A globe with a red point

AI-generated content may be incorrect. A black rectangular object with a black border

AI-generated content may be incorrect.

**Real Estate recommendation system**

The Graduation Project Submitted to

The Faculty of Computers and Artificial Intelligence,

Cairo University

In Partial Fulfillment of the Requirements

for the bachelor’s degree

In

**Operations Research and Decision Support**

**Esraa Ahmed Saad Mobarak 20210062**

**Fatma Abd El-Monam Gad 20210285**

**Mawada Ebrahim Ali 20210415**

**Mostafa Abd El-Wahab Abbas 20210392**

**Seif Maher Hamdy 20210171**

**Yasseen Emad El-Deen Saber 20200635**

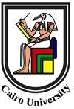
Under Supervision of:

DR. Aymen Sabry Ghoiem

TA.Youssef

JULY.2025

*Real Estate recommendation system Using ML Algorithms*



**Real Estate Recommendation System**

**Esraa**

**Fatma**

**Mawada**

**Mostafa**

**Seif**

**Yaseen**

Supervised By

**Dr. Ayman Ghoneim**

**TA. Youssef**

CAIRO UNIVERSITY

JULY 2025

## Abstract

The real estate market in Egypt presents diverse opportunities and challenges for both buyers and sellers, making intelligent recommendation systems increasingly valuable. This project aims to assist users in identifying suitable properties and predicting their market value by applying machine learning techniques to real estate data collected from Egyptian listings.

The system is based on a content-based recommendation approach using the k-Nearest Neighbors (KNN) algorithm. KNN was selected due to its simplicity, effectiveness, and strong performance on medium-sized datasets with mixed-type features. The model analyzes user preferences, such as location, price range, number of rooms, and furnishing status, to suggest similar property listings. Additionally, the model estimates property prices by learning from historical data patterns.

Data preprocessing was a crucial step, involving the encoding of categorical variables, normalization of numerical features, and handling of missing or inconsistent values. Although working with real estate data posed challenges like regional variability and feature sparsity, the use of KNN provided interpretable and accurate results. The model successfully generates personalized property suggestions and reliable price predictions, showing its potential to support decision-making in Egypt’s dynamic real estate sector.

.

## DECLARATION

We hereby declare that our dissertation is entirely our work and genuine / original. We understand that in case of discovery of any PLAGIARISM at any stage, our group will be assigned an F (FAIL) grade and it may result in withdrawal of our Bachelor’s degree.

Group members:

**Name Signature**

Esraa Ahmed Saad Mobarak

Fatma Abd El-Monem Gad

Mawada Ebrahim Ali

Mostafa Abd El-Wahab Abbas

Seif Maher Hamdy

Yaseen Emad El-Deen Saber

## PLAIGRISM CERTIFICATE

This is to certify that the project entitled “ ”, which is being submitted

here with for the award of the “**Bachelor of Computer and Artificial Intelligence Degree” in “Operations Research and Decision Support** ”. This is the result of the original work by **Student 1** and **Student 2** under my supervision and guidance. The work embodied in this project has not been done earlier for the basis of award of any degree or compatible certificate or similar tile of this for any other diploma/examining body or university to the best of my knowledge and belief.

**Turnitin Originality Report** Processed on 31-May-2017 00:14 PKT ID: 300502964

Word Count: 12948

|  |  |  |
| --- | --- | --- |
| Similarity Index |  |  |
| 10% |  |
| Similarity by Source |  |
| Internet Sources: | 06% |
| Publications: | 0 % |  |
| Student Papers: | 08% |  |
| Date: 10/07/2025 |  | Supervisor Name (Supervisor) |

**TABLE OF CONTENTS**

Table of Contents

[Abstract 3](#_Toc202994625)

[DECLARATION 4](#_Toc202994626)

[PLAIGRISM CERTIFICATE 5](#_Toc202994627)

[CHAPTER 1 10](#_Toc202994628)

[INTRODUCTION TO PROJET 10](#_Toc202994629)

[1.0 Introduction 10](#_Toc202994630)

[1.1 Problem Domain 10](#_Toc202994631)

[1.2 Problem Statement 11](#_Toc202994632)

[1.3 Proposed system 11](#_Toc202994633)

[1.3.1 Objectives 11](#_Toc202994634)

[1.3.2 Proposed system features 12](#_Toc202994635)

[1.4 Development Methodology 12](#_Toc202994636)

[1.5 Resource Requirements 13](#_Toc202994637)

[Subsystem Requirements 13](#_Toc202994638)

[Data Requirements 13](#_Toc202994639)

[Localization Requirements 13](#_Toc202994640)

[1.6 Report Layout 14](#_Toc202994641)

[CHAPTER 2 15](#_Toc202994642)

[2.0 Introduction 15](#_Toc202994643)

[2.1 Project Overview 15](#_Toc202994644)

[1. Property Recommendation System 16](#_Toc202994645)

[User Input Simulation: 16](#_Toc202994646)

[Similarity Matching: 16](#_Toc202994647)

[2. Property Price Prediction System 16](#_Toc202994648)

[3. Project Impact 16](#_Toc202994649)

[**Data Quality and User Input** 17](#_Toc202994650)

[**Encoding and Feature Limitations** 17](#_Toc202994651)

[**Fuzzy Matching Ambiguity** 17](#_Toc202994652)

[**Static Nature of Data** 17](#_Toc202994653)

[**Limited External Integration** 17](#_Toc202994654)

[**Scalability for Real-Time Recommendations** 17](#_Toc202994655)

[2.2.1 Innovations in the Project 18](#_Toc202994656)

[**ML-Powered Property Matching** 18](#_Toc202994657)

[**Price Prediction via Regression Models** 18](#_Toc202994658)

[**Flexible Search with Fuzzy Matching** 18](#_Toc202994659)

[**Dynamic Filtering Logic** 18](#_Toc202994660)

[**Content-Based Filtering Approach** 18](#_Toc202994661)

[**Problem-Solving Oriented Code Design** 18](#_Toc202994662)

[2.2.2 Design of Project 19](#_Toc202994663)

[**1. Search Engine and Fuzzy Filter Module** 19](#_Toc202994664)

[**2. Property Recommendation Engine (KNN)** 19](#_Toc202994665)

[**3. Price Prediction Engine** 19](#_Toc202994666)

[**4. User Flow Overview** 19](#_Toc202994667)

[**5. System Structure (Backend-Only ML)** 19](#_Toc202994668)

[3. Related Work 20](#_Toc202994669)

[3.1 Comparing models: 21](#_Toc202994670)

[CHAPTER 3 22](#_Toc202994671)

[Requirements and Work Details 22](#_Toc202994672)

[3.1 Project Stakeholders 22](#_Toc202994673)

[3.2 Use Case Requirements 22](#_Toc202994674)

[3.2.2 User System Requirements 24](#_Toc202994675)

[Prediction Pipeline 24](#_Toc202994676)

[Recommendation System 24](#_Toc202994677)

[Model Comparison Framework 24](#_Toc202994678)

[3.3 Project Non-Functional Requirements 24](#_Toc202994679)

[3.3.2 Evaluation Qualities 25](#_Toc202994680)

