Project report on   
FLAT PRICE Prediction System

AIML six week course at NIELIT Ropar

Simran Kaur

Contents

[Chapter 1 : 3](#_Toc171674547)

[AI Ecosystem 3](#_Toc171674548)

[1.1. Introduction 3](#_Toc171674549)

[1.2. benefits of incorporating AI ecosystems 4](#_Toc171674550)

[Chapter 2: 7](#_Toc171674551)

[Python Programming 7](#_Toc171674552)

[2.1 Introduction 7](#_Toc171674553)

[2.2 Why Python? 8](#_Toc171674554)

[2.3 Python programming with Colab 9](#_Toc171674555)

[Chapter 3 : 10](#_Toc171674556)

[Numpy 10](#_Toc171674557)

[3.1 Introduction 10](#_Toc171674558)

[Chapter 4 : 11](#_Toc171674559)

[EDA with Pandas 11](#_Toc171674560)

[4.1 Introduction 11](#_Toc171674561)

[Chapter 5: 14](#_Toc171674562)

[Machine Learning 14](#_Toc171674563)

[5.1 Introduction 14](#_Toc171674564)

[Chapter 6 16](#_Toc171674565)

[Flat Price Prediction System 16](#_Toc171674566)

[6.1 About Project 16](#_Toc171674567)

[6.2 Dataset 17](#_Toc171674568)

[https://raw.githubusercontent.com/simranbrar411/codes/main/data.csv 17](#_Toc171674569)

[6.3 Pre-Processing Data 17](#_Toc171674570)

[6.3.1 Import the packages 17](#_Toc171674571)

[6.3.2 reading the CSV file 18](#_Toc171674572)

[6.3.3 cleaning the data - Numpy/Pandas 18](#_Toc171674573)

[6.3.5 Machine learning 21](#_Toc171674574)

[6.4 Front End 22](#_Toc171674575)

[6.4.1 : HTML / CSS/ Javascript 22](#_Toc171674576)

[6.4.2 Web server Development using Flask 23](#_Toc171674577)

[6.5 Project walkthrough 24](#_Toc171674578)

[Bibliography 25](#_Toc171674579)

List of Figures

[Figure 1:AI Ecosystem 4](#_Toc171674672)

[Figure 2: AI Tools 5](#_Toc171674673)

[Figure 3 : colab 10](#_Toc171674674)

[Figure 4 numpy 11](#_Toc171674675)

[Figure 5 dataset 18](#_Toc171674676)

[Figure 6 Importing packages 18](#_Toc171674677)

[Figure 7 Reading the CSV file 19](#_Toc171674678)

[Figure 8 Data visualization using scatter graph 20](#_Toc171674679)

[Figure 9: Pie chart (student Result column) 21](#_Toc171674680)

[Figure 10 Machine Learning 22](#_Toc171674681)

[Figure 11 Front End 23](#_Toc171674682)

[Figure 12web server development 24](#_Toc171674683)

[Figure 13 Prediction Model 25](#_Toc171674684)

List of Program Code

[ProgramCode 1 : first machine learning code 6](#_Toc171328981)

# Chapter 1 :

# AI Ecosystem

## Introduction

The AI ecosystem represents the intricate web of interconnected entities, technologies, and processes that collectively drive the development, deployment, and utilization of artificial intelligence (AI) technologies across various domains. It encompasses a diverse range of stakeholders, from researchers and developers to businesses, governments, and regulatory bodies, all contributing to the advancement and responsible adoption of AI.

At its core, the AI ecosystem revolves around innovation and collaboration. It is propelled by breakthroughs in AI research, which are translated into practical applications through the efforts of technology providers and startups. These innovations are supported by robust infrastructures, including AI frameworks, data platforms, and specialized hardware, which enable the training and deployment of AI models at scale.

Ethical considerations and regulatory frameworks play a pivotal role within the ecosystem, ensuring that AI technologies are developed and used in a manner that respects privacy, addresses biases, and promotes transparency and accountability. This is complemented by ongoing efforts in AI education and talent development, which aim to cultivate a skilled workforce capable of leveraging AI's potential while mitigating its risks.

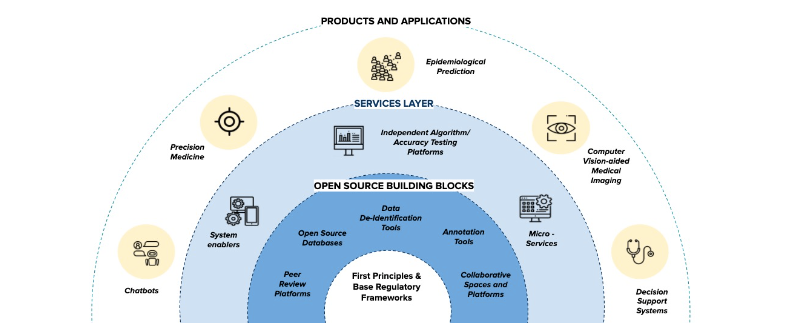


Figure 1:AI Ecosystem

Moreover, the AI ecosystem thrives on collaboration and knowledge sharing within global communities, facilitated by open-source initiatives, conferences, and partnerships

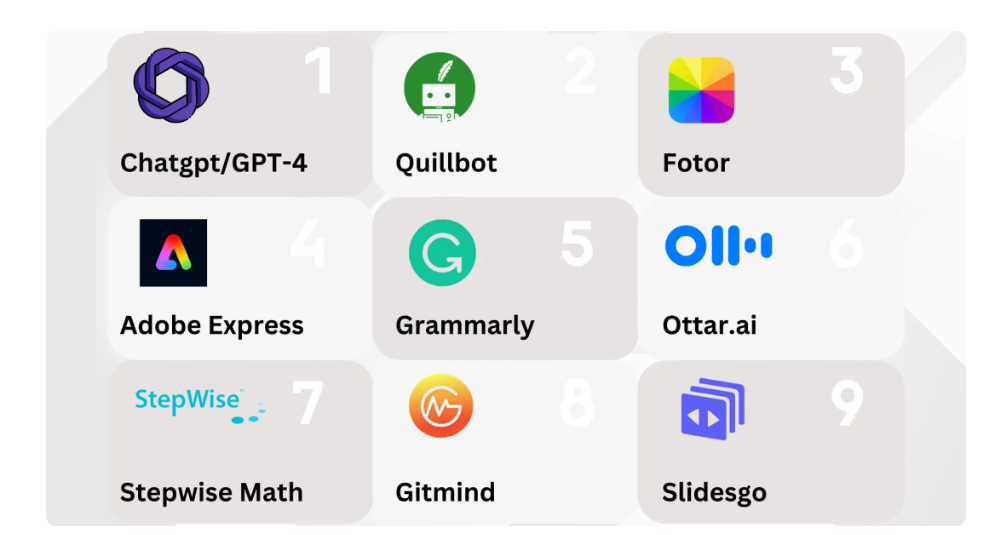
across academia, industry, and government sectors. These collaborations foster innovation, accelerate research breakthroughs, and facilitate the rapid adoption of AI across diverse applications, from healthcare and finance to transportation and entertainment.

