

# **E-STATE PRICE PREDICTION**

## **A MINOR PROJECT REPORT**

*Submitted in partial Fulfilment of the  
requirement for the award of the degree  
Of*

**Master of Computer Applications (MCA)**

**HARSH NEGI**

**23FS20MCA00002**

**CHETANYA SONI**

**23FS20MCA00030**



**MANIPAL UNIVERSITY  
JAIPUR**

**Department of Computer Applications**

**Manipal University Jaipur, RAJ**

**Dec 2024**

## DECLARATION

I, Harsh Negi & Chetanya Soni, declare that my project, "**E-State Price Prediction**" represents my hard work for my **Master of Computer Applications (MCA)**. This project highlights my dedication to building an innovative e-commerce platform.

The work embodied in this report is the result of my own independent efforts, and all the information, data, and results included are authentic to the best of my knowledge. I have duly acknowledged all sources of data, materials, or assistance utilized during the completion of this project.

This project was undertaken as part of the curriculum requirement and submitted in partial fulfillment of the requirements for the degree of **Master of Computer Applications (MCA)**. It has not been submitted elsewhere for the award of any degree, diploma, or certification.

I extend my sincere gratitude to my guide, faculty members, and my institution, **Manipal University Jaipur**, for their support, encouragement, and resources that made this project possible.

Lastly, I take full responsibility for any errors or omissions found in the report and welcome constructive feedback for improvement

Signature:

Harsh Negi

Chetanya

Soni

DEPARTMENT OF COMPUTER APPLICATION  
MANIPAL UNIVERSITY JAIPUR, JAIPUR-303007 (RAJASTHAN) INDIA

19-12-2024

**CERTIFICATE**

This is to certify that **Harsh Negi & Chetanya Soni (23F520MCA00083 & 23FS20MCA00030)**, a student of **Manipal University Jaipur**, has successfully completed the project titled "**E-Real Estate Price Prediction**" as part of the curriculum requirements for the **Master of Computer Applications (MCA)** program in the **3rd Semester**. Completed during the period from 30.07.2024 to 19.12.2024

The project involves developing a predictive model to estimate real estate prices using machine learning techniques. The work showcases the application of data preprocessing, feature engineering, and predictive modeling using algorithms, with the aim of providing accurate and efficient price predictions.

This project was carried out under the guidance of **[Dr. Linesh Raja ]** in the academic session **2024-2025**.

**Dr. Linesh Raja**

*Project Guide, Dept of Computer Application*

*Manipal University Jaipur*

**Dr. Shilpa Sharma**

*HOD, Dept of Computer Application*

*Manipal University Jaipur*

## ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have supported and guided me throughout the development of my project titled "**E-Real Estate Price Prediction**".

First and foremost, I am deeply grateful to **Dr. Linesh Raja**, my project guide, for their valuable guidance, constructive feedback, and constant encouragement during the course of this project. Their insights and expertise played a significant role in the successful completion of this work.

I extend my heartfelt thanks to **Manipal University Jaipur** and the **Department of Computer Applications** for providing me with the opportunity and resources to undertake this project. The supportive academic environment and infrastructure greatly contributed to my learning experience.

I would also like to thank my friends and classmates for their support and collaboration, which motivated me to overcome challenges during this journey.

This project would not have been possible without the contributions and assistance of all these individuals.

Date: 19.12.2024

Place: Manipal University Jaipur

Harsh Negi

23FS20MCA00083

Chetanya Soni

23FS20MCA00030

## ABSTRACT

The real estate industry is one of the most dynamic and influential sectors, where property prices are subject to various factors such as location, size, amenities, market trends, and economic conditions. Accurate price prediction in real estate can significantly benefit buyers, sellers, and investors by enabling data-driven decision-making.

This project, "**E-Real Estate Price Prediction**", focuses on building a machine learning-based model to predict real estate prices. The project involves data collection, preprocessing, and analysis of various attributes influencing property prices. Advanced regression techniques and machine learning algorithms, such as Linear Regression, Decision Trees, and Random Forest, have been employed to develop a robust predictive model.

The system is designed to provide precise and reliable price estimations based on user-provided parameters, including property location, size, number of bedrooms, and other relevant features. The goal is to make real estate transactions more transparent and efficient by leveraging data science techniques.

This project not only showcases the integration of technology in real estate but also highlights the importance of data analysis and predictive modeling in solving real-world problems.

**Keywords:** Real Estate, Price Prediction, Machine Learning, Data Analysis, Regression Models

Real estate price prediction involves using data-driven models to estimate property values based on a range of influencing factors. The process leverages machine learning, statistical analysis, and domain knowledge to identify patterns and trends that impact prices.

Key components include:

- **Data Collection:** Gathering data on properties, including location, size, amenities, market trends, and economic indicators.
- **Feature Engineering:** Identifying and transforming relevant features like location coordinates, proximity to schools, or neighborhood crime rates.
- **Model Development:** Choosing algorithms (e.g., linear regression, decision trees, or neural networks) and training them on historical data.
- **Evaluation:** Assessing model performance using metrics like RMSE (Root Mean Square Error) or R-squared.

## TABLE OF CONTENT

CONTENTS	PAGE NO.
1. Introduction	7 - 9
1.1. Objective	8
1.2. Purpose & Scope	9
2. Survey of Technology	10 - 11
3. Requirement Analysis	12 - 15
3.1. Problem Definition	12
3.2. Drawback of Existing System	12
3.3. Requirement Specification	13
3.4. Feasibility Study	13-15
4. Planning and Scheduling	16
5. System Design	17 - 26
5.1. Data Flow Diagram	17
5.2. ER Diagram	18-21
5.3. Data Dictionary and Data Model	22- 26
6. Coding Section	27 - 40
7. Testing	41
8. Limitation and Future Scope	42
9. References	45

# 1. INTRODUCTION

The real estate market plays a vital role in the global economy, influencing financial decisions at both individual and organizational levels. Property prices are determined by numerous factors, including geographical location, market trends, property type, amenities, and macroeconomic variables. However, accurately predicting real estate prices remains a challenging task due to the complexity and variability of these factors.

With the advent of advanced data analytics and machine learning, it has become possible to analyze large datasets and identify patterns that were previously undetectable. The objective of this project, "**E-Real Estate Price Prediction**", is to develop a predictive model that utilizes historical real estate data to provide accurate price estimations for properties.

The project involves collecting and preprocessing real estate datasets, exploring features that influence property prices, and applying machine learning algorithms to build a robust predictive system. The model aims to assist various stakeholders, including buyers, sellers, investors, and real estate professionals, by enabling informed decision-making.

The application of machine learning in real estate price prediction not only enhances accuracy but also reduces the time and effort required for manual market analysis. This project underscores the significance of integrating technology into the real estate sector to improve transparency, efficiency, and user experience.

This document provides an overview of the methodology, tools, and techniques used in developing the predictive model, along with its applications and potential impact on the real estate industry.

The real estate industry is a cornerstone of the global economy, with property values significantly influencing financial markets, individual wealth, and economic trends. Accurately predicting real estate prices is a complex yet essential task for various stakeholders, including buyers, sellers, investors, and policymakers. This process involves analyzing historical and current data to forecast the value of properties, enabling better decision-making.

## ☐ **For Buyers and Sellers:**

- Helps sellers set competitive prices based on market trends.
- Assists buyers in identifying fair deals and avoiding overpayment.

## ☐ **For Investors:**

- Enables identification of high-return investment opportunities.
- Supports risk assessment by predicting future market trends.

## ☐ **For Governments and Policymakers:**

- Informs housing policies and urban development projects.
- Aids in monitoring market health to prevent housing bubbles.