[3.4 Alternative Flow 26](#_Toc202994681)

[CHAPTER 4 27](#_Toc202994682)

[Data 27](#_Toc202994683)

[**4.1 Features Description** 27](#_Toc202994684)

[**4.2 Data Cleaning and Preprocessing** 28](#_Toc202994685)

[**4.3 Data Analysis** 29](#_Toc202994686)

[**4.4 Summary of Insights** 33](#_Toc202994687)

[CHAPTER 5 34](#_Toc202994688)

[Modeling and design 34](#_Toc202994689)

[**5.0 Introduction** 34](#_Toc202994690)

[**5.1 Our Models** 34](#_Toc202994691)

[**5.1.1 K-Nearest Neighbors (KNN)** 34](#_Toc202994692)

[**5.1.2 Filtering and Fuzzy Search** 36](#_Toc202994693)

[**A. Dynamic Filtering (Rule-Based Search)** 36](#_Toc202994694)

[**B. Fuzzy Filtering (via RapidFuzz)** 37](#_Toc202994695)

[**5.1.3 Ensemble Regression Models** 38](#_Toc202994696)

[**A. Bagging Regressor** 38](#_Toc202994697)

[**B. Histogram-Based Gradient Boosting (HistGradientBoosting)** 39](#_Toc202994698)

[**5.2 Required Technologies and Libraries** 40](#_Toc202994699)

[CHAPTER 6 41](#_Toc202994700)

[CONCLUSION AND FUTUURE WORK 41](#_Toc202994701)

[**6.1 Project Conclusion** 41](#_Toc202994702)

[**6.2 Project Limitations** 41](#_Toc202994703)

[**6.3 Future Work** 42](#_Toc202994704)

[**References** 43](#_Toc202994705)

[APPENDICES 44](#_Toc202994706)

[**Appendix i: Terminologies and Definitions** 44](#_Toc202994707)

[**Appendix ii: Sample Recommendation Output** 44](#_Toc202994708)

[**Appendix iii: Source Code Summary** 45](#_Toc202994709)

[**Appendix iv: User Manual** 45](#_Toc202994710)

**LIST OF FIGURES**

[Figure 1 Project Timeline 14](#_Toc202994419)

[Figure 2 Project Logo 15](#_Toc202994420)

[Figure 3 Use Case Diagram 22](#_Toc202994421)

[Figure 4 Project Flochart Diagram 26](#_Toc202994422)

[Figure 5 Sample Raw Data Snapshot 27](#_Toc202994423)

[Figure 6 Missing Values Summary Before Cleaning 28](#_Toc202994424)

[Figure 7 Data Info and Types After Preprocessing 29](#_Toc202994425)

[Figure 8 Price Distribution Plot 29](#_Toc202994426)

[Figure 9 Property Type Frequency 30](#_Toc202994427)

[Figure 10 Median Price by Property Type 30](#_Toc202994428)

[Figure 11 City-Wise Property Count 31](#_Toc202994429)

[Figure 12 Property Location Map 31](#_Toc202994430)

[Figure 13 Furnishing Status Distribution 32](#_Toc202994431)

[Figure 14 Completion Status Distribution 32](#_Toc202994432)

[Figure 15 Correlation Heatmap of Numerical Features 33](#_Toc202994433)

# CHAPTER 1

# INTRODUCTION TO PROJET

## 1.0 Introduction

The real estate sector in Egypt has witnessed continuous growth, with increasing demand for intelligent tools that can assist both buyers and sellers in making informed decisions. With a wide range of property options differing by location, price, type, and features, navigating the market has become complex for individual users. This project aims to address this challenge by developing a real estate recommendation system that not only suggests suitable properties based on user preferences but also predicts the expected price of real estate listings.

The system applies a **content-based machine learning approach**, using user-defined criteria such as city, district, number of bedrooms, property type, and furnishing status to generate personalized recommendations. The **k-Nearest Neighbors (KNN)** algorithm was chosen for this task due to its simplicity, interpretability, and effectiveness in finding similar items within a dataset that includes both numerical and categorical features.

In addition to the recommendation task, the project incorporates **price prediction** models that estimate property values. This was achieved by applying **regression algorithms** Decision Tree Regression and enhancing their accuracy through **ensemble techniques** like Bagging. These models help users understand the fair market price of a property and reduce the risk of overpaying or underselling.

The goal of this work is to provide a data-driven solution that combines both recommendation and prediction in a unified system. By leveraging historical property data and machine learning techniques, the system can support smarter, faster, and more confident real estate decisions tailored to the Egyptian market.

### 1.1 Problem Domain

Building a personalized real estate recommendation and price prediction system involves addressing several critical challenges. The system must process a wide range of user preferences, such as preferred location, property type, price range, and other features like size, number of rooms, and furnishing status. At the same time, it needs access to a comprehensive, up-to-date property dataset that accurately reflects the Egyptian real estate market.

The core objective is to provide property recommendations that align with individual needs and market behavior, while also predicting property prices based on historical data. This requires advanced machine learning algorithms capable of interpreting both numerical and categorical data, especially in a domain where location-based trends and economic factors can heavily influence property value.

User interaction also plays an important role. Although the current system focuses on the backend models rather than user interfaces, future extensions must ensure that input from users can be effectively used to generate accurate, relevant recommendations. A key technical challenge is handling mixed-type data efficiently and producing reliable outputs, even with limited or incomplete user inputs.

Another challenge lies in the **price prediction task**, which demands robust regression models. These models must consider multiple influencing factors such as city, district, amenities, and offering type. To enhance accuracy, ensemble techniques like Random Forest or Gradient Boosting can be used to reduce overfitting and improve generalization.

Finally, ensuring that the system is adaptable to changing market dynamics is essential. The model should be regularly updated with fresh data to reflect current trends and prices. Successfully addressing these challenges will result in a powerful recommendation and pricing system that helps users make better real estate decisions in Egypt’s dynamic and diverse property market.

### 1.2 Problem Statement

Navigating the real estate market in Egypt can be overwhelming for both buyers and sellers. With a vast number of property listings differing in price, location, size, and features, individuals often face difficulties in identifying properties that align with their personal needs and budget. Traditional property search methods are usually time-consuming, inefficient, and lack the level of personalization required to make confident real estate decisions.

There is an increasing need for a system that can offer **personalized property recommendations** and accurate **price predictions**, helping users find the right property while understanding its true market value. Such a system must be capable of analyzing diverse user preferences and large datasets to deliver relevant suggestions and price estimations.

The core challenge is to develop a machine learning–based system that effectively connects users to properties that match their needs and predicts prices based on historical and market data. This involves:

* Understanding user preferences such as city, district, price range, property type, and amenities.
* Handling and analyzing real estate data with both numerical (e.g., size, price) and categorical (e.g., location, furnishing) features.
* Applying regression and ensemble models to generate reliable and accurate property price predictions.
* Ensuring the system adapts and improves as more data becomes available.

### 1.3 Proposed system

Before designing our real estate recommendation and price prediction system, we studied several existing platforms such as **Property Finder**, **OLX Egypt**, and **Bayut**. While these platforms provide basic search and filter options, they often lack intelligent personalization and predictive capabilities. Our goal was to overcome these limitations by introducing a system that offers more accurate, tailored recommendations and price insights based on user preferences and real market trends.