In essence, the AI ecosystem is dynamic and multifaceted, continually evolving as technological advancements, regulatory landscapes, and societal expectations shape its trajectory. By navigating these complexities and embracing responsible practices, the AI ecosystem holds the promise of transforming industries, enhancing human capabilities, and addressing some of the world's most pressing challenges in the years to come.

## benefits of incorporating AI ecosystems

Incorporating AI ecosystems offers a wide array of benefits across various sectors and functions. Here are some key advantages:

1. Enhanced Efficiency and Automation:

 - AI systems can automate repetitive tasks, reducing human effort and error. This efficiency improvement allows businesses to streamline operations, increase productivity, and allocate human resources to more strategic and creative endeavors.

2. Improved Decision Making:

Figure 2: AI Tools

- AI algorithms can analyze vast amounts of data quickly and accurately, providing insights that aid in better decision-making processes. Whether in finance for investment decisions, healthcare for diagnosis and treatment plans, or logistics for route optimization, AI enhances decision-making capabilities by leveraging data-driven insights.

3. Personalization and Customer Experience:

- AI enables personalized experiences by analyzing customer data and behavior patterns. This personalization spans across sectors like retail (recommendation engines), healthcare (personalized medicine), and entertainment (content recommendations), enhancing customer satisfaction and loyalty.

4. Cost Savings and Resource Optimization:

- By automating processes and optimizing resource allocation, AI can lead to significant cost savings. For instance, predictive maintenance powered by AI can reduce downtime and maintenance costs in manufacturing, while smart energy grids optimize energy distribution and consumption.

5. Innovation and New Opportunities:

- AI fosters innovation by enabling the development of new products, services, and business models. Startups and enterprises alike leverage AI ecosystems to explore novel applications in areas such as autonomous vehicles, robotics, augmented reality, and more, opening up new revenue streams and market opportunities.

6. Enhanced Healthcare Outcomes:

- AI-driven medical diagnostics, personalized treatment plans, drug discovery, and patient monitoring systems improve healthcare outcomes. AI can analyze medical images, genomic data, and patient records to assist healthcare professionals in making more accurate diagnoses and providing tailored treatments.

7. Safety and Security:

- AI enhances security through applications like fraud detection in financial transactions, cybersecurity threat detection, and surveillance systems. AI-powered technologies can analyze patterns and anomalies in real-time data streams to identify and mitigate potential risks.

8. Environmental Sustainability:

- AI contributes to sustainability efforts through applications such as precision agriculture (optimizing water and fertilizer use), smart energy management (predicting demand and optimizing distribution), and monitoring environmental changes (wildlife conservation and climate modeling).

9. Improved Accessibility and Inclusivity:

- AI technologies can improve accessibility for people with disabilities by providing tools like speech recognition, text-to-speech, and computer vision systems. These innovations promote inclusivity and empower individuals to participate more fully in society and the workforce.

10. Global Collaboration and Knowledge Sharing:

- AI ecosystems facilitate global collaboration among researchers, developers, and organizations, accelerating innovation and knowledge sharing. Open-source AI frameworks, shared datasets, and collaborative projects enable advancements that benefit the global community.

In summary, incorporating AI ecosystems unlocks numerous benefits across sectors, transforming industries, enhancing decision-making capabilities, improving efficiency, fostering innovation, and ultimately contributing to societal progress and well-being.

# Chapter 2:

# Python Programming

## Introduction

Python programming is a versatile and powerful language known for its simplicity and readability, making it a popular choice for beginners and experienced developers alike. Originally created by Guido van Rossum and first released in 1991, Python has since grown into one of the most widely used programming languages worldwide.

**Key Features of Python:**

1. **Ease of Learning**: Python's syntax is straightforward and easy to understand, making it accessible for newcomers to programming. Its readability encourages developers to write clean and maintainable code.
2. **Interpreted and Interactive**: Python is an interpreted language, meaning code can be executed line by line, making it ideal for scripting and rapid prototyping. It also supports interactive mode, where commands can be executed directly from the interpreter.
3. **Cross-platform Compatibility**: Python runs on various platforms including Windows, macOS, and Linux, allowing developers to write code that can be executed on different operating systems without modification.
4. **Rich Standard Library**: Python comes with a comprehensive standard library that provides modules and packages for tasks ranging from file I/O and networking to web development and data manipulation. This reduces the need for developers to write code from scratch for common functionalities.
5. **Support for Multiple Paradigms**: Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming styles. This flexibility allows developers to choose the approach that best suits their project requirements.
6. **Community and Ecosystem**: Python has a vibrant community of developers who contribute to open-source libraries and frameworks. Popular libraries such as NumPy (for numerical computing), pandas (for data analysis), Django (for web development), and TensorFlow (for machine learning) extend Python's capabilities significantly.

**Applications of Python:**

* **Web Development**: Frameworks like Django and Flask simplify web application development with Python.
* **Data Science and Machine Learning**: Python's libraries (NumPy, pandas, scikit-learn, TensorFlow, PyTorch) are widely used for data analysis, machine learning, and artificial intelligence applications.
* **Scripting and Automation**: Python's readability and ease of use make it ideal for scripting tasks and automating repetitive processes.
* **Scientific Computing**: Python is used extensively in scientific computing and research, thanks to libraries like SciPy and matplotlib for numerical computations and data visualization.
* **Game Development**: Python is used in game development for scripting and creating game engines.
* **Desktop GUI Applications**: Python's tkinter library and frameworks like PyQt and wxPython enable developers to create graphical user interface (GUI) applications.

## Why Python?

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-oriented way or a functional way.

Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

## Python programming with Colab

Colab, or "Colaboratory", allows you to write and run Python code directly in your web browser using Google's cloud infrastructure. It's great for learning Python, experimenting with data analysis, and building machine learning models without needing to install anything on your computer.

* Zero configuration required
* Access to GPUs free of charge
* Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier.