### ***1.1. Objective***

The primary objectives of the project "**E-Real Estate Price Prediction**" are as follows:

1. **To Analyze Real Estate Data**
  - Collect and preprocess historical real estate data to understand key features and trends that influence property prices.
2. **To Develop a Predictive Model**
  - Design and implement a machine learning-based predictive model to accurately estimate real estate prices based on user-input features such as location, size, number of rooms, and amenities.
3. **To Enhance Decision-Making**
  - Provide stakeholders, including buyers, sellers, and investors, with a reliable tool for making data-driven decisions in property transactions.
4. **To Improve Accuracy and Efficiency**
  - Use advanced regression techniques and machine learning algorithms to enhance the accuracy of price predictions compared to traditional estimation methods.
5. **To Increase Market Transparency**
  - Leverage technology to create a transparent and efficient pricing mechanism, reducing the reliance on manual market analysis.
6. **To Explore Real-World Applications**
  - Investigate the practical applications of the model in the real estate industry and its potential to address challenges in pricing dynamics.
7. **To Bridge the Gap Between Data and Real Estate**
  - Demonstrate the integration of data science and machine learning in solving real-world problems in the real estate sector.

### ***1.2. Purpose & Scope***

#### ***Purpose***

The primary purpose of the project "**E-Real Estate Price Prediction**" is to create an efficient and reliable system for estimating property prices in the ever-evolving real estate market. Real estate pricing is influenced by a variety of factors, such as location, property size, amenities, and prevailing market trends, making accurate valuation a complex task. This project aims to simplify the process of price estimation by leveraging machine learning techniques to analyze historical data and predict prices with high accuracy.

The purpose extends to empowering stakeholders such as buyers, sellers, and investors by providing them with data-driven insights for making informed decisions. By integrating advanced predictive models, this project seeks to reduce reliance on traditional and often subjective valuation methods, thereby increasing market transparency and efficiency.

Furthermore, the project highlights the practical application of modern technology in addressing real-world challenges, demonstrating how data science can transform industries like real estate. Ultimately, the purpose of this initiative is to contribute to a more transparent, accurate, and user-friendly real estate ecosystem while encouraging innovation and technological advancement in the sector.



## ***Scope***

The project "**E-Real Estate Price Prediction**" focuses on creating a machine learning-based system capable of accurately predicting real estate prices by analyzing factors such as location, property size, number of rooms, and amenities. The system is intended to assist buyers, sellers, and investors in making informed decisions, reducing reliance on subjective pricing methods, and improving market transparency and efficiency.

The scope of the project includes the use of advanced machine learning algorithms for predictive modeling, ensuring accurate and reliable results. It can be extended to incorporate real-time data updates, regional customization for specific markets, and additional features like price trend analysis and future forecasting.

Furthermore, the project has potential applications as a web or mobile-based tool, enabling easy accessibility for users. In the long term, it can contribute to transforming the real estate sector by integrating data-driven insights into property valuation, ultimately making the buying and selling process more transparent and efficient for all stakeholders.

## **. Applications Across Stakeholders**

- **For Homebuyers and Sellers:**
  - Provide personalized property valuation.
  - Assist in negotiating prices based on market trends.
- **For Real Estate Agents and Brokers:**
  - Enhance pricing strategies for listings.
  - Improve market analysis and sales forecasting.
- **For Investors and Developers:**
  - Identify undervalued properties and high-growth areas.
  - Forecast long-term returns on investment.
- **For Financial Institutions:**
  - Evaluate property collateral for loans and mortgages.
  - Assess risks associated with lending.
- **For Government and Urban Planners:**
  - Aid in policy development and housing initiatives.
  - Monitor market health to avoid housing bubbles.

## 2. SURVEY OF TECHNOLOGY

The field of real estate price prediction is rapidly evolving with advancements in data science and machine learning. The integration of these technologies allows for more accurate and reliable predictions that can assist buyers, sellers, investors, and real estate professionals in making informed decisions. In this project, we leverage several modern technologies to build an efficient and scalable real estate price prediction system. The following technologies and methodologies are fundamental to the development of the **E-Real Estate Price Prediction** model:

### 1. *Machine Learning Algorithms*

Machine learning is at the core of this project, enabling the system to learn from historical real estate data and predict property prices based on relevant features.

- **Linear Regression:** A simple yet effective algorithm used to model the relationship between the dependent variable (price) and independent variables (features like area, number of rooms).
- **Random Forest:** An ensemble method that creates multiple decision trees and combines their predictions to improve accuracy, especially when dealing with complex data.
- **Gradient Boosting:** This algorithm improves model performance by correcting the errors made by previous models, making it highly effective for non-linear relationships between features and property prices.

These algorithms are implemented using **Scikit-learn**, a popular Python library for machine learning.

### 2. *Data Preprocessing*

Data preprocessing ensures the dataset is clean and suitable for training machine learning models. It includes steps such as:

- **Handling Missing Data:** Imputing missing values using techniques like mean imputation or dropping rows with missing values.
- **Feature Encoding:** Converting categorical variables, such as property type or location, into numeric format using methods like one-hot encoding.
- **Feature Scaling:** Normalizing numerical features to ensure that the model does not favor larger values, using techniques like **Standardization** or **Min-Max Scaling**.

### 3. *Programming Language: Python*

Python is used for implementing machine learning models, data processing, and visualization due to its simplicity and powerful libraries.

- **Pandas:** Used for data manipulation and analysis, such as importing, cleaning, and transforming datasets.
- **NumPy:** Essential for numerical computations and handling arrays.

- **Scikit-learn:** Provides functions to implement machine learning algorithms and evaluate models.
- **Matplotlib and Seaborn:** Used for data visualization, helping to visualize trends, correlations, and model performance.

#### ***4. Data Visualization***

Data visualization is an important part of the project, helping users and stakeholders interpret the data and model predictions. **Matplotlib** and **Seaborn** are used to create various visualizations such as:

- **Correlation Heatmaps:** To see how features are related to the target variable (price).
- **Scatter Plots:** To visualize relationships between features like size and price.
- **Prediction Accuracy Plots:** To compare predicted prices against actual prices and assess model performance.

#### ***5. Web Development (Optional)***

If the system is deployed as a web application, frameworks like **Flask** or **Django** can be used for backend development. **HTML**, **CSS**, and **JavaScript** can be used for the frontend, providing a user-friendly interface for users to input property details and view predicted prices.

#### ***6. Cloud Computing and Real-Time Data***

To deploy the application and manage large datasets, cloud platforms like **AWS** or **Google Cloud** can be used. Real-time data integration through APIs can provide up-to-date property information, ensuring the system remains current and scalable.

By combining these technologies, the project aims to provide a scalable, efficient, and accurate system for real estate price prediction. This integration of machine learning, data preprocessing, and web development will help stakeholders make informed decisions based on reliable predictions.

## 3. REQUIREMENT ANALYSIS

### 3.1. Problem Definition

The real estate market is highly dynamic, with prices being influenced by various factors like property features, location, amenities, and historical trends. Predicting real estate prices is a complex task that involves analyzing large datasets, understanding market trends, and accounting for external influences.

The **problem** with the traditional method of pricing properties is that it often relies on subjective assessments or outdated market data. Real estate agents, buyers, and investors struggle to find reliable methods to determine accurate property values.

The goal of this project is to develop an automated **E-Real Estate Price Prediction** system that uses machine learning algorithms to predict property prices based on a variety of factors. By leveraging data and advanced algorithms, the system aims to provide accurate, real-time predictions to stakeholders in the real estate market, making the pricing process faster, more transparent, and data-driven.