### 1.3.1 Objectives

The primary objective of this project is to develop a **personalized real estate recommendation and pricing system** using a **content-based machine learning approach**. The system aims to:

1. **Analyze and understand individual user preferences**, including location, budget, property type, number of bedrooms, and furnishing status.
2. **Utilize a rich real estate dataset** that includes both categorical and numerical features to provide relevant property suggestions.
3. **Apply the K-Nearest Neighbors (KNN) algorithm** to generate property recommendations based on content similarity.
4. **Implement regression and ensemble techniques** (e.g., Random Forest, Gradient Boosting) to accurately predict property prices based on historical and regional data.
5. **Ensure model scalability and efficiency**, making it adaptable to larger datasets and different market conditions in Egypt.

### 1.3.2 Proposed system features

* **Personalized Property Recommendations**: Suggest properties tailored to user preferences, including city, district, price range, number of bedrooms and bathrooms, property type, furnishing status, and other relevant features.
* **Price Prediction**: Estimate the fair market value of a property using regression and ensemble models based on location, size, condition, offering type, and other property attributes.
* **Preference-Based Filtering**: Allow users to input specific property needs (e.g., furnished/unfurnished, number of rooms, offering type) to refine and personalize the recommendation results.
* **Neighborhood and Location Analysis**: Use location-based data (such as city, town, or compound) to recommend properties in areas with similar features or price trends.
* **User Profile Adaptation**: Adapt recommendations over time as user preferences become more refined or new data becomes available, improving system responsiveness and personalization.
* **Predictive Insight**: Offer users market insight by showing how predicted prices compare to current listing prices, helping them evaluate affordability and market fairness.

## 1.4 Development Methodology

The development of the real estate recommendation and price prediction system follows an **Agile-inspired approach**, focused on flexibility, iterative experimentation, and continuous improvement of the underlying models. Since this project is centered on **machine learning algorithms and data analysis only**, without frontend or backend application development, the methodology emphasizes **data preparation, model selection, training, and evaluation**. The development process includes the following key phases:

**1. Requirements Gathering and Data Collection**

A comprehensive real estate dataset relevant to the Egyptian market was collected from online sources. The data includes attributes such as property type, price, location, size, furnishing, and offering type. Data cleaning was performed to handle missing, inconsistent, and duplicated entries. Categorical variables were encoded, and numerical values were scaled to prepare the dataset for modeling.

**2. Data Analysis and Feature Engineering**

The dataset was thoroughly analyzed to understand correlations between features and to identify key variables affecting both recommendations and price prediction. Feature engineering was performed to improve model performance, such as converting geographic data into meaningful clusters or zones and extracting new attributes like price per square meter.

**3. Model Design and Selection**

A **content-based recommendation model** was designed using the **k-Nearest Neighbors (KNN)** algorithm to identify similar properties based on user-defined preferences. For the **price prediction task**, regression models such as **Gradient Boosting and Random Forest** were applied and compared to find the most accurate and reliable predictor.

**4. Model Training and Evaluation**

The models were trained using the prepared dataset and evaluated using appropriate metrics such as Mean Absolute Error (MAE) for price prediction and precision-based measures for recommendation accuracy. Hyperparameter tuning and cross-validation were used to improve the performance and robustness of each model.

**5. System Validation and Refinement**

The system was tested using real-world queries and user scenarios to ensure that property recommendations were relevant and price predictions aligned with actual market values. Based on evaluation results, further refinements were made to the preprocessing pipeline and model parameters.

## 1.5 Resource Requirements

* Google Colab: Used as the primary platform for coding, model training, and data analysis. Colab provides a free cloud-based environment with pre-installed libraries, supports GPU acceleration, and enables collaboration and easy integration with Google Drive.

### Subsystem Requirements

* Python Programming Language: Core language used for all data preprocessing, machine learning model development, and analysis.
* Scikit-learn: Essential machine learning library used to implement KNN, regression, and ensemble models.
* Pandas & NumPy: Libraries for data manipulation, transformation, and handling of structured data.
* Matplotlib & Seaborn: Used to visualize feature distributions, correlations, and model outputs.

### Data Requirements

* Real Estate Dataset: Includes property listings from the Egyptian market, with features such as price, location (city, district, compound), size, number of rooms, furnishing status, and offering type.
* User Preference Data: Inputs such as desired city, price range, bedrooms, and property type used to generate personalized recommendations.
* and Preprocessing Tools: Techniques and scripts used to handle missing values, encode categorical data, normalize numerical features, and ensure dataset quality.

### Localization Requirements

Although the system is designed for the Egyptian real estate market, it can be adapted to other countries or regions by incorporating localized data. The system structure allows easy integration of regional property features, pricing behaviors, and terminology, ensuring broader applicability in the future.

## 1.6 Report Layout

Figure Project Timeline

# CHAPTER 2

**BACKGROUND/EXISTING WORK**

## 2.0 Introduction

In recent years, the integration of machine learning into the real estate domain has gained significant attention. With increasing access to large volumes of property data, researchers and developers have explored intelligent systems that can support users in selecting properties and making investment decisions. These systems often aim to enhance user experience by offering personalized recommendations and accurate price estimations based on historical trends and individual preferences.

This chapter presents an overview of existing work related to real estate recommendation systems and property price prediction. It highlights key methodologies, commonly used algorithms, and practical implementations in real-world applications. By examining current approaches and their limitations, this chapter provides the foundation for understanding how the proposed system aims to offer improvements in personalization, prediction accuracy, and relevance—particularly within the context of the Egyptian real estate market.

### 2.1 Project Overview

**

Figure Project Logo

The system developed serves two primary functions:

1. Recommending properties based on user-defined preferences using a content-based filtering approach.
2. Predicting property prices through regression and ensemble models based on detailed real estate features.

Unlike a full web or mobile application, this system functions purely as a **data-driven backend module**, focusing on **machine learning algorithms** for recommendation and price prediction. It utilizes real property listings from the Egyptian real estate market.

### ****1. Property Recommendation System****

This module allows users to input their preferred property characteristics, and the system returns a **ranked list of the top five properties** that are most similar to the input query. The similarity is determined using the **k-Nearest Neighbors (KNN)** algorithm applied to both categorical and numerical features.

### ****User Input Simulation****:

Users (or test cases) provide details such as:

* City, town, or district
* Property type
* Number of bedrooms and bathrooms
* Size and furnishing status
* Offering type and price range

### ****Similarity Matching****:

The KNN algorithm identifies properties most similar to the input profile by calculating feature-based distances. The system ranks and returns the **top five most relevant properties**, allowing users to explore similar options based on their preferences.

## ****2. Property Price Prediction System****

In addition to recommending similar properties, the system can predict the likely **market value** of a given property using **regression models**.

**Feature-Based Price Estimation**:  
Based on the input features (such as location, size, furnishing, completion status, etc.), the system uses machine learning models—**Linear Regression**, **Random Forest**, and **Gradient Boosting**—to estimate the expected price.