My first Colab code file :

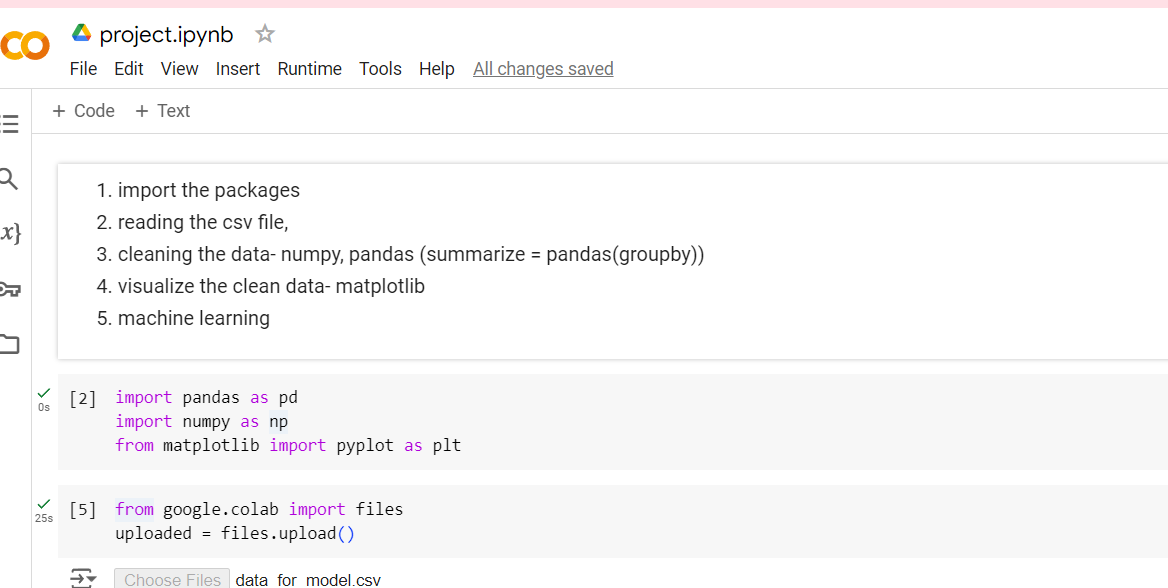


Figure 3 : colab

ProgramCode 1 : first machine learning code

<https://colab.research.google.com/drive/1cRsaoLPDginw9R2MlNSVMoLXkCJDZhmE?usp=sharing>

# Chapter 3 :

# Numpy

## 3.1 Introduction

NumPy, short for Numerical Python, is a fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is essential for tasks involving numerical data, making it a cornerstone library in the Python ecosystem for scientific computing.

**Key Features of NumPy:**

1. **Multi-dimensional Arrays**: NumPy arrays (ndarrays) are homogeneous, which means they contain elements of the same data type. These arrays can be one-dimensional, two-dimensional, or multi-dimensional, allowing for efficient storage and manipulation of data.
2. **Efficient Operations**: NumPy provides efficient operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, and more.
3. **Broadcasting**: NumPy's broadcasting capability allows for arithmetic operations on arrays of different shapes. It extends the arrays to perform element-wise operations without explicitly creating multiple copies of the data.
4. **Integration with Other Libraries**: NumPy is a foundational library in the Python scientific computing stack and integrates seamlessly with other libraries like SciPy (for advanced mathematics), Matplotlib (for plotting), pandas (for data analysis), and scikit-learn (for machine learning).

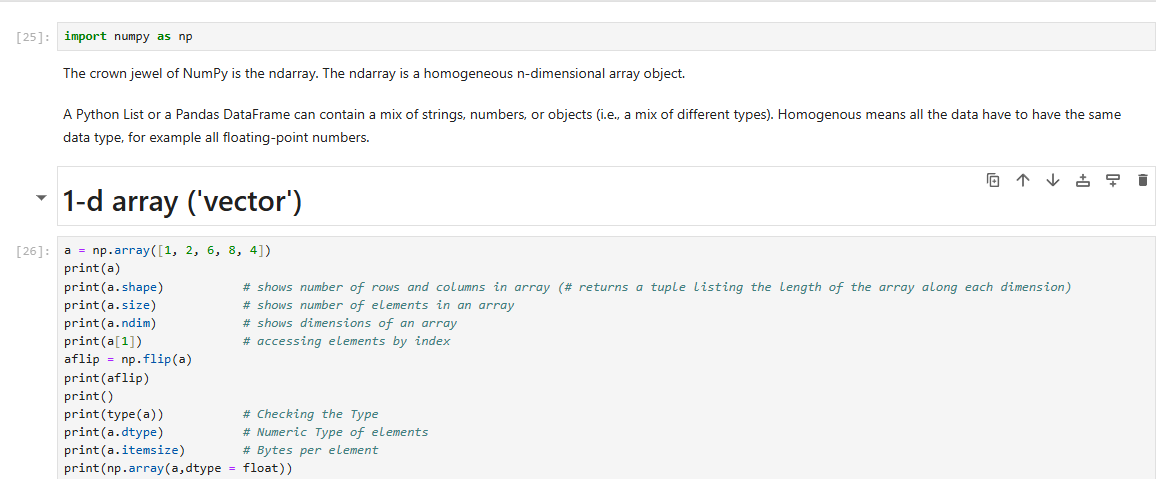


Figure 4 numpy

# Chapter 4 :

# EDA with Pandas

## 4.1 Introduction

The pandas library offers numerous intuitive methods that can be successfully applied for exploratory data analysis. EDA is an important step in any data science project. It helps grasp the size and structure of the data, identify the critical variables and potential issues with them, explore the statistics of the data, find the correlation between different columns, and discover hidden patterns in the data for further investigation. EDA in pandas allows running a few lines of simple code to efficiently solve all these tasks and more.

**Introduction to Exploratory Data Analysis**

Exploratory data analysis (EDA) is a vital initial step of any data analysis or machine learning project. It is necessary for:

* getting an overall understanding of the data, including first insights
* identifying the size of the dataset, its structure, and the features that are crucial for the project goal
* gathering fundamental statistics of the data
* detecting potential issues to fix (such as missing values, duplicates, or outliers)

The pandas library is the most popular Python package for conducting data wrangling and data analysis. It offers a vast spectrum of analytical tools and methods to explore the data from all angles. Performing EDA in pandas is an easy task because of the intuitive names and syntaxes of its functions, their flexibility, and comprehensive library documentation.

**EDA Functions in Pandas**

Let us take a look at the essential methods that allow for conducting fast and efficient EDA in pandas. Note that some of those functions are applicable only for data frames, others – only for series, and others – for both. In the context of EDA in pandas, we usually talk about a series object as an individual column of a dataframe.

**head() and tail()**

By default, the head() method returns the first five and tail() – the last five rows of a dataframe or a series. To return the number of rows different from five, we need to pass in that number.

**shape**

For a dataframe, it returns a tuple with the number of rows and columns. For a series, it returns a one-element tuple with the number of rows.