### 3.2. Drawbacks of Existing Systems

The existing systems used for real estate price prediction often have several limitations:

1. **Lack of Real-Time Data:**

Many existing systems rely on outdated or static datasets. As a result, predictions are often not aligned with current market trends, which can lead to significant pricing inaccuracies.

2. **Limited Accuracy:**

Traditional methods often struggle to incorporate the vast number of variables that affect property prices. Manual evaluations or simplistic models may not capture the complexity of the market, leading to inaccurate predictions.

3. **Inefficient Processing:**

Many systems are not optimized for processing large datasets or scaling to meet the growing demands of the real estate industry. These systems may experience performance issues when handling complex queries or real-time predictions.

4. **Limited User Interactivity:**

Existing systems often have basic or non-interactive user interfaces. This limits the ability of users to experiment with different property features or explore the underlying factors influencing predictions.

5. **Lack of Personalization:**

Many systems do not take into account the unique needs of different user groups, such as buyers, sellers, investors, or real estate agents. A one-size-fits-all approach makes the system less useful for specific market needs.

### 3.3. Requirement Specification

Based on the problem definition and analysis of existing systems, the **E-Real Estate Price Prediction** system

must meet the following specifications:

### **Functional Requirements:**

- **Data Collection:** The system should be able to collect property data such as location, size, number of rooms, age, amenities, historical sale prices, and market trends.
- **Data Preprocessing:** The system should handle missing data, encode categorical features, scale numerical values, and detect outliers for improved model performance.
- **Prediction Models:** The system must support various machine learning algorithms (Linear Regression, Random Forest, Gradient Boosting) to predict property prices based on input data.
- **Real-Time Predictions:** Users should be able to input property details and receive immediate price predictions.
- **Visualization:** The system should present prediction results with graphs, price trends, and visual insights for users to interpret.
- **User Interface:** The system should have an intuitive and interactive web interface for users to input data, view results, and understand the model's output.

### **Non-Functional Requirements:**

- **Accuracy:** The system must provide high accuracy in predicting property prices, with error metrics such as MAE, RMSE, and  $R^2$  for evaluation.
- **Scalability:** The system should be scalable to handle large datasets, increasing user demand, and the addition of new features over time.
- **Performance:** The system should process predictions quickly, with minimal delay in generating results.
- **Security:** The system must ensure that user data is secure and protected from unauthorized access.
- **Usability:** The system should be easy to use, with a simple interface and clear instructions for all types of users.

## **3.4. Feasibility Study**

The feasibility of the **E-Real Estate Price Prediction** system can be evaluated in terms of its technical, operational, and economic viability.

### **1. Technical Feasibility**

- **Data Availability:**
  - **Sources:** Public property records, real estate APIs (e.g., Zillow, Redfin), census data, GIS systems, and economic indicators.
  - **Challenges:** Ensuring data completeness, consistency, and legality in accessing proprietary datasets.
- **Technology Requirements:**
  - **Infrastructure:** Cloud-based platforms (e.g., AWS, Google Cloud) for scalability and storage.
  - **Tools:** Python, R, and machine learning frameworks (e.g., TensorFlow, Scikit-learn, XGBoost).
  - **Hardware:** High-performance servers or GPUs for training complex models.
- **Expertise:**

- Availability of data scientists, machine learning engineers, and domain experts.
- Need for GIS analysts for geospatial data integration.
- **Scalability:**
  - Models must handle expanding datasets as the system grows geographically or in market segments.

## 2. Economic Feasibility

- **Initial Investment:**
  - Costs for data acquisition, model development, and infrastructure setup.
  - Hiring skilled professionals.
- **Operational Costs:**
  - Ongoing expenses for cloud services, model maintenance, and data updates.
- **Revenue Potential:**
  - Monetization through SaaS platforms for real estate agents, investors, or individual buyers.
  - Subscription models for providing market insights or API access.
- **Cost-Benefit Analysis:**
  - Higher upfront costs can be offset by long-term revenue from accurate predictions and expanded user bases.

## 3. Market Feasibility

- **Demand Analysis:**
  - Increasing demand for data-driven insights in real estate transactions.
  - Interest from diverse stakeholders: agents, buyers, sellers, investors, and governments.
- **Competition:**
  - Established players like Zillow and Redfin already offer valuation tools.
  - Potential to differentiate by improving accuracy, incorporating local nuances, and providing tailored insights.
- **Market Size:**
  - Real estate is a multi-trillion-dollar global industry with significant investment in data analytics.

## 4. Operational Feasibility

- **Stakeholder Collaboration:**
  - Collaboration with real estate agencies, financial institutions, and government bodies to access reliable data.
- **User Accessibility:**
  - Developing intuitive interfaces (e.g., web dashboards, mobile apps) for non-technical users.
- **Regulatory Compliance:**
  - Adhering to data protection laws like GDPR or CCPA when handling user data.

## 5. Legal and Ethical Feasibility

- **Data Privacy:**
  - Implement secure data handling practices to protect sensitive property and user data.
- **Bias Mitigation:**
  - Ensure algorithms do not perpetuate discrimination based on socioeconomic or geographic factors.
- **Regulatory Requirements:**
  - Compliance with local and national regulations related to property data and predictions.

## 6. Risk Analysis

- **Technical Risks:**
  - Challenges in handling incomplete or biased datasets.
  - Model performance degradation due to dynamic market changes.
- **Market Risks:**
  - Competition from established players.
  - Difficulty in convincing traditional stakeholders to adopt advanced analytics.
- **Financial Risks:**
  - High initial costs may not yield immediate returns.
  - Potential for limited adoption in certain markets.

## 7. Recommendations

- Start with a pilot project targeting a specific city or market segment to test the model's accuracy and usability.
- Partner with local real estate agencies to access high-quality data and gain market insights.
- Focus on user-friendly design and actionable insights to differentiate from competitors.
- Develop scalable and adaptive systems to accommodate market changes and future growth.

## 4. PLANNING AND SCHEDULING

Effective planning and scheduling are critical to ensuring the timely completion of the project. The development of the **E-Real Estate Price Prediction** system can be broken down into several phases, each with its own set of tasks and milestones.

### 4.1. Project Phases and Tasks

1. **Requirement Analysis and Research (Week 1-2):**
  - Understand the problem and define requirements.
  - Research existing systems and tools.
  - Collect real estate data for the project.
2. **System Design and Architecture (Week 3-4):**
  - Design the system architecture and choose the appropriate technologies.
  - Create wireframes or prototypes for the user interface.
3. **Data Collection and Preprocessing (Week 5-6):**
  - Collect and clean the dataset.
  - Handle missing values, outliers, and categorical features.
4. **Model Development and Evaluation (Week 7-8):**
  - Implement machine learning models.
  - Train models on the dataset and evaluate performance.
  - Tune the models to improve accuracy.
5. **User Interface Development (Week 9-10):**
  - Develop the frontend using HTML, CSS, and JavaScript.
  - Create interactive forms for input and display results.
6. **Testing and Deployment (Week 11-12):**
  - Conduct unit and integration testing.
  - Deploy the system to the cloud for hosting.
  - Ensure the system is stable and secure.
7. **Final Review and Documentation (Week 13):**
  - Review the project outcomes and ensure all requirements are met.
  - Document the system design, code, and user guide.

### 4.2. Gantt Chart

A **Gantt Chart** can be used to visualize the project schedule and track progress over time. The chart will include tasks, deadlines, and responsible team members for each phase.



## 5. SYSTEM DESIGN

### 5.1 Data Flow Diagram

A real estate system connects buyers, sellers, and agents through a platform offering property listings, search filters, interactive maps, and secure payment options. Users can browse or post properties with detailed information like price, location, and amenities, while agents and admins manage listings and user interactions through dashboards. The system's architecture includes a front-end (React/Angular) for a responsive interface, a back-end (Node.js/Django) handling APIs and business logic, and a database (PostgreSQL/MongoDB) for storing user and property data. Integrations like Google Maps, payment gateways, and notification systems enhance functionality, with cloud hosting ensuring scalability and security through encryption, authentication, and regular audits.