**Predictive Analysis Integration**:  
This module helps users (or potential buyers/sellers) understand whether a given price is fair based on historical data and similar listings in the same region.

## ****3. Project Impact****

**From a user or market perspective**, the system offers several key benefits in the Egyptian real estate context:

* **Personalized Recommendations**: Users receive a ranked list of properties matching their specific needs, improving the property search experience.
* **Data-Driven Pricing Insights**: Sellers can set more competitive prices, and buyers can avoid overpriced listings.
* **Flexibility and Adaptability**: New user preferences or market trends can be incorporated without major changes to the system.
* **Scalability**: The system can be expanded to include more advanced features like clustering or area-based filtering, or to cover new regions in the future.

**2.2 Project Limitations**

Despite the capabilities of our real estate recommendation and price prediction system, several limitations should be acknowledged:

**Data Quality and User Input**

* The accuracy of recommendations is highly dependent on the availability and precision of real-world real estate listings. If the user-provided filtering criteria (e.g., misspelled town names or missing features) are inaccurate or vague, the resulting recommendations may not be optimal.

**Encoding and Feature Limitations**

* While our model leverages categorical and numerical property features, the system may not account for abstract qualitative factors (like building aesthetics, view, or neighborhood safety) that influence buyer decisions but are hard to quantify.

**Fuzzy Matching Ambiguity**

* Although fuzzy search improves usability by correcting input errors (e.g., "Doky" → "Dokki"), it may occasionally match to unintended locations if multiple similar town names exist, especially under a low similarity threshold.

**Static Nature of Data**

* The dataset is not live or automatically updated. Since it is based on static historical Egyptian real estate data, newer listings or market shifts may not be reflected unless the dataset is manually updated.

**Limited External Integration**

* The system currently operates in isolation without integrating real-time property platforms, maps, or financial tools (e.g., mortgage calculators), which could enhance usability in a real deployment.

**Scalability for Real-Time Recommendations**

* The current system is designed for demonstration and experimentation. If deployed with very large datasets or concurrent users, performance bottlenecks may arise, especially with high-dimensional encoding and KNN lookups.

### 2.2.1 Innovations in the Project

Despite the above limitations, this project introduces several innovative features and technical strengths:

**ML-Powered Property Matching**

* The system uses the **K-Nearest Neighbors (KNN)** algorithm with cosine similarity to recommend properties most similar to a user's selected or described property. This enables personalized property discovery based on spatial, structural, and financial features.

**Price Prediction via Regression Models**

* The system integrates **regression and ensemble learning models** (e.g., Random Forest, Linear Regression) to estimate property prices based on historical listings. This helps users evaluate property affordability or detect overpricing.

**Flexible Search with Fuzzy Matching**

* The project incorporates **fuzzy filtering** using the RapidFuzz library, allowing users to search properties even if input values (like city or town names) are slightly misspelled or incomplete.

**Dynamic Filtering Logic**

* Users can apply **multi-criteria filters**—including city, town, compound, price range, bedrooms, and bathrooms—with the option to skip any field, enabling a flexible and inclusive search process.

**Content-Based Filtering Approach**

* Rather than relying on collaborative filtering, the system applies a **content-based** approach, using each property’s encoded feature vector to generate relevant recommendations.

**Problem-Solving Oriented Code Design**

* The entire system is built as a modular, **algorithm-first backend**, focused on data science and problem solving. While there is no UI or API in the current version, the logic is fully integrable with Streamlit or FastAPI if needed.

### 2.2.2 Design of Project

Our project is designed as a **modular machine learning system** built to operate directly in notebooks or scripts. It follows a clean process from user input to filtered recommendations and predictive analytics.

**1. Search Engine and Fuzzy Filter Module**

* Users input criteria such as city, town, compound, price range, number of bedrooms, and number of bathrooms.
* Fuzzy matching ensures minor spelling mistakes are corrected (e.g., “6 Octobar” → “6th of October”).
* All inputs are optional, allowing personalized and flexible filtering.

**2. Property Recommendation Engine (KNN)**

* Once the user filters the dataset, a KNN model is trained on the filtered encoded data.
* The system then recommends the **top 5 similar properties** to the one the user selects from the results.
* Recommendations are ranked by similarity using **cosine distance** on normalized feature vectors.

**3. Price Prediction Engine**

* The system allows the user to either:
  + Choose a property and predict its estimated price using a trained regression model.
  + Or, input custom property features to get a price estimate.
* Models used include **Random Forest Regressor** and **Linear Regression**, trained on historical real estate data.

**4. User Flow Overview**

* **Step 1:** User provides inputs (e.g., city = "Maadi", min\_price = 1,000,000).
* **Step 2:** Fuzzy filter narrows the dataset.
* **Step 3:** User selects a property or inputs a feature set.
* **Step 4:** System recommends top 5 similar properties using KNN.
* **Step 5:** User can also get a **predicted price** for a property or custom features.

**5. System Structure (Backend-Only ML)**

* Implemented entirely in Python using:
  + pandas, scikit-learn, rapidfuzz, matplotlib
  + Cleanly separated into filtering, encoding, recommendation, and regression modules
* Suitable for running in **Jupyter Notebook or Google Colab**
* Easily integratable into Streamlit or FastAPI apps for UI/API development in the future

## 3. Related Work

**1. Zillow**

Zillow is one of the largest online real estate marketplaces in the U.S., providing property listings, home value estimates, and market analytics. It helps users buy, sell, rent, and finance properties with a data-driven approach.

**2. Realtor.com**

Realtor.com is a leading U.S.-based real estate platform that provides property listings, market insights, and connections with real estate professionals. It is officially affiliated with the National Association of Realtors (NAR), giving it credibility and access to extensive MLS (Multiple Listing Service) data.

**3 DocuSign**

DocuSign is a leading digital transaction management (DTM) platform specializing in **electronic signatures (e-signatures) and document automation**. It is widely used across various industries, including **real estate, finance, and legal services**, to simplify and secure contract management.

**5. Aqarmap**

Aqarmap is a leading real estate platform in the Middle East, primarily focuse d on the Egyptian and Gulf markets. It provides property listings for **sale, rent**, and **commercial real estate**. Aqarmap aims to streamline the property buying and selling process through its user-friendly platform.

**6. OLX Real Estate**

OLX Real Estate is a section of the global OLX platform, which is known for classifieds and buying/selling items, including real estate. It is available in many countries and allows users to post property listings for sale or rent. OLX targets a broad audience, providing both individuals and agents with an easy way to advertise properties.