**size**

Returns the number of elements in a dataframe or a series. For a series object, it makes more sense to use size rather than shape. The obtained information is the same in both cases, but size returns it in a more handy form – as an integer rather than a one-element tuple.

**info()**

Returns overall information about a dataframe, including the index data type, the number of rows and columns, column names, indices, and data types, the number of non-null values by column, and memory usage.

**describe()**

Returns the major statistics of a dataframe or a series, including the number of non-null values, the minimum, maximum, and mean values, and percentiles. For a dataframe, it returns the information by column, and by default, only for numeric columns. To include the statistics for the columns of an object type as well, we need to pass in include='all'. For object columns, the method returns the number of non-null values, the number of unique values, the most frequent value, and the number of times it is encountered in the corresponding column.

**columns**

Returns the column names of a dataframe.

**count()**

Returns the count of non-null values in a dataframe or a series. For a dataframe, by default, returns the results by column. Passing in axis=1 or axis='columns' will give the results by row.

**unique()**

The unique() method returns the unique values of a series, while nunique() – the number of unique values in a dataframe or a series.

For a dataframe, nunique(), by default, returns the results by column. Otherwise, passing in axis=1 or axis='columns' will give the results by row.

**is\_unique**

Returns True if all the values in a series are unique.

**isnull()**

Both methods return a boolean same-sized object showing which values are null (True) and which are not (False). These functions apply to both series and dataframes.

isnull() and isna() work best when chained with sum() (e.g., df.isnull().sum()) returning the number of null values by column for a dataframe or their total number for a series. For a dataframe, the method chaining df.isnull().sum().sum() gives the total number of null values.

**value\_counts()**

Returns the count of each unique value in a series. By default, the outputs are not normalized, or sorted in descending order, and the null values are not considered. To override the defaults, we can set the optional parameters normalize, ascending, and dropna accordingly.

**nsmallest() and nlargest()**

By default, nsmallest() returns the five smallest while nlargest() – the five largest values of a series together with their indices. To return the number of values different from five, we need to pass in that number.

**plot()**

Allows the creation of simple plots of various kinds for a data frame or a series. The main parameters are x, y, and kind. The popular types of supported plots are line, bar, barh, hist, box, area, density, pie, and scatter.

In general, pandas are not the best choice for creating compelling visualizations in Python. However, for the purposes of EDA in pandas, the plot() method works just fine.

# Chapter 5:

# Machine Learning

## 5.1 Introduction

Machine learning (ML) is a branch of artificial intelligence (AI) that involves developing algorithms and statistical models that enable computers to learn from and make decisions or predictions based on data, without explicit programming instructions. It focuses on the development of systems that can automatically learn and improve from experience.

**Key Concepts in Machine Learning:**

1. **Learning from Data**: Machine learning algorithms are designed to analyze and learn patterns from large amounts of data. The quality and quantity of data play a crucial role in the performance and accuracy of the models.
2. **Types of Learning**:
   * **Supervised Learning**: Algorithms learn from labeled data, where the input-output pairs are explicitly provided (e.g., classification, regression).
   * **Unsupervised Learning**: Algorithms find patterns and structure in unlabeled data (e.g., clustering, dimensionality reduction).
   * **Reinforcement Learning**: Algorithms learn through trial and error, by interacting with an environment and receiving feedback in the form of rewards or penalties.
3. **Model Training and Evaluation**: In supervised learning, models are trained on a training dataset to learn patterns. They are then evaluated on a separate test dataset to assess their performance and generalization ability.
4. **Feature Engineering**: The process of selecting, extracting, and transforming features (variables) from data to improve model performance and accuracy.
5. **Model Selection and Tuning**: Choosing the appropriate machine learning algorithm and optimizing model parameters to achieve the best performance on unseen data.
6. **Deployment and Monitoring**: Implementing trained models into production environments and continuously monitoring their performance and behavior.

**Applications of Machine Learning:**

* **Natural Language Processing (NLP)**: Language translation, sentiment analysis, chatbots.
* **Computer Vision**: Object detection, image classification, facial recognition.
* **Recommendation Systems**: Personalized recommendations in e-commerce and content platforms.
* **Healthcare**: Disease prediction, medical image analysis, drug discovery.
* **Finance**: Fraud detection, risk assessment, algorithmic trading.
* **Automotive**: Autonomous driving, predictive maintenance.

**Tools and Libraries:**

* **Python**: Popular programming language for machine learning with libraries like scikit-learn, TensorFlow, PyTorch, and pandas.
* **R**: Statistical programming language with libraries like caret, ggplot2, and dplyr.
* **Machine Learning Platforms**: Cloud-based platforms such as Google AI Platform, Amazon SageMaker, and Microsoft Azure Machine Learning for scalable machine learning workflows.

**Ethical Considerations:**

* **Bias and Fairness**: Ensuring that models are fair and unbiased across different demographic groups.
* **Privacy and Security**: Safeguarding sensitive data used for training and deployment of models.
* **Transparency and Accountability**: Understanding and explaining how decisions are made by machine learning models.

Machine learning continues to revolutionize industries and everyday life, driving innovations and advancements in various domains. It represents a powerful toolset for solving complex problems and making data-driven decisions based on patterns and insights extracted from data.

Top of Form

Bottom of Form

# Chapter 6

# Flat Price Prediction System

## 6.1 About Project

A flat price prediction system using Python, data science, and machine learning aims to predict the price of flats (apartments) based on various features or attributes. This system leverages data science techniques and machine learning algorithms to analyse historical data and make predictions about future prices. Here's a brief introduction to such a system:

**Introduction to Flat Price Prediction System:**

In today's real estate market, predicting property prices accurately is crucial for buyers, sellers, and investors alike. A flat price prediction system uses data science methodologies and machine learning algorithms to process historical data related to real estate transactions and property attributes. By analysing this data, the system can forecast the price of flats based on factors such as location, size, amenities, neighbourhood characteristics, and market trends.

**Key Components and Process:**

1. **Data Collection and Preprocessing**
2. **Exploratory Data Analysis (EDA**)
3. **Model Selection and Training**
4. **Model Evaluation and Validation**
5. **Prediction and Deployment**

**Benefits and Applications:**

* **Informed Decision Making**: Buyers can use predicted prices to make informed decisions about purchasing flats.
* **Real Estate Investment**: Investors can forecast potential returns and make strategic investment decisions.
* **Market Analysis**: Real estate professionals can analyse market trends and price fluctuations.

**Conclusion:**

A flat price prediction system powered by data science and machine learning empowers stakeholders in the real estate industry with actionable insights derived from data analysis and predictive model. By leveraging advanced algorithms and techniques, this system enhances decision-making processes and helps navigate the complexities of the real estate market effectively.