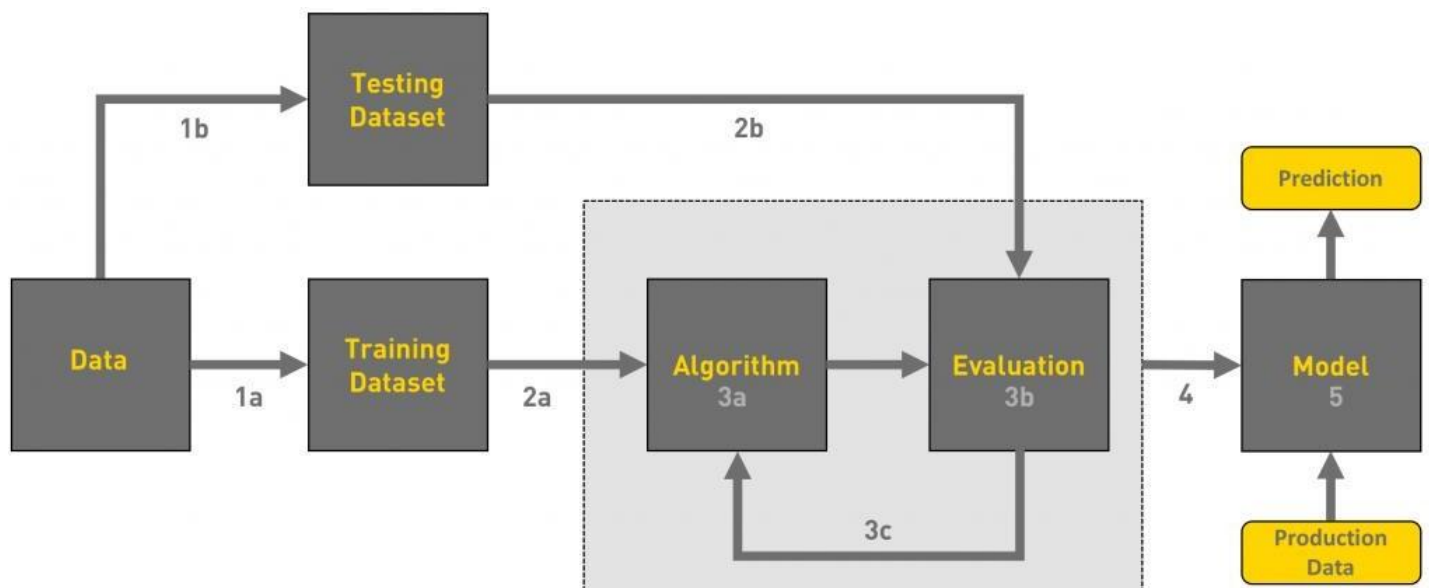


Fig ( 5.1 ) DATA FOLW DAIGRAM

## ER – DAIGRAM

Below is an **Entity-Relationship (ER) Diagram** description for a Real Estate Price Prediction system. This system includes entities, their attributes, and relationships for organizing the data effectively.

### Entities and Attributes:

#### 1. Property

- **Attributes:**
  - Property\_ID (Primary Key)
  - Address
  - Zip\_Code
  - City
  - State
  - Size (in square feet)
  - Number\_of\_Bedrooms
  - Number\_of\_Bathrooms
  - Year\_Built
  - Property\_Type (e.g., residential, commercial)
  - Listing\_Price
  - Sold\_Price
  - Date\_Listed
  - Date\_Sold

#### 2. Location

- **Attributes:**
  - Location\_ID (Primary Key)
  - Neighborhood\_Name
  - Crime\_Rate
  - School\_Proximity

- Transport\_Accessibility
- Average\_Income
- Population

### **3. Market**

- **Attributes:**

- Market\_ID (Primary Key)
- Market\_Name (e.g., city or regional market)
- Interest\_Rate
- Economic\_Growth\_Rate
- Unemployment\_Rate
- Housing\_Demand\_Index
- Date\_Updated

### **4. User**

- **Attributes:**

- User\_ID (Primary Key)
- Name
- Email
- Role (e.g., buyer, seller, agent, investor)
- Registration\_Date

### **5. Prediction**

- **Attributes:**

- Prediction\_ID (Primary Key)
- Property\_ID (Foreign Key)
- Predicted\_Price
- Prediction\_Date
- Model\_Used
- Accuracy\_Score

## 6. Transaction

- **Attributes:**

- Transaction\_ID (Primary Key)
- Property\_ID (Foreign Key)
- Buyer\_ID (Foreign Key from User)
- Seller\_ID (Foreign Key from User)
- Agent\_ID (Foreign Key from User, if applicable)
- Transaction\_Date
- Transaction\_Amount

### **Relationships:**

#### **1. Property and Location:**

- A property belongs to one location.
- A location can have many properties.
- Relationship: **1:N (Location:Property)**

#### **2. Market and Location:**

- A market can cover multiple locations.
- A location belongs to one market.
- Relationship: **1:N (Market:Location)**

#### **3. Property and Prediction:**

- A property can have multiple predictions over time.
- A prediction is for one property.
- Relationship: **1:N (Property:Prediction)**

#### **4. Transaction and Property:**

- A transaction involves one property.
- A property can have multiple transactions over time.
- Relationship: **1:N (Property:Transaction)**

## 5. Transaction and User:

- A transaction involves a buyer, seller, and optionally an agent.
- A user can participate in multiple transactions in different roles.
- Relationship: **1:N (User:Transaction)**