### 3.1 Comparing models:

* **Aqar-Hunt** leads in: AI-powered recommendations, anomaly detection, and smart documentation management.
* **Zillow** **& Realtor.com** excel in market presence and data volume, but they lack advanced AI-driven document management.
* **DocuSign** is strong in electronic contracts, but lacks real estate recommendations and market insights.
* **Bayut, Aqarmap**, and **OLX** have strong regional market reach, but lack AI-driven fraud detection and smart recommendations.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Feature / Application** | **Aqar-Hunt** | **Zillow** | **Realtor.com** | **DocuSign** | **Bayut** | **Aqarmap** | **OLX Real Estate** |
| **Smart Recommendation System (Content-Based Filtering + K-Means Clustering)** | **✔** | ❌ | ❌ | ❌ | ❌ | ❌ | ❌ |
| **Anomaly Detection (Fraud & Pricing Issues)** | **✔** | ❌ | ❌ | ❌ | ❌ | ❌ | ❌ |
| **Advanced Data Analysis for Market Trends** | **✔** | **✔** | **✔** | ❌ | ❌ | ❌ | ❌ |
| **Smart Real Estate Documentation Management** | **✔** | ❌ | ❌ | **✔** | ❌ | ❌ | ❌ |
| **Global Market Reach** | ❌ | **✔** | **✔** | **✔** | **✔** | **✔** | **✔** |
| **Large Property Database** | ❌ | **✔** | **✔** | ❌ | **✔** | **✔** | **✔** |
| **Easy Property Search** | **✔** | **✔** | **✔** | ❌ | **✔** | **✔** | **✔** |
| **Modern & User-Friendly Interface** | **✔** | **✔** | **✔** | **✔** | **✔** | **✔** | **✔** |
| **Focus on the Arab Market** | **✔** | ❌ | ❌ | ❌ | **✔** | **✔** | **✔** |

## CHAPTER 3

## Requirements and Work Details

### ****3.1 Project Stakeholders****

Our system targets two core user groups interacting with real estate pricing and recommendation data:

#### **Property Sellers and Real Estate Companies**

* These users contribute property listings including multiple features such as price, location (city, town, latitude, longitude), property type, size, bedroom and bathroom count, furnishing status, completion status, and more.
* Their input serves as the base for clustering, training, and improving the pricing and recommendation engine.

#### **Property Seekers (Buyers and Investors)**

* These users receive personalized real estate recommendations based on their location, preferences, and budget.
* The system uses machine learning models (e.g., ensemble regressors) to estimate the fair market value of properties and suggest alternatives.
* Buyers can view ranked properties based on proximity, price similarity, or quality of match with their preferences.

### ****3.2 Use Case Requirements****

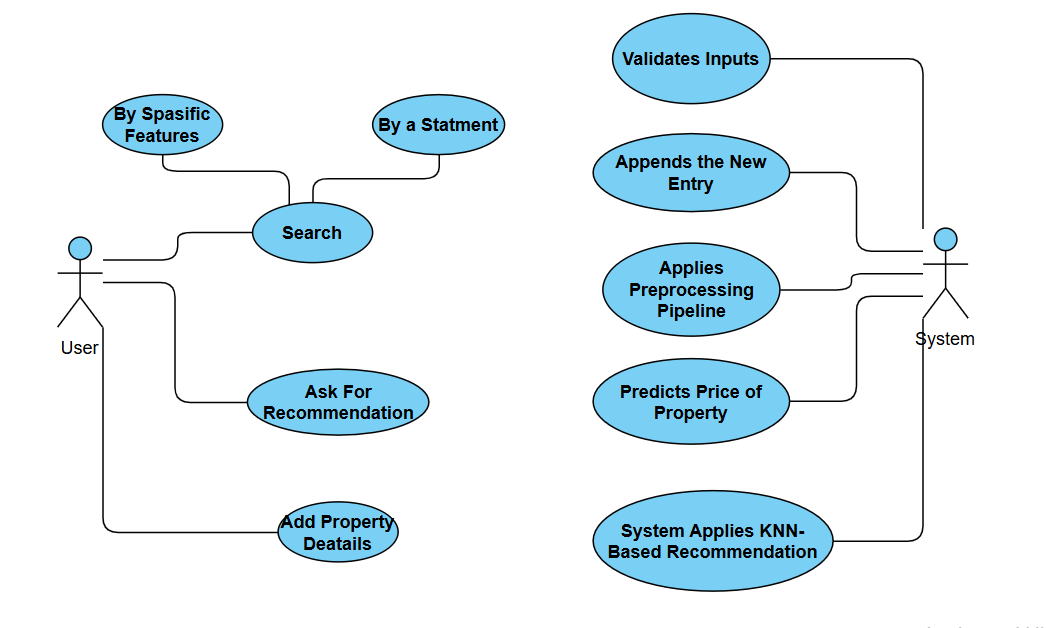
****

Figure Use Case Diagram

#### **3.2.1 Requirements**

To build and deploy the real estate pricing prediction and recommendation engine, the following core use cases and functionalities are identified:

#### **Property Dataset Management**

* **Data Input**
* Sellers or system admins must be able to input property records with multiple attributes:  
  {property\_type, city, town, district, lat, lon, bedrooms, bathrooms, size, completion\_status, furnished, offering\_type, down\_payment\_price, price}
* **Data Storage & Cleaning**
* The system preprocesses the dataset: removes outliers, handles nulls, and encodes categorical features using one-hot encoding, Label Encoding and scaling for numerical features.

#### **Price Prediction**

* **Automatic Price Estimation**
* The system predicts property price based on features using a trained ensemble regressor (including bagging and boosting models).
* **Model Evaluation and Selection**
* Multiple regressors are evaluated using R², MAE, MSE, and RMSE. The best-performing model is selected for deployment.

#### **Recommendation Engine**

* **KNN-Based Recommendation**
* Users input a selected property; the system returns the top-N most similar listings based on cosine similarity in feature space.
* **Dynamic Filtering**
* Buyers can apply filters such as price range, location, or furnishing status, which the system dynamically incorporates into its recommendation logic.

#### **Backend and Deployment**

* **Pipeline Integration**
* All preprocessing steps and models are wrapped in reusable pipelines for deployment and batch prediction.
* **Dense Feature Conversion**
* HistGradientBoostingRegressor and similar models require dense inputs after transformation (handled via FunctionTransformer).

### ****3.2.2 User System Requirements****

The system requirements are designed for researchers, developers, and real estate analysts working with ML-based property pricing tools.

### ****Prediction Pipeline****

* The ML pipeline is modular and consists of:
  + ColumnTransformer: Handles categorical and numerical encoding.
  + Ensemble Regressor: Includes HistGradientBoosting.
  + Evaluation Metrics: Real-time feedback through R², MAE, MSE, and RMSE.

### ****Recommendation System****

* Uses NearestNeighbors (KNN) for similarity search.
* Includes filtering for city, type, budget range, and furnishing.
* Users can optionally view fuzzy matches based on partial criteria.

### ****Model Comparison Framework****

* Includes baseline models (e.g., Decision Tree, Linear Regression) and advanced ensemble models.
* Each model is evaluated and visualized using Actual vs Predicted scatter plots and metric tables.

## ****3.3 Project Non-Functional Requirements****

#### **3.3.1 Execution Qualities**

**• Performance**

* Prediction and recommendation latency < 2s for test inputs.
* Models trained on thousands of records without noticeable delay using optimized algorithms and parallel jobs.

**• Scalability**

* Designed to scale with larger datasets (from few hundred to 100K+ records).
* Uses n\_jobs=-1 and efficient LightGBM/XGBoost methods.

**• Reliability**

* Preprocessing and pipelines tested across multiple runs with consistent results.
* Gracefully handles missing or malformed data.

**• Security**

* Although the system is ML-only (no deployed UI), it's structured for safe integration with future APIs or UIs using role-based access and secure preprocessing.