## 6.2 Dataset

## <https://raw.githubusercontent.com/simranbrar411/codes/main/data.csv>

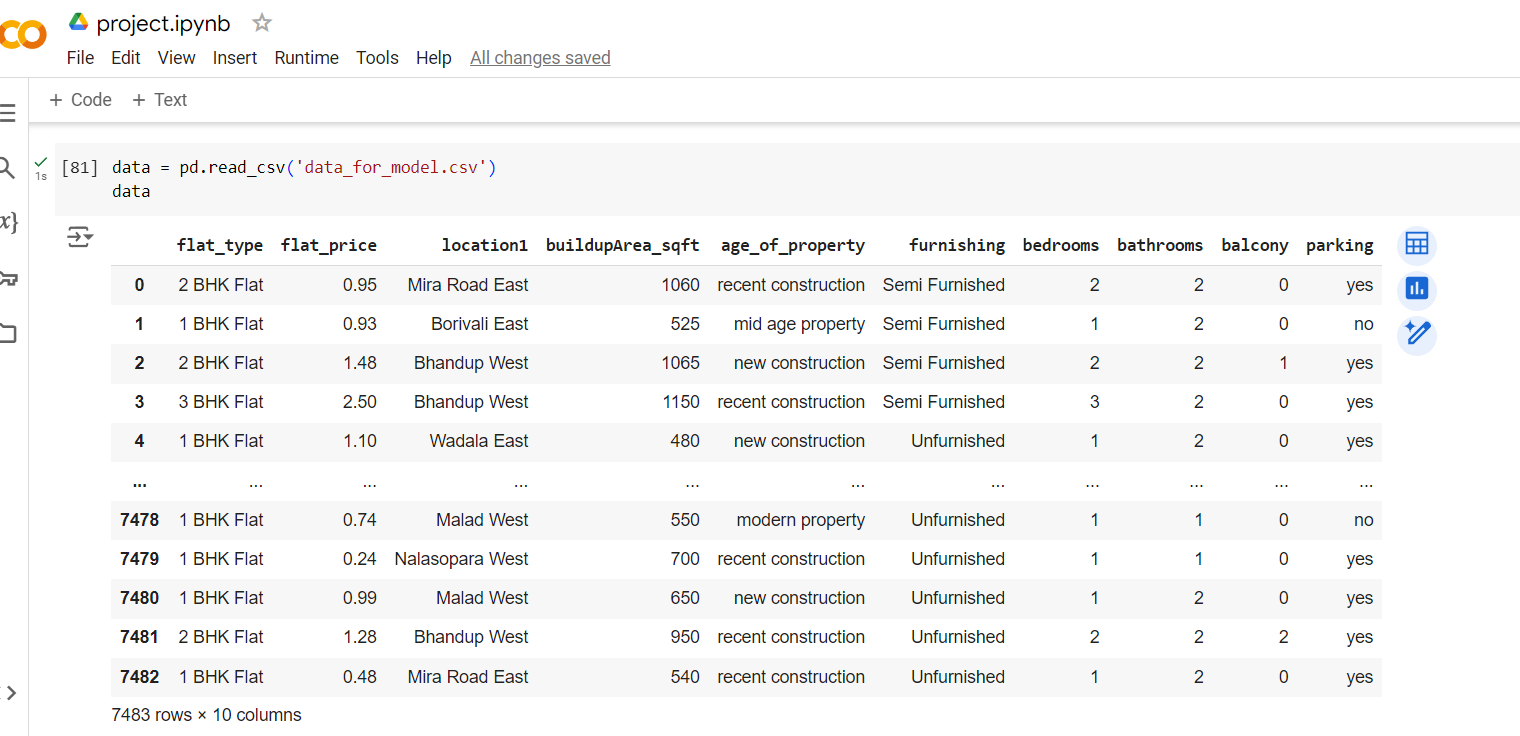


Figure 5 dataset

## 6.3 Pre-Processing Data

### 6.3.1 Import the packages

the basic steps to import commonly used packages in Python, particularly in the context of data analysis and machine learning:

**Importing Packages in Python:**

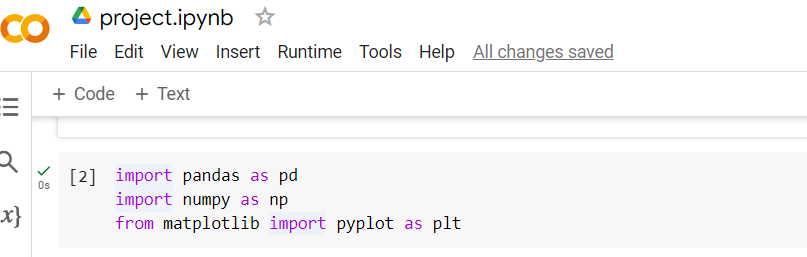


Figure 6 Importing packages

### 6.3.2 reading the CSV file

Reading a CSV (Comma-Separated Values) file is a common task in data analysis and machine learning workflows using Python.