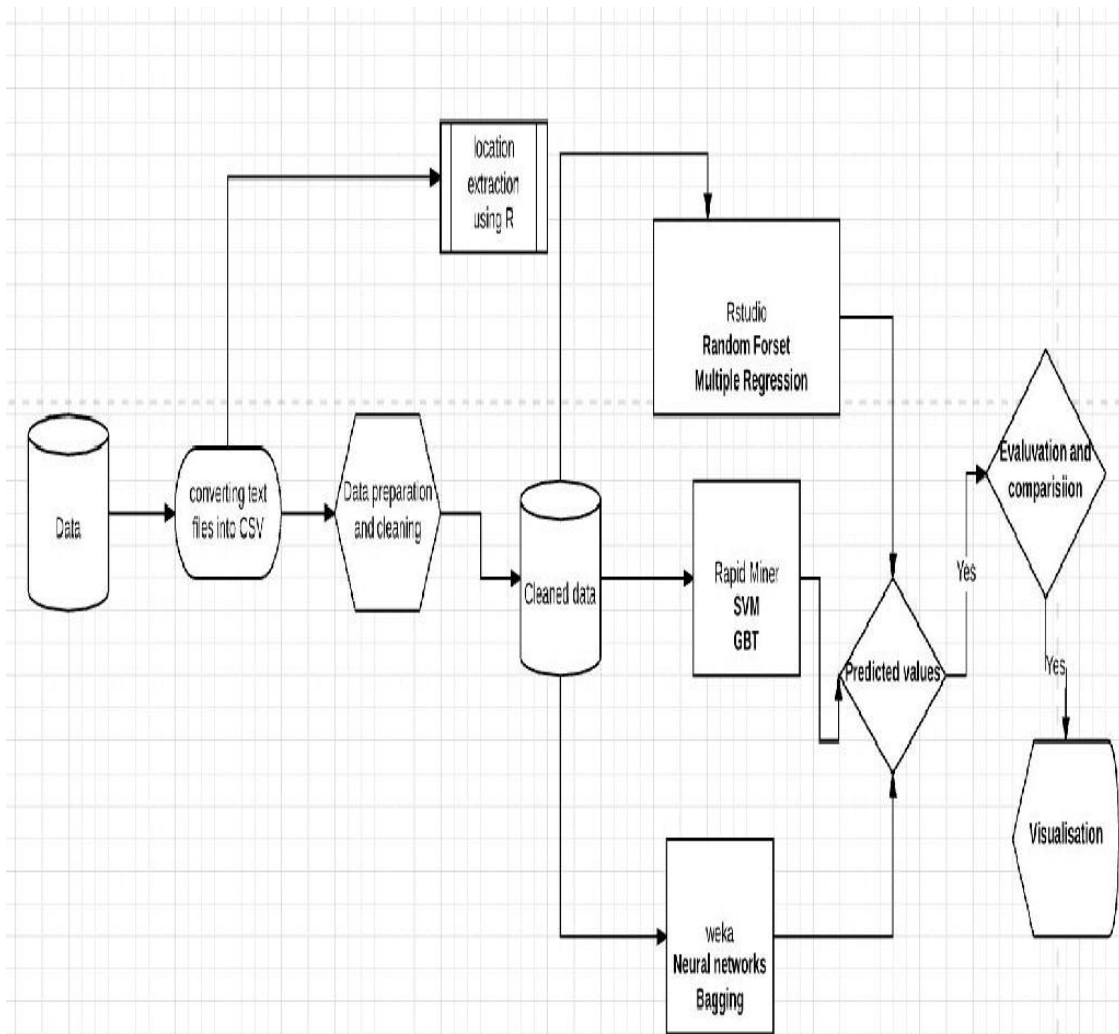


Fig (5.2) ER-DAIGRAM

## Data Dictionary

A **data dictionary** provides a detailed description of the data elements used in the system, including attributes, data types, and constraints. Below is a comprehensive data dictionary for the entities defined in the ER diagram.

### 1. Property

Attribute	Data Type	Description	Constraints
Property_ID	Integer	Unique identifier for each property	Primary Key, Auto-increment
Address	String	Full address of the property	Not Null
Zip_Code	String	Zip code of the property's location	Not Null
City	String	City where the property is located	Not Null
State	String	State where the property is located	Not Null
Size	Float	Property size in square feet	Not Null, > 0
Number_of_Bedrooms	Integer	Number of bedrooms in the property	Not Null, >= 0
Number_of_Bathrooms	Float	Number of bathrooms in the property	Not Null, >= 0
Year_Built	Integer	Year the property was constructed	Not Null, > 0
Property_Type	String	Type of property (e.g., residential, commercial)	Not Null
Listing_Price	Float	Price at which the property is listed	Not Null, >= 0
Sold_Price	Float	Final sale price of the property	Nullable
Date_Listed	Date	Date when the property was listed	Not Null

Attribute	Data Type	Description	Constraints
Date_Sold	Date	Date when the property was sold	Nullable

## 2. Location

Attribute	Data Type	Description	Constraints
Location_ID	Integer	Unique identifier for each location	Primary Key, Auto-increment
Neighborhood_Name	String	Name of the neighborhood	Not Null
Crime_Rate	Float	Crime rate in the neighborhood (per 1,000 people)	Not Null, $\geq 0$
School_Proximity	Float	Distance to the nearest school (in km)	Not Null, $\geq 0$
Transport_Accessibility	Float	Distance to the nearest transport hub (in km)	Not Null, $\geq 0$
Average_Income	Float	Average income of residents in the area	Not Null, $\geq 0$
Population	Integer	Population of the neighborhood	Not Null, $> 0$

## 3. Market

Attribute	Data Type	Description	Constraints
Market_ID	Integer	Unique identifier for each market	Primary Key, Auto-increment
Market_Name	String	Name of the market (e.g., city or	Not Null

<b>Attribute</b>	<b>Data Type</b>	<b>Description</b>	<b>Constraints</b>
		region)	
Interest_Rate	Float	Current interest rate	Not Null, $\geq 0$
Economic_Growth_Rate	Float	Annual economic growth rate (in %)	Not Null
Unemployment_Rate	Float	Unemployment rate in the market area	Not Null
Housing_Demand_Index	Float	Index representing housing demand	Not Null
Date_Updated	Date	Last updated date for market information	Not Null

#### 4. User

<b>Attribute</b>	<b>Data Type</b>	<b>Description</b>	<b>Constraints</b>
User_ID	Integer	Unique identifier for each user	Primary Key, Auto-increment
Name	String	Full name of the user	Not Null
Email	String	Email address of the user	Not Null, Unique
Role	String	Role of the user (buyer, seller, agent, etc.)	Not Null
Registration_Date	Date	Date of user registration	Not Null



## 5. Prediction

Attribute	Data Type	Description	Constraints
Prediction_ID	Integer	Unique identifier for each prediction	Primary Key, Auto-increment
Property_ID	Integer	Identifier of the property being predicted	Foreign Key (Property)
Predicted_Price	Float	Predicted price of the property	Not Null, $\geq 0$
Prediction_Date	Date	Date when the prediction was made	Not Null
Model_Used	String	Name of the model used for prediction	Not Null
Accuracy_Score	Float	Accuracy of the prediction model (%)	Nullable, 0–100

## 6. Transaction

Attribute	Data Type	Description	Constraints
Transaction_ID	Integer	Unique identifier for each transaction	Primary Key, Auto-increment
Property_ID	Integer	Identifier of the property involved	Foreign Key (Property)
Buyer_ID	Integer	Identifier of the buyer	Foreign Key (User)
Seller_ID	Integer	Identifier of the seller	Foreign Key (User)
Agent_ID	Integer	Identifier of the agent (if applicable)	Nullable, Foreign Key (User)
Transaction_Date	Date	Date when the transaction occurred	Not Null
Transaction_Amount	Float	Amount involved in the transaction	Not Null, $\geq 0$

This dictionary ensures a clear understanding of the data structure, supporting seamless implementation and data integrity

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	TAXRM
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	45.019011
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	37.688834
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	33.681280
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	31.723350
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	31.061984

Fig(5.3) DATA SET OF DATA DICTONARY

## 6. CODING SECTION

```
{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {},
      "source": [
        "## Dragon Real Estate - Price Predictor"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {},
      "outputs": [],
      "source": [
        "import pandas as pd"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 2,
      "metadata": {},
      "outputs": [],
      "source": [
        "housing = pd.read_csv(\"data.csv\")"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 3,
      "metadata": {},
      "outputs": [
        {
          "data": {
            "text/html": [
              "<div>\n",
              "<style scoped>\n",
              "  .dataframe tbody tr th:only-of-type {\n",
              "    vertical-align: middle;\n",
              "  }\n",
              "\n",
              "  .dataframe tbody tr th {\n",
              "    vertical-align: top;\n",
              "  }\n",
              "\n",
              "\n"
            ]
          }
        ]
      ]
    }
  ]
}
```

```

" .dataframe thead th {\n",
"   text-align: right;\n",
" } \n",
"</style>\n",
"<table border='1' class='dataframe'>\n",
" <thead>\n",
"   <tr style='text-align: right;'>\n",
"     <th></th>\n",
"     <th>CRIM</th>\n",
"     <th>ZN</th>\n",
"     <th>INDUS</th>\n",
"     <th>CHAS</th>\n",
"     <th>NOX</th>\n",
"     <th>RM</th>\n",
"     <th>AGE</th>\n",
"     <th>DIS</th>\n",
"     <th>RAD</th>\n",
"     <th>TAX</th>\n",
"     <th>PTRATIO</th>\n",
"     <th>B</th>\n",
"     <th>LSTAT</th>\n",
"     <th>MEDV</th>\n",
"   </tr>\n",
" </thead>\n",
" <tbody>\n",
"   <tr>\n",
"     <th>0</th>\n",
"     <td>0.00632</td>\n",
"     <td>18.0</td>\n",
"     <td>2.31</td>\n",
"     <td>0</td>\n",
"     <td>0.538</td>\n",
"     <td>6.575</td>\n",
"     <td>65.2</td>\n",
"     <td>4.0900</td>\n",
"     <td>1</td>\n",
"     <td>296</td>\n",
"     <td>15.3</td>\n",
"     <td>396.90</td>\n",
"     <td>4.98</td>\n",
"     <td>24.0</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>1</th>\n",
"     <td>0.02731</td>\n",
"     <td>0.0</td>\n",
"     <td>7.07</td>\n",
"     <td>0</td>\n",
"     <td>0.469</td>\n",
"     <td>6.421</td>\n",
"     <td>78.9</td>\n",