**• Usability**

* Predefined pipelines and comments ensure easy readability and extension for new developers or ML engineers.

**• Maintainability**

* Code is modular, organized by function, and ready for deployment.
* Hyperparameters and model selection are easy to update without restructuring.

### ****3.3.2 Evaluation Qualities****

**• Accuracy**

* Best models (e.g, HistGradientBoosting, Random Forest) achieve R² > 0.81 on test sets.
* The recommendations provided should be accurate and relevant to the user's input data, with a success rate of at least 90% in meeting user dietary preferences and needs.

**• User Satisfaction**

* Designed for accurate price insights and personalized property recommendations, ensuring users receive meaningful and trustworthy results.

**• Interoperability**

* Output predictions and recommendations are provided in interpretable formats (DataFrames) and easily exported or integrated with other tools.

**• Compliance**

* All processing adheres to ML fairness and reproducibility principles. Data handling is structured for anonymized use.

**• Efficiency**

* Uses optimized algorithms and avoids computational waste by applying sparse-to-dense conversion only when necessary.
* The system should optimize resource usage, ensure minimal CPU and memory consumption while delivering high performance.
* Efficient algorithms and data structures should be used to process user inputs and generate recommendations.

## 3.4 Alternative Flow

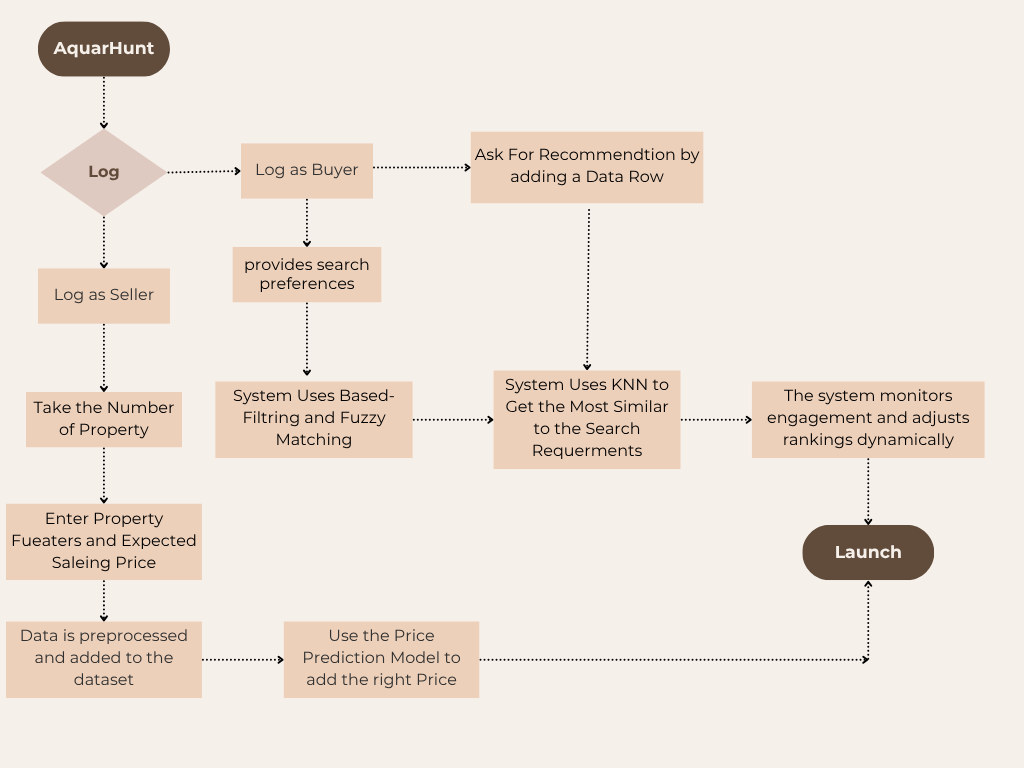


Figure 4 Project Flochart Diagram

# CHAPTER 4

# Data

Data collection is the process of obtaining relevant and meaningful data to support the predictive and recommendation goals of our real estate system. In our case, we gathered a comprehensive real estate dataset by scraping property listings from the "Property Finder" website. This allowed us to collect live, diverse, and updated property information relevant to the Egyptian market.

The dataset includes both numerical and categorical attributes describing various aspects of real estate properties such as their location, features, financial offerings, and construction status. This data serves as the foundation for building both our price prediction and recommendation components.

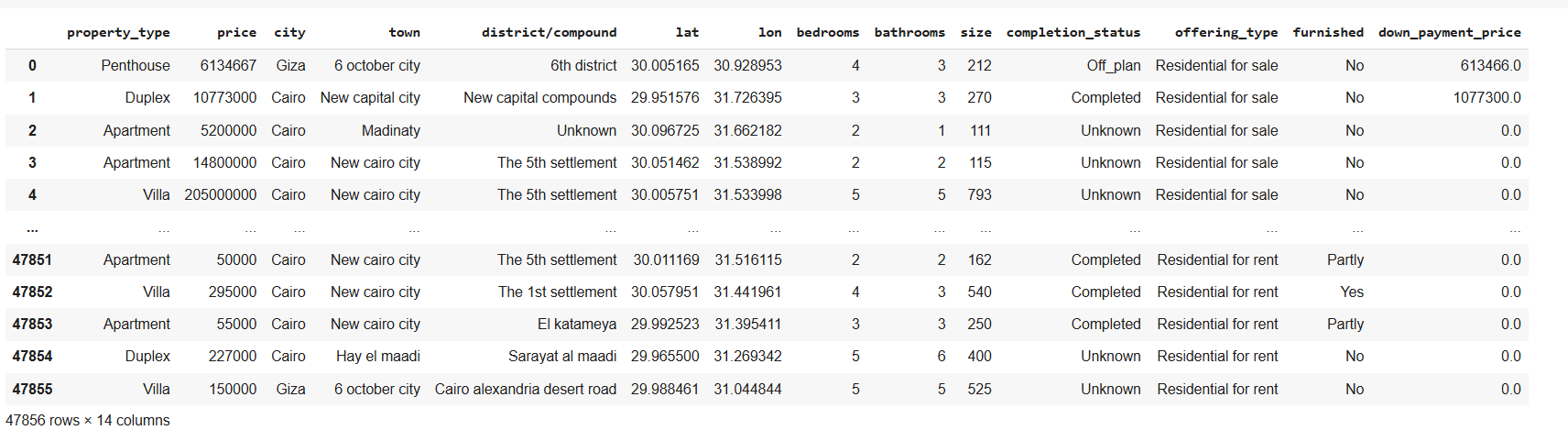


Figure 5 Sample Raw Data Snapshot

**4.1 Features Description**

The dataset comprises the following key features:

