43] model.score(X_test,y_test)
0s
0.6687594935356307

✓ [44] # R Squared
0s
r2_score(y_test,y_pred)

0.6687594935356307

✓ [45] # MSE
0s
mean_squared_error(y_test,y_pred)

24.291119474973616

✓ [46] # MAE
0s
mean_absolute_error(y_test,y_pred)

3.189091965887853

✓ [47] # RMSE
0s
np.sqrt(mean_squared_error(y_test,y_pred))

4.9286021826653466

```

```

✓ [48] plt.scatter(y_test,y_pred)
0s
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.grid()
plt.plot([min(y_test),max(y_test)],[min(y_pred),max(y_pred)], color='red')
plt.title('Actual Price V/s Predicted Price')

```

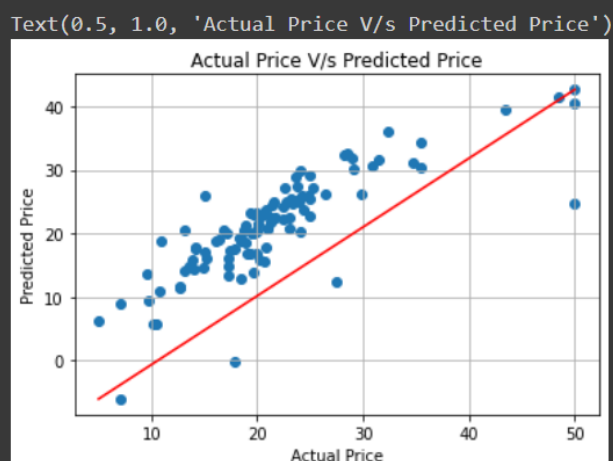


Figure 27

APPENDIX B

SOURCE CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
from sklearn.linear_model import LinearRegression
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

```
boston = load_boston()
df = pd.DataFrame(boston.data)
```

```
df.head()
df.tail()
```

```
df.shape[0]
df.shape[1]
```

```
df.columns
```

```
df.columns = boston.feature_names
```

```
df.columns
```

```
df.info()
```

```
df['Price'] = boston.target
df.head()
df.nunique()
```

```
df.isnull()
df.isnull().sum()
```

```
df.describe()
df.corr
```

```
plt.figure(figsize=(10,10))
sns.heatmap(data=df.corr(), annot=True, cmap='Greens')
```

```
# Plot a Boxplot
plt.figure(figsize=(50,50))
df.boxplot()
```

```
sns.pairplot(df, size=5)
```

```
# Minimum Price
df.Price.min()
# Maximum Price
df.Price.max()
# Standard Deviation
df.Price.std()
```

```
df.to_csv('boston_dataset.csv')
```

```
df.head()
```

```
X = np.array(df.drop('Price', axis=1))
y = np.array(df.Price)

# x = boston.data
# y = boston.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
len(X_train)
len(y_train)
len(X_test)
len(y_test)
```

```
model = LinearRegression()
model.fit(X_train,y_train)
# Intercept Value
model.intercept_
# Coefficient Value
model.coef_
```

```
y_test
y_pred = model.predict(X_test)
```


y_pred

```
model.score(X_test,y_test)
# R Squared
r2_score(y_test,y_pred)
# MSE
mean_squared_error(y_test,y_pred)
# MAE
mean_absolute_error(y_test,y_pred)
# RMSE
np.sqrt(mean_squared_error(y_test,y_pred))
plt.scatter(y_test,y_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.grid()
plt.plot([min(y_test),max(y_test)],[min(y_pred),max(y_pred)], color='red')
plt.title('Actual Price V/s Predicted Price')
```