```

```

"      <td>4.9671</td>\n",
"      <td>2</td>\n",
"      <td>242</td>\n",
"      <td>17.8</td>\n",
"      <td>396.90</td>\n",
"      <td>9.14</td>\n",
"      <td>21.6</td>\n",
"    </tr>\n",
"  <tr>\n",
"    <th>2</th>\n",
"    <td>0.02729</td>\n",
"    <td>0.0</td>\n",
"    <td>7.07</td>\n",
"    <td>0</td>\n",
"    <td>0.469</td>\n",
"    <td>7.185</td>\n",
"    <td>61.1</td>\n",
"    <td>4.9671</td>\n",
"    <td>2</td>\n",
"    <td>242</td>\n",
"    <td>17.8</td>\n",
"    <td>392.83</td>\n",
"    <td>4.03</td>\n",
"    <td>34.7</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>3</th>\n",
"    <td>0.03237</td>\n",
"    <td>0.0</td>\n",
"    <td>2.18</td>\n",
"    <td>0</td>\n",
"    <td>0.458</td>\n",
"    <td>6.998</td>\n",
"    <td>45.8</td>\n",
"    <td>6.0622</td>\n",
"    <td>3</td>\n",
"    <td>222</td>\n",
"    <td>18.7</td>\n",
"    <td>394.63</td>\n",
"    <td>2.94</td>\n",
"    <td>33.4</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>4</th>\n",
"    <td>0.06905</td>\n",
"    <td>0.0</td>\n",
"    <td>2.18</td>\n",
"    <td>0</td>\n",
"    <td>0.458</td>\n",
"    <td>7.147</td>\n",
"    <td>54.2</td>\n",

```

```

"    <td>6.0622</td>\n",
"    <td>3</td>\n",
"    <td>222</td>\n",
"    <td>18.7</td>\n",
"    <td>396.90</td>\n",
"    <td>5.33</td>\n",
"    <td>36.2</td>\n",
"  </tr>\n",
" </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"   CRIM   ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX
PTRATIO  \\\n",
"0  0.00632  18.0   2.31    0  0.538  6.575  65.2  4.0900   1  296   15.3  \n",
"1  0.02731   0.0   7.07    0  0.469  6.421  78.9  4.9671   2  242   17.8  \n",
"2  0.02729   0.0   7.07    0  0.469  7.185  61.1  4.9671   2  242   17.8  \n",
"3  0.03237   0.0   2.18    0  0.458  6.998  45.8  6.0622   3  222   18.7  \n",
"4  0.06905   0.0   2.18    0  0.458  7.147  54.2  6.0622   3  222   18.7  \n",
"\n",
"   B  LSTAT  MEDV  \n",
"0  396.90   4.98  24.0  \n",
"1  396.90   9.14  21.6  \n",
"2  392.83   4.03  34.7  \n",
"3  394.63   2.94  33.4  \n",
"4  396.90   5.33  36.2  "
]
},
"execution_count": 3,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"housing.head()"
]
},
{
"cell_type": "code",
"execution_count": 4,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"<class 'pandas.core.frame.DataFrame'>\n",
"RangeIndex: 506 entries, 0 to 505\n",
"Data columns (total 14 columns):\n",
"CRIM      506 non-null float64\n",

```

```

    "ZN      506 non-null float64\n",
    "INDUS   506 non-null float64\n",
    "CHAS     506 non-null int64\n",
    "NOX      506 non-null float64\n",
    "RM       501 non-null float64\n",
    "AGE      506 non-null float64\n",
    "DIS      506 non-null float64\n",
    "RAD      506 non-null int64\n",
    "TAX      506 non-null int64\n",
    "PTRATIO  506 non-null float64\n",
    "B        506 non-null float64\n",
    "LSTAT    506 non-null float64\n",
    "MEDV     506 non-null float64\n",
    "dtypes: float64(11), int64(3)\n",
    "memory usage: 55.4 KB\n"
  ]
},
"source": [
  "housing.info()"
],
{
  "cell_type": "code",
  "execution_count": 5,
  "metadata": {},
  "outputs": [
    {
      "data": {
        "text/plain": [
          "0   471\n",
          "1    35\n",
          "Name: CHAS, dtype: int64"
        ]
      },
      "execution_count": 5,
      "metadata": {},
      "output_type": "execute_result"
    }
  ],
  "source": [
    "housing['CHAS'].value_counts()"
  ],
  {
    "cell_type": "code",
    "execution_count": 6,
    "metadata": {},
    "outputs": [
      {
        "data": {

```

[illegible]



```

"    <td>506.000000</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>mean</th>\n",
"    <td>3.613524</td>\n",
"    <td>11.363636</td>\n",
"    <td>11.136779</td>\n",
"    <td>0.069170</td>\n",
"    <td>0.554695</td>\n",
"    <td>6.284341</td>\n",
"    <td>68.574901</td>\n",
"    <td>3.795043</td>\n",
"    <td>9.549407</td>\n",
"    <td>408.237154</td>\n",
"    <td>18.455534</td>\n",
"    <td>356.674032</td>\n",
"    <td>12.653063</td>\n",
"    <td>22.532806</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>std</th>\n",
"    <td>8.601545</td>\n",
"    <td>23.322453</td>\n",
"    <td>6.860353</td>\n",
"    <td>0.253994</td>\n",
"    <td>0.115878</td>\n",
"    <td>0.705587</td>\n",
"    <td>28.148861</td>\n",
"    <td>2.105710</td>\n",
"    <td>8.707259</td>\n",
"    <td>168.537116</td>\n",
"    <td>2.164946</td>\n",
"    <td>91.294864</td>\n",
"    <td>7.141062</td>\n",
"    <td>9.197104</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>min</th>\n",
"    <td>0.006320</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.460000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.385000</td>\n",
"    <td>3.561000</td>\n",
"    <td>2.900000</td>\n",
"    <td>1.129600</td>\n",
"    <td>1.000000</td>\n",
"    <td>187.000000</td>\n",
"    <td>12.600000</td>\n",
"    <td>0.320000</td>\n",
"    <td>1.730000</td>\n",