* **Property\_Type**: (Categorical) Type of the property, e.g., Apartment, Villa, Chalet, etc.
* **Price**: (Numerical) Total price of the property in EGP.
* **City**: (Categorical) The main city where the property is located.
* **Town**: (Categorical) The sub-area or locality within the city.
* **District/Compound**: (Categorical) Name of the district or compound.
* **Lat, Lon**: (Numerical) Geographical coordinates of the property.
* **Bedrooms**: (Numerical) Number of bedrooms.
* **Bathrooms**: (Numerical) Number of bathrooms.
* **Size**: (Numerical) Size of the property in square meters.
* **Completion\_Status**: (Categorical) Indicates whether the property is completed or under construction.
* **Offering\_Type**: (Categorical) The type of offering, such as Sale or Rent.
* **Furnished**: (Categorical) Whether the property is furnished, partially furnished, or not.
* **Down\_Payment\_Price**: (Numerical) Amount of down payment required.
* **Listed\_Date**: (Timestamp) Original listing date (removed after preprocessing).
* **URL/ID/Amenities**: (Dropped irrelevant metadata columns)

**4.2 Data Cleaning and Preprocessing**

Data preprocessing ensures that our dataset is clean, consistent, and machine-learning-ready. We performed the following key steps:

* **Missing Values Handling**:
  + Removed rows where 'bathrooms' had invalid 'None' values.
  + Converted bedrooms and bathrooms to numeric and dropped rows with conversion errors.
  + Filled missing values in down\_payment\_price with 0.
  + Filled missing values in district/compound with "Unknown".
  + Set missing completion\_status based on the furnished value.
* **Duplicates**:
  + Detected and removed duplicate listings.
  + Reset index after duplicate removal.
* **Data Consistency**:
  + Standardized all object/categorical values using consistent capitalization.
  + Dropped irrelevant or noisy columns such as URLs, IDs, and listing metadata.
* **Datetime Conversion**:
  + Converted listed\_date from UTC format to datetime (removed after use).

*A screenshot of a computer

AI-generated content may be incorrect.*

Figure Missing Values Summary Before Cleaning

*A screenshot of a computer

AI-generated content may be incorrect.*

Figure 7 Data Info and Types After Preprocessing

**4.3 Data Analysis**

We conducted a comprehensive data analysis to better understand the characteristics of the dataset and the distribution of property prices and features.

**4.3.1 Distribution and Trends**

* **Price Distribution**:  
  A histogram showing the price distribution (filtered below 95th percentile) indicates a right-skewed distribution with most properties clustered under mid-range prices.

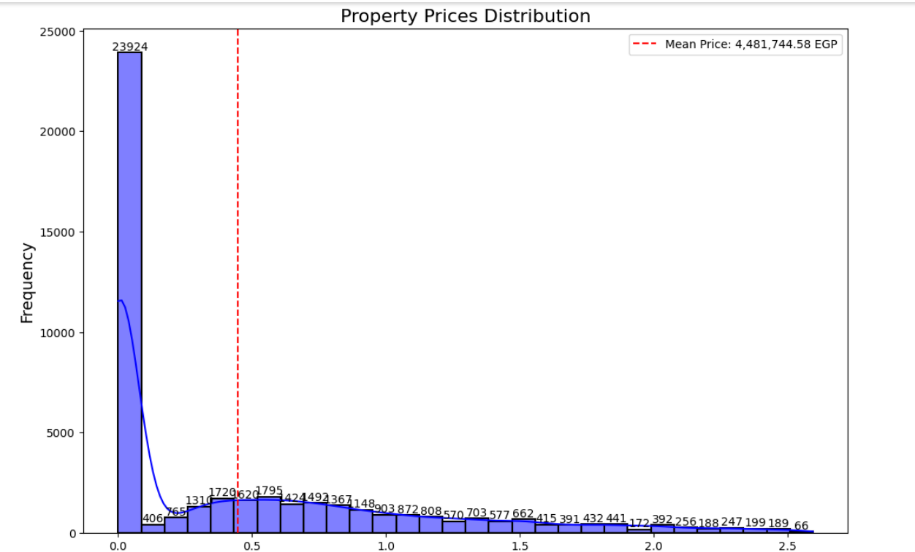
****

Figure 8 Price Distribution Plot

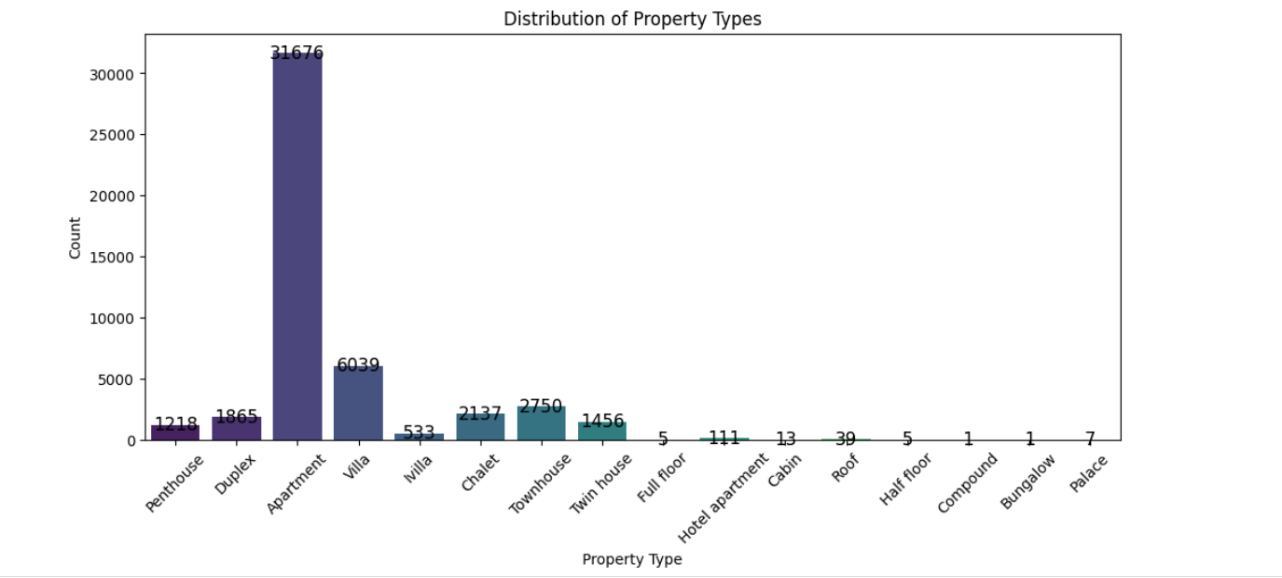
* **Property Types**:  
  Bar plot reveals the dominance of apartments, followed by villas and chalets.

Figure 9 Property Type Frequency

* **Median Price by Property Type**:  
  Used bar chart to highlight how median prices vary by property type.



Figure 10 Median Price by Property Type

**Geographic Trends**:

* + **City Distribution**: Number of properties listed in each major city.
  + **Map Visualization**: An interactive scatter map shows property locations with color-coded pricing.