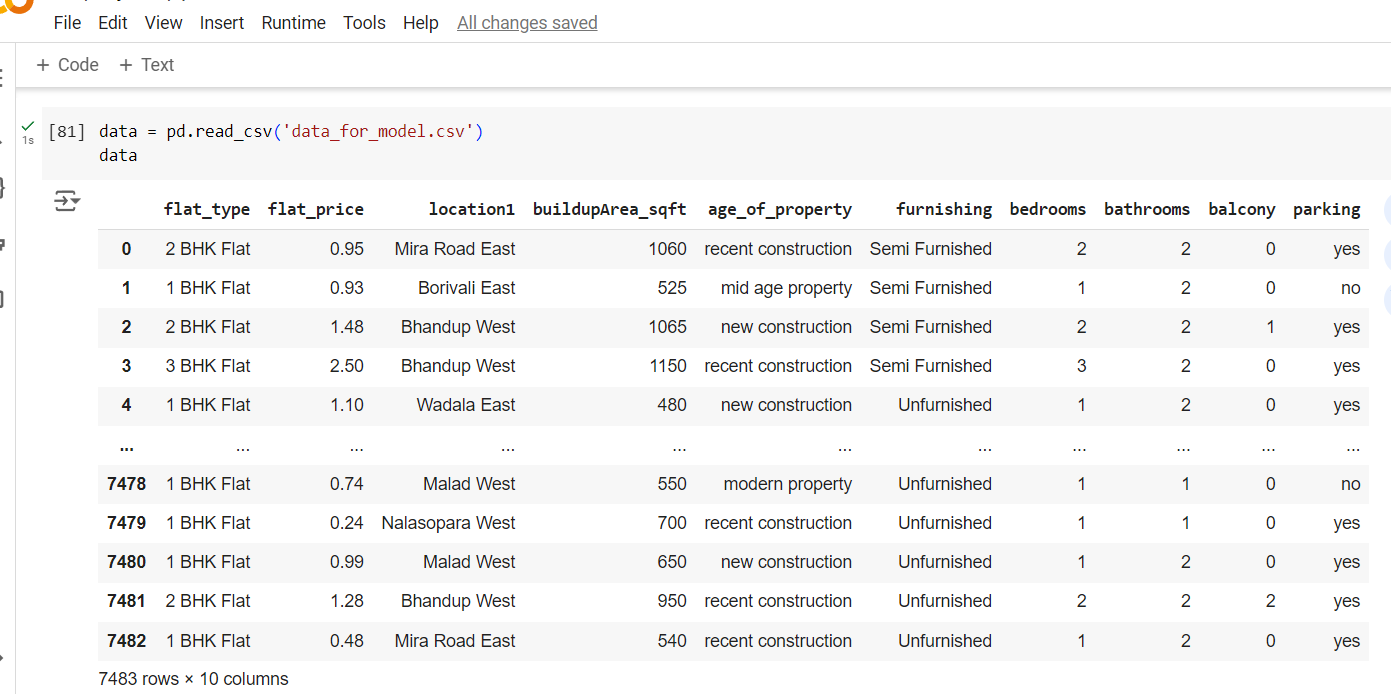


Figure 7 Reading the CSV file

### 6.3.3 cleaning the data - Numpy/Pandas

Cleaning data is a crucial step in data analysis and machine learning projects to ensure accuracy and reliability of results. Both NumPy and pandas provide powerful tools for data cleaning.

6.3.4 Visualize the clean data - matplotlib...

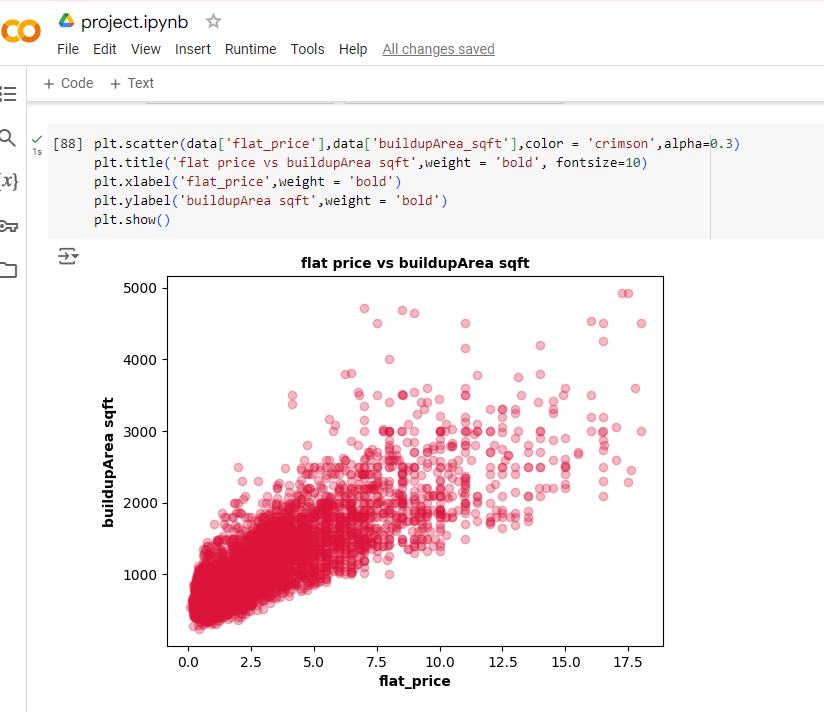
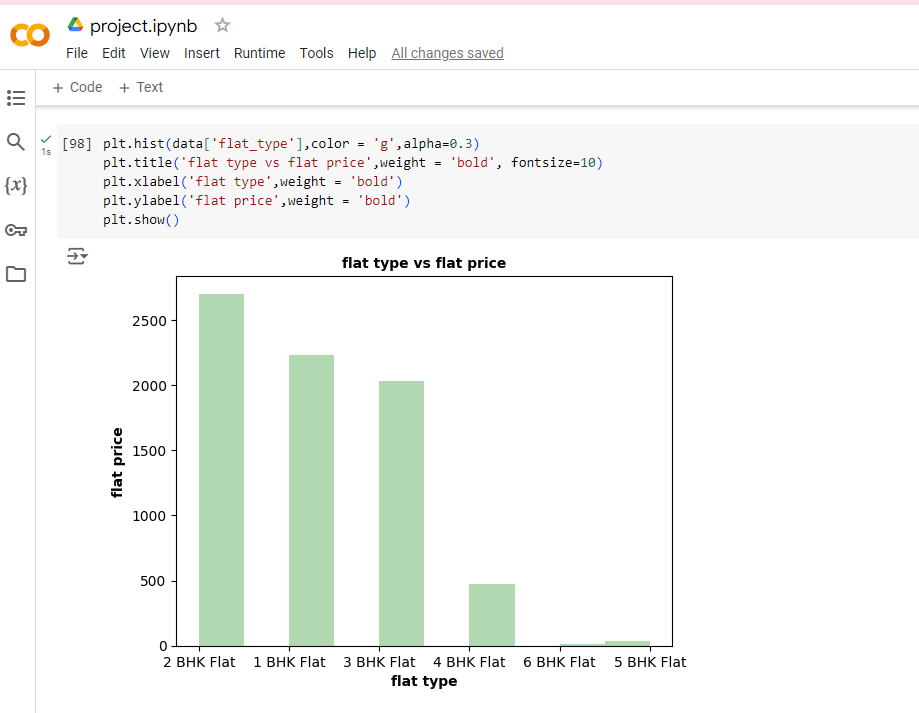