```

```

"    <td>5.000000</td>\n",
"  </tr>\n",
" <tr>\n",
"   <th>25%</th>\n",
"   <td>0.082045</td>\n",
"   <td>0.000000</td>\n",
"   <td>5.190000</td>\n",
"   <td>0.000000</td>\n",
"   <td>0.449000</td>\n",
"   <td>5.884000</td>\n",
"   <td>45.025000</td>\n",
"   <td>2.100175</td>\n",
"   <td>4.000000</td>\n",
"   <td>279.000000</td>\n",
"   <td>17.400000</td>\n",
"   <td>375.377500</td>\n",
"   <td>6.950000</td>\n",
"   <td>17.025000</td>\n",
" </tr>\n",
" <tr>\n",
"   <th>50%</th>\n",
"   <td>0.256510</td>\n",
"   <td>0.000000</td>\n",
"   <td>9.690000</td>\n",
"   <td>0.000000</td>\n",
"   <td>0.538000</td>\n",
"   <td>6.208000</td>\n",
"   <td>77.500000</td>\n",
"   <td>3.207450</td>\n",
"   <td>5.000000</td>\n",
"   <td>330.000000</td>\n",
"   <td>19.050000</td>\n",
"   <td>391.440000</td>\n",
"   <td>11.360000</td>\n",
"   <td>21.200000</td>\n",
" </tr>\n",
" <tr>\n",
"   <th>75%</th>\n",
"   <td>3.677082</td>\n",
"   <td>12.500000</td>\n",
"   <td>18.100000</td>\n",
"   <td>0.000000</td>\n",
"   <td>0.624000</td>\n",
"   <td>6.625000</td>\n",
"   <td>94.075000</td>\n",
"   <td>5.188425</td>\n",
"   <td>24.000000</td>\n",
"   <td>666.000000</td>\n",
"   <td>20.200000</td>\n",
"   <td>396.225000</td>\n",
"   <td>16.955000</td>\n",

```

```

"    <td>25.000000</td>\n",
"  </tr>\n",
" <tr>\n",
"   <th>max</th>\n",
"   <td>88.976200</td>\n",
"   <td>100.000000</td>\n",
"   <td>27.740000</td>\n",
"   <td>1.000000</td>\n",
"   <td>0.871000</td>\n",
"   <td>8.780000</td>\n",
"   <td>100.000000</td>\n",
"   <td>12.126500</td>\n",
"   <td>24.000000</td>\n",
"   <td>711.000000</td>\n",
"   <td>22.000000</td>\n",
"   <td>396.900000</td>\n",
"   <td>37.970000</td>\n",
"   <td>50.000000</td>\n",
" </tr>\n",
" </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"      CRIM      ZN      INDUS      CHAS      NOX      RM  \\n",
"count 506.000000 506.000000 506.000000 506.000000 506.000000 501.000000
\n",
"mean   3.613524 11.363636 11.136779 0.069170 0.554695 6.284341 \n",
"std    8.601545 23.322453 6.860353 0.253994 0.115878 0.705587 \n",
"min    0.006320 0.000000 0.460000 0.000000 0.385000 3.561000 \n",
"25%    0.082045 0.000000 5.190000 0.000000 0.449000 5.884000 \n",
"50%    0.256510 0.000000 9.690000 0.000000 0.538000 6.208000 \n",
"75%    3.677082 12.500000 18.100000 0.000000 0.624000 6.625000 \n",
"max    88.976200 100.000000 27.740000 1.000000 0.871000 8.780000 \n",
"\n",
"      AGE      DIS      RAD      TAX      PTRATIO      B  \\n",
"count 506.000000 506.000000 506.000000 506.000000 506.000000 506.000000
\n",
"mean   68.574901 3.795043 9.549407 408.237154 18.455534 356.674032
\n",
"std    28.148861 2.105710 8.707259 168.537116 2.164946 91.294864 \n",
"min    2.900000 1.129600 1.000000 187.000000 12.600000 0.320000 \n",
"25%    45.025000 2.100175 4.000000 279.000000 17.400000 375.377500
\n",
"50%    77.500000 3.207450 5.000000 330.000000 19.050000 391.440000
\n",
"75%    94.075000 5.188425 24.000000 666.000000 20.200000 396.225000
\n",
"max    100.000000 12.126500 24.000000 711.000000 22.000000 396.900000
\n",
"\n",

```

```

"      LSTAT      MEDV \n",
"count 506.000000 506.000000 \n",
"mean  12.653063 22.532806 \n",
"std   7.141062  9.197104 \n",
"min   1.730000  5.000000 \n",
"25%   6.950000 17.025000 \n",
"50%   11.360000 21.200000 \n",
"75%   16.955000 25.000000 \n",
"max   37.970000 50.000000 "
]
},
"execution_count": 6,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"housing.describe()"
]
},
{
"cell_type": "code",
"execution_count": 7,
"metadata": {},
"outputs": [],
"source": [
"%matplotlib inline"
]
},
{
"cell_type": "code",
"execution_count": 8,
"metadata": {},
"outputs": [],
"source": [
"# # For plotting histogram\n",
"# import matplotlib.pyplot as plt\n",
"# housing.hist(bins=50, figsize=(20, 15))"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
"### Train-Test Splitting"
]
},
{
"cell_type": "code",
"execution_count": 9,
"metadata": {},

```

```

"outputs": [],
"source": [
    "# For learning purpose\n",
    "import numpy as np\n",
    "def split_train_test(data, test_ratio):\n",
    "    np.random.seed(42)\n",
    "    shuffled = np.random.permutation(len(data))\n",
    "    print(shuffled)\n",
    "    test_set_size = int(len(data) * test_ratio)\n",
    "    test_indices = shuffled[:test_set_size]\n",
    "    train_indices = shuffled[test_set_size:] \n",
    "    return data.iloc[train_indices], data.iloc[test_indices]"
]
},
{
    "cell_type": "code",
    "execution_count": 10,
    "metadata": {},
    "outputs": [],
    "source": [
        "# train_set, test_set = split_train_test(housing, 0.2)"
    ]
},
{
    "cell_type": "code",
    "execution_count": 11,
    "metadata": {},
    "outputs": [],
    "source": [
        "# print(f"Rows in train set: {len(train_set)}\n\nRows in test set: {len(test_set)}\n\n")"
    ]
},
{
    "cell_type": "code",
    "execution_count": 12,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "Rows in train set: 404\n",
                "Rows in test set: 102\n",
                "\n"
            ]
        }
    ],
    "source": [
        "from sklearn.model_selection import train_test_split\n",
        "train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)\n",
        "print(f"Rows in train set: {len(train_set)}\n\nRows in test set: {len(test_set)}\n\n")"
    ]

```

```

]
},
{
  "cell_type": "code",
  "execution_count": 13,
  "metadata": {},
  "outputs": [],
  "source": [
    "from sklearn.model_selection import StratifiedShuffleSplit\n",
    "split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)\n",
    "for train_index, test_index in split.split(housing, housing['CHAS']):\n",
    "    strat_train_set = housing.loc[train_index]\n",
    "    strat_test_set = housing.loc[test_index]"
  ]
},
{
  "cell_type": "code",
  "execution_count": 14,
  "metadata": {},
  "outputs": [
    {
      "data": {
        "text/plain": [
          "0    95\n",
          "1     7\n",
          "Name: CHAS, dtype: int64"
        ]
      },
      "execution_count": 14,
      "metadata": {},
      "output_type": "execute_result"
    }
  ],
  "source": [
    "strat_test_set['CHAS'].value_counts()"
  ]
},
{
  "cell_type": "code",
  "execution_count": 15,
  "metadata": {},
  "outputs": [
    {
      "data": {
        "text/plain": [
          "0   376\n",
          "1    28\n",
          "Name: CHAS, dtype: int64"
        ]
      },
      "execution_count": 15,

```

```

    "metadata": {},
    "output_type": "execute_result"
  },
  "source": [
    "strat_train_set['CHAS'].value_counts()"
  ],
  {
    "cell_type": "code",
    "execution_count": 16,
    "metadata": {},
    "outputs": [],
    "source": [
      "# 95/7"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 17,
    "metadata": {},
    "outputs": [],
    "source": [
      "# 376/28"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 18,
    "metadata": {},
    "outputs": [],
    "source": [
      "housing = strat_train_set.copy()"
    ]
  },
  {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
      "## Looking for Correlations"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 19,
    "metadata": {},
    "outputs": [
      {
        "data": {
          "text/plain": [
            "MEDV      1.000000\n",