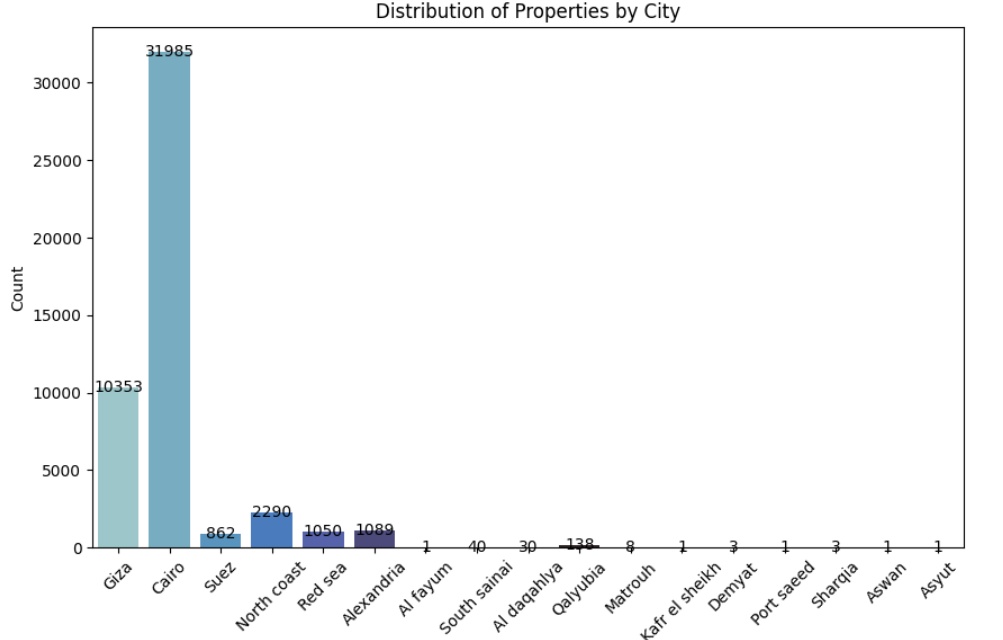
****

Figure 11 City-Wise Property Count

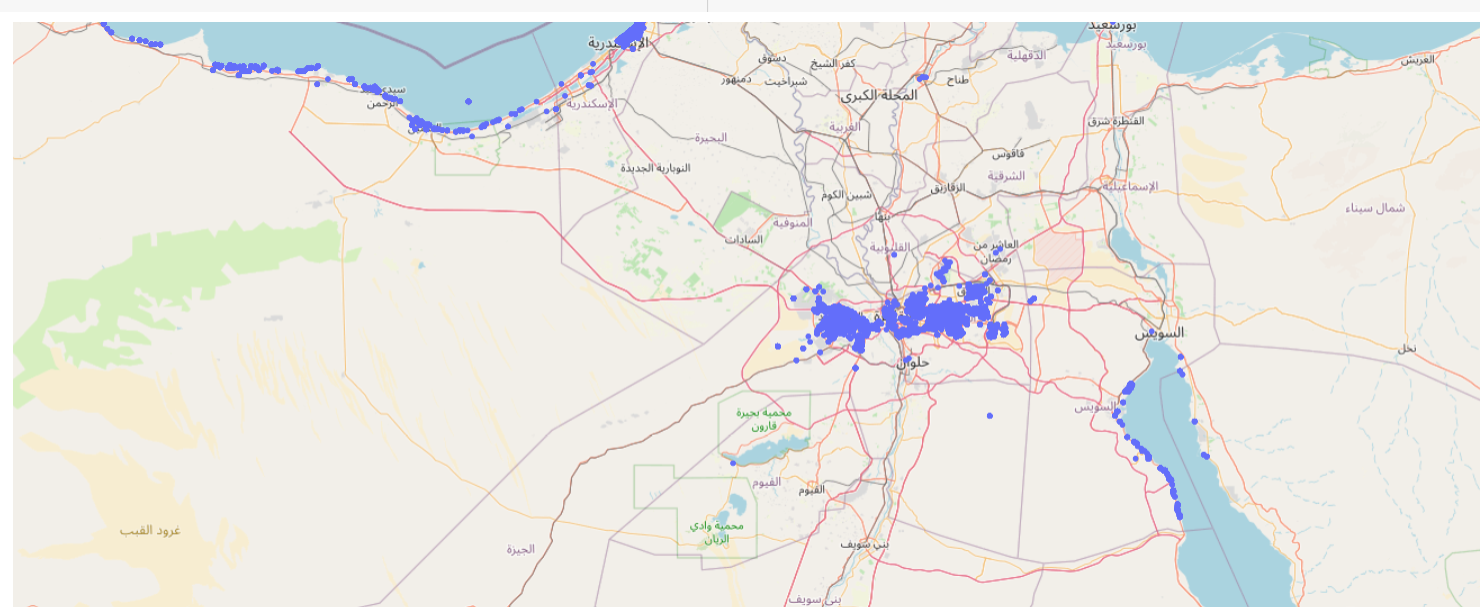
**

Figure 12 Property Location Map

* **Categorical Feature Distributions**:
  + Completion Status
  + Furnished vs. Unfurnished

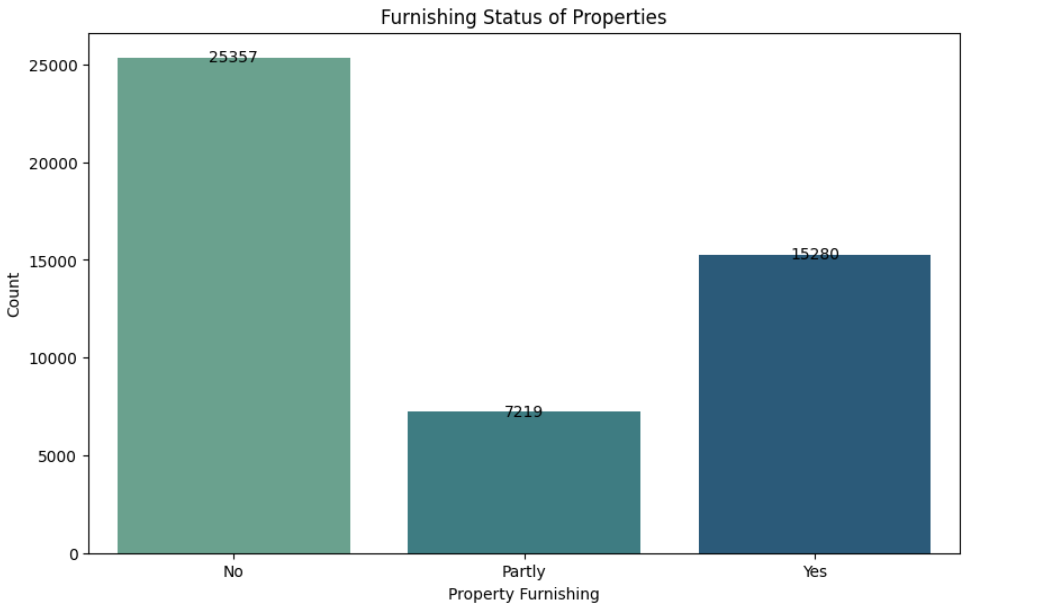


Figure 13 Furnishing Status Distribution

A graph of a graph of properties

AI-generated content may be incorrect.

Figure 14 Completion Status Distribution

**4.3.2 Correlation Matrix**

We constructed a correlation matrix for all numeric features to analyze dependencies:

* **Price** has moderate correlations with size, bedrooms, and bathrooms.
* Very low multicollinearity is observed, confirming the usefulness of each variable.