Figure 8 Data visualization using scatter graph



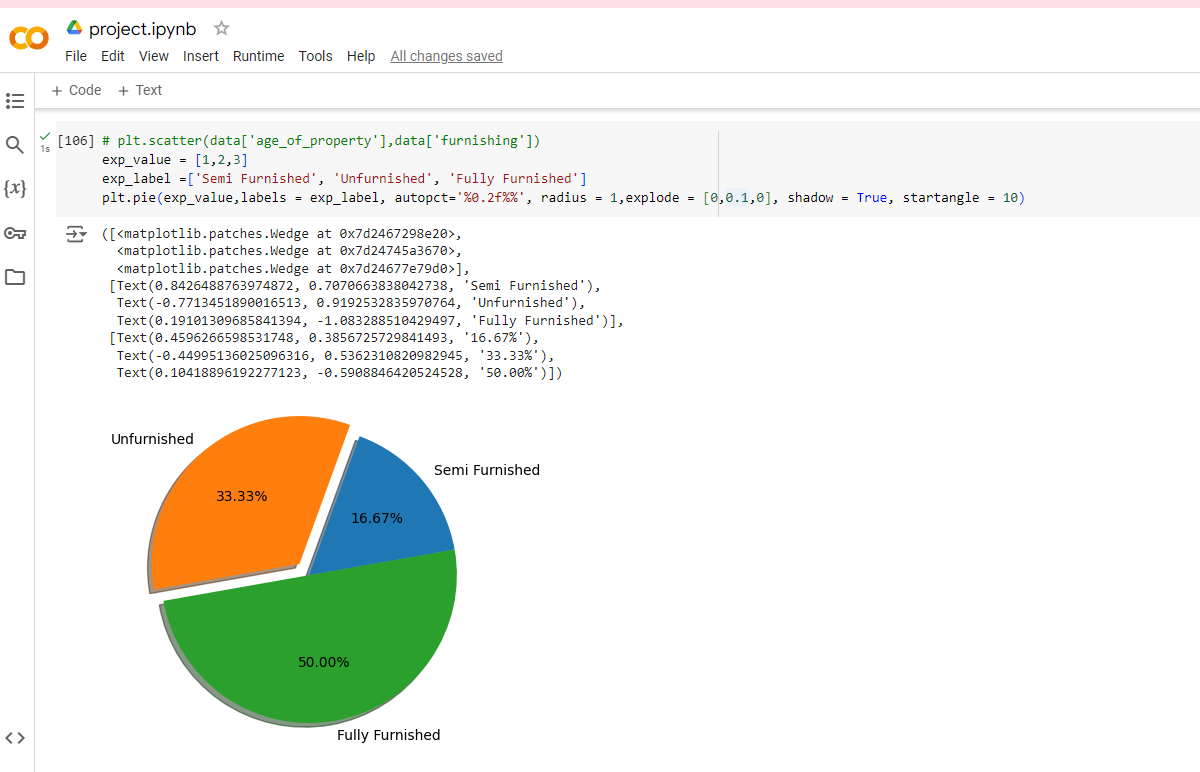


Figure 9: Pie chart (student Result column)

### 6.3.5 Machine learning

Machine learning is a powerful technique that allows computers to learn patterns and make predictions from data without being explicitly programmed. Your flat price prediction model likely leverages algorithms that learn from historical flat prices and features such as location, size, amenities, etc.

machine learning model for flat price prediction is a blend of data science, statistical analysis, and algorithmic computation aimed at making accurate predictions based on historical patterns in flat pricing data.

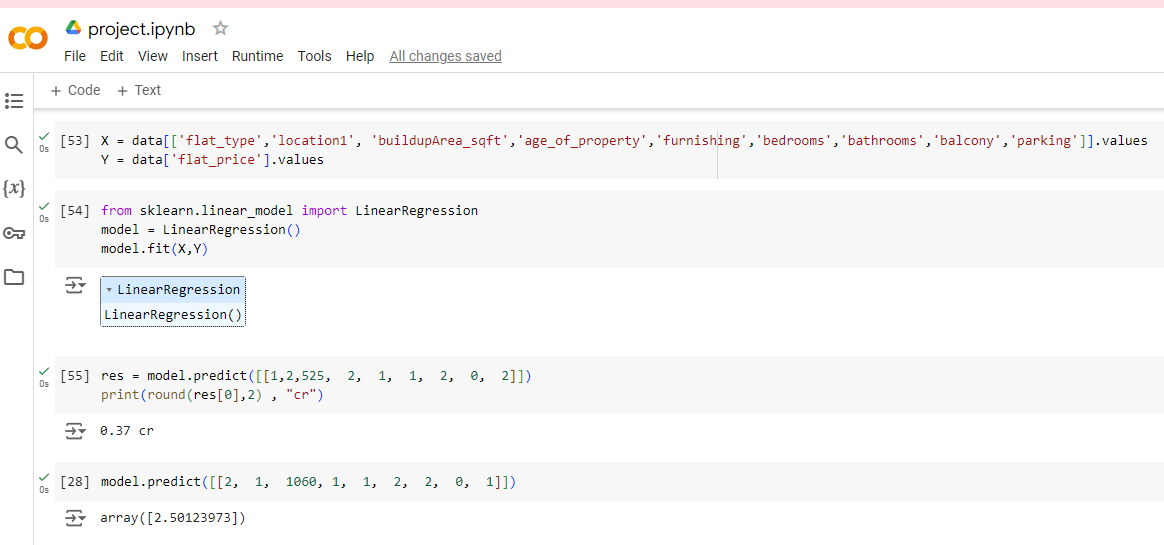
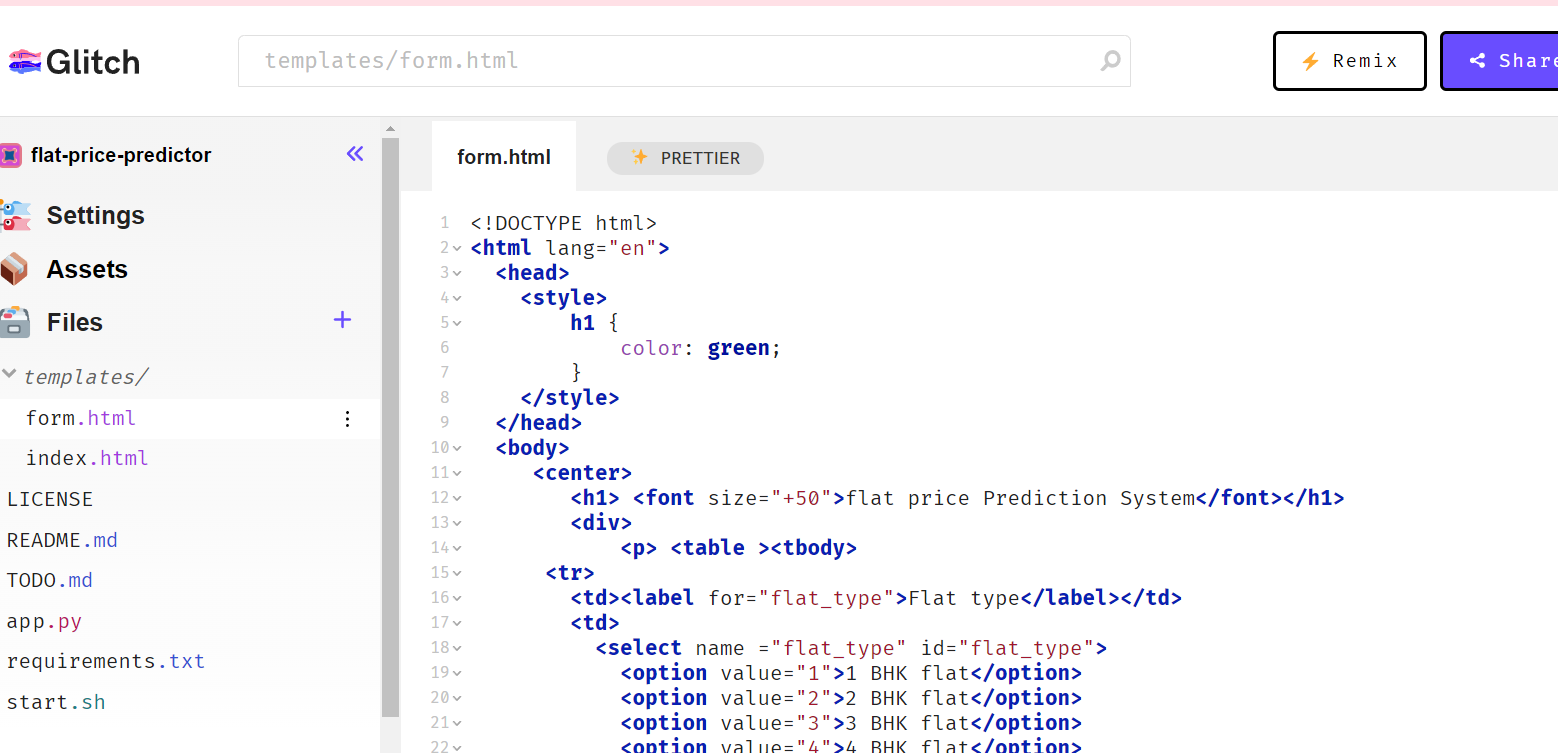