```

```
"RM      0.680857\n",  
"B       0.361761\n",  
"ZN      0.339741\n",  
"DIS     0.240451\n",  
"CHAS    0.205066\n",  
"AGE     -0.364596\n",  
"RAD     -0.374693\n",  
"CRIM    -0.393715\n",  
"NOX     -0.422873\n",  
"TAX     -0.456657\n",  
"INDUS   -0.473516\n",  
"PTRATIO -0.493534\n",  
"LSTAT   -0.740494\n",  
"Name: MEDV, dtype: float64"  
]  
},  
"execution_count": 19,  
"metadata": {},  
"output_type": "execute_result"
```



## 7. TESTING

### 7.1. Unit Testing

Let's explore some practical ways to apply testing to your project:

#### 1. Unit Tests:

- **What they are:** Unit tests focus on individual functions or methods within your code. They verify that each component works as expected in isolation.
- **Example:** If you have a function `calculate_area()` that calculates the area of a rectangle, you'd write unit tests to check its behavior with different input values (e.g., positive numbers, negative numbers, zero).

#### 2. Integration Tests:

- **What they are:** Integration tests check how different parts of your code interact with each other. They ensure that components work together as intended.
- **Example:** If you have a user interface (UI) and a backend API, integration tests could simulate user interactions and verify that the UI correctly communicates with the API.

#### 3. End-to-End Tests:

- **What they are:** End-to-end tests simulate real-world user scenarios. They test the entire system from start to finish to ensure it works as expected.
- **Example:** If you're building a web application, you could write end-to-end tests to simulate user login, navigation, and data submission.

#### 4. Regression Tests:

- **What they are:** Regression tests are run after making changes to your code to ensure that existing functionality still works correctly. They help prevent unintended side effects of modifications.

### Train-Test Splitting

```
# For learning purpose
import numpy as np
def split_train_test(data, test_ratio):
    np.random.seed(42)
    shuffled = np.random.permutation(len(data))
    print(shuffled)
    test_set_size = int(len(data) * test_ratio)
    test_indices = shuffled[:test_set_size]
    train_indices = shuffled[test_set_size:]
    return data.iloc[train_indices], data.iloc[test_indices]
```

```
# train_set, test_set = split_train_test(housing, 0.2)
```

Pyth

Pyth

## 8. LIMITATION AND FUTURE SCOPE

While the **E-Real Estate Price Prediction** project offers valuable insights, there are certain limitations that can affect its accuracy, usability, and overall performance:

### 1. Data Quality and Availability:

- The model heavily depends on the quality and completeness of the data. Missing, outdated, or incorrect data can lead to inaccurate predictions. Furthermore, obtaining high-quality real estate data, especially for different regions, can be challenging.
- If certain regions or types of properties are underrepresented in the dataset, the model may not generalize well to those properties.

### 2. Feature Limitations:

- The model uses a limited set of features (e.g., size, location, number of rooms, age, and amenities). There are many other factors influencing property prices, such as economic conditions, market demand, crime rates, schools, and transport availability, which are not considered in the current model.
- The inclusion of additional features might improve the model's predictive power but would require more complex data gathering and preprocessing.

### 3. Model Simplicity:

- The current project uses a **Linear Regression** model, which, while simple and interpretable, may not capture the complexity of real estate price dynamics. More advanced models, such as **Random Forests**, **Gradient Boosting Machines (GBM)**, or **Deep Learning**, may provide better performance but require more computational resources and a larger dataset.
- The model may also fail to account for non-linear relationships between variables (e.g., property prices in specific locations) and interactions between features.

### 4. Geographical Limitations:

- The current model may be trained using data from specific regions or cities. Real estate markets can vary significantly across different geographies, so the model may not work well if applied to new areas without retraining on local data.
- Globalization and regional factors like political instability, economic recessions, or changes in the local economy are factors that might not be reflected in the data and could impact the model's accuracy.

### 5. Seasonality and Temporal Trends:

- Real estate prices are often affected by seasonal trends (e.g., summer and winter markets) and temporal factors such as inflation or interest rates. If the model does not account for these factors, it may not predict prices accurately at certain times of the year or during economic changes.

### 6. User Input Variability:

- In real-world scenarios, the accuracy of the predictions could be affected by the quality of user input. For example, incorrect or incomplete property details provided by users could lead to inaccurate price predictions.

### 7. Overfitting and Model Interpretability:

- Overfitting can occur if the model is too complex for the dataset, resulting in a model that performs well on the training data but poorly on unseen data.
- While simple models like linear regression offer interpretability, more complex models like deep learning may become a "black box," making it difficult for users to understand why a specific prediction was made.

## Future Scope

The **E-Real Estate Price Prediction** project has significant potential for further development and improvement. Here are some areas where the project can be expanded and enhanced:

### 1. Incorporating More Features:

- Adding more influential features, such as **economic indicators**, **local infrastructure**, **crime rates**, **school rankings**, **neighborhood trends**, and **historical sales data**, can improve the accuracy of the predictions.
- **Geospatial features** (e.g., distance to amenities like shopping centers, parks, or public transport) and **weather patterns** could also play an important role in pricing predictions.

### 2. Use of Advanced Machine Learning Models:

- Transitioning to more sophisticated models like **Random Forest**, **XGBoost**, or **Gradient Boosting** would likely improve prediction accuracy. These models are better at capturing complex non-linear relationships between features.
- **Deep Learning** methods, such as **Neural Networks** or **Convolutional Neural Networks (CNNs)** (for analyzing image-based data, such as property photos), can also be explored to capture more nuanced patterns.
- **Ensemble models** could be used to combine the predictions of several models to increase overall accuracy and robustness.

### 3. Real-Time Data Integration:

- Integrating real-time data from real estate listings, economic indicators, or news sources (such as changes in property taxes or new developments in a city) could allow the system to provide up-to-date price predictions.
- APIs from real estate platforms, such as Zillow, Redfin, or Realtor.com, could be used to gather real-time property listings and augment the data used by the model.

### 4. Geographical Flexibility:

- To improve the model's ability to predict prices in different regions, we can create region-specific models or build a more generalized model that includes region-specific data and geographical features.
- Geospatial analysis tools (such as **GIS mapping**) could help improve predictions by considering proximity to amenities and other geographical features.


### 5. Time Series Analysis:

- Incorporating **time series analysis** techniques would allow the model to capture trends and seasonality in property prices over time. This could make the predictions more accurate by considering price fluctuations due to temporal factors (e.g., market cycles, economic conditions, etc.).
- Techniques such as **ARIMA** (Auto-Regressive Integrated Moving Average) or **LSTM** (Long Short-Term Memory) networks could be used to predict future prices based on historical trends.

### 6. User Interface and User Experience (UI/UX):

- Improving the user interface (UI) can make the system more user-friendly and accessible. For example, adding features like interactive graphs, property comparison tools, or a map-based search for properties can enhance the user experience.
- A **mobile application** could be developed for real-time predictions and updates on property prices.

## 9. REFERENCES

-  **Scikit-Learn Documentation**  
<https://scikit-learn.org/stable/>
-  **Pandas Documentation**  
<https://pandas.pydata.org/docs/>
-  **NumPy Documentation**  
<https://numpy.org/doc/>
-  **Matplotlib Documentation**  
<https://matplotlib.org/stable/contents.html>
-  **Seaborn Documentation**  
<https://seaborn.pydata.org/>