Figure 10 Machine Learning

## 6.4 Front End

### 6.4.1 : HTML / CSS/ Javascript

Figure 11 Front End

<https://glitch.com/edit/#!/flat-price-predictor?path=templates%2Fform.html%3A76%3A104>

### 6.4.2 Web server Development using Flask

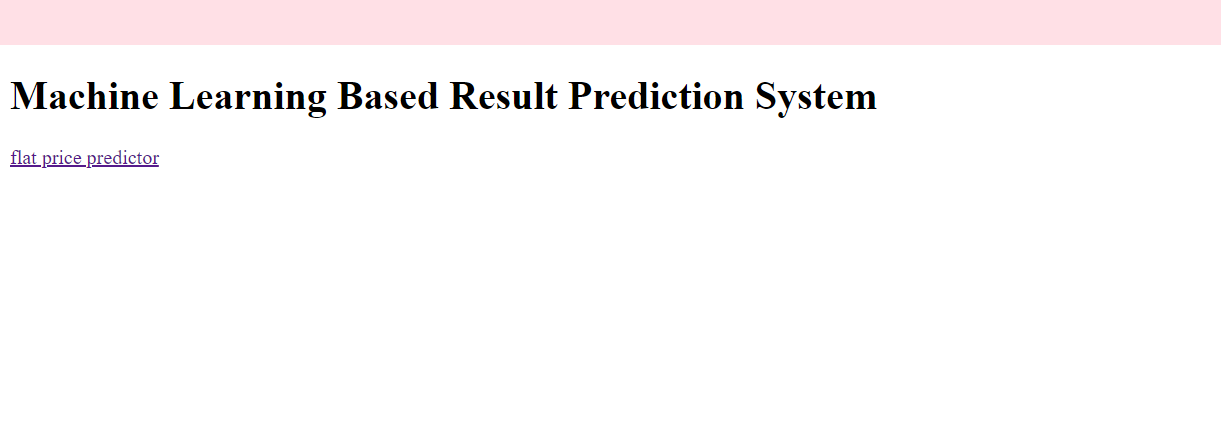


Figure 12web server development

## 6.5 Project walkthrough

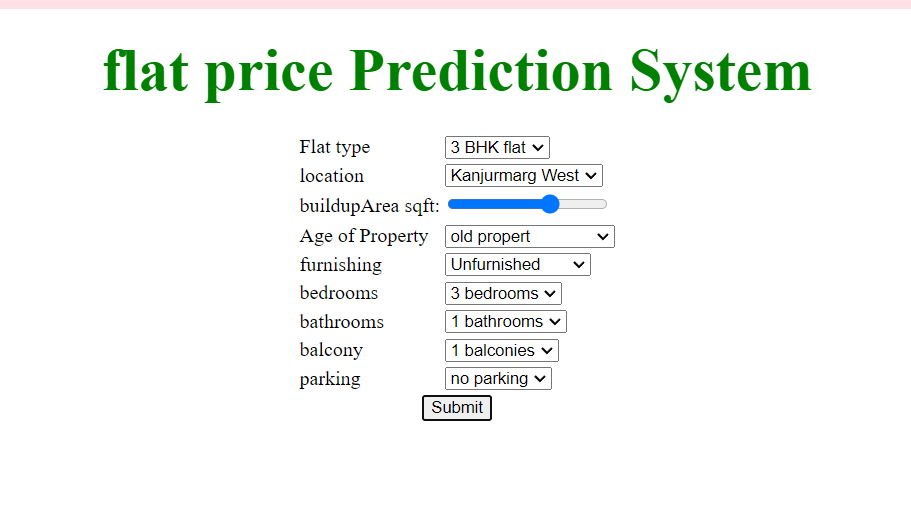


Figure 13 Prediction Model

List of Program Code

matpotlib

<https://colab.research.google.com/drive/1N545PqsqpoptTQm6H7sRgTIgfN9nPAVA?usp=sharing>

assignment

<https://colab.research.google.com/drive/1kaUZpyRwd4DWVil0Z4cFaKrAAJvGT5wP?usp=sharing>

model

<https://colab.research.google.com/drive/1QW_nsSytIkcpWBVJeLELwkMQswGHf1ax?usp=sharing>

numpy

<https://colab.research.google.com/drive/1aB60JY59uE1fy1NUFYypc5yNSWEf4lcm?usp=sharing>

pandas

<https://colab.research.google.com/drive/1N51snsiVa3C8GOhvpuASMrwopJyyNfUc?usp=sharing>

# Bibliography

* [W3Schools Online Web Tutorials](https://www.w3schools.com/)
* [Glitch: The friendly community where everyone builds the web](https://glitch.com/)
* [https://colab.research.google.com](https://colab.research.google.com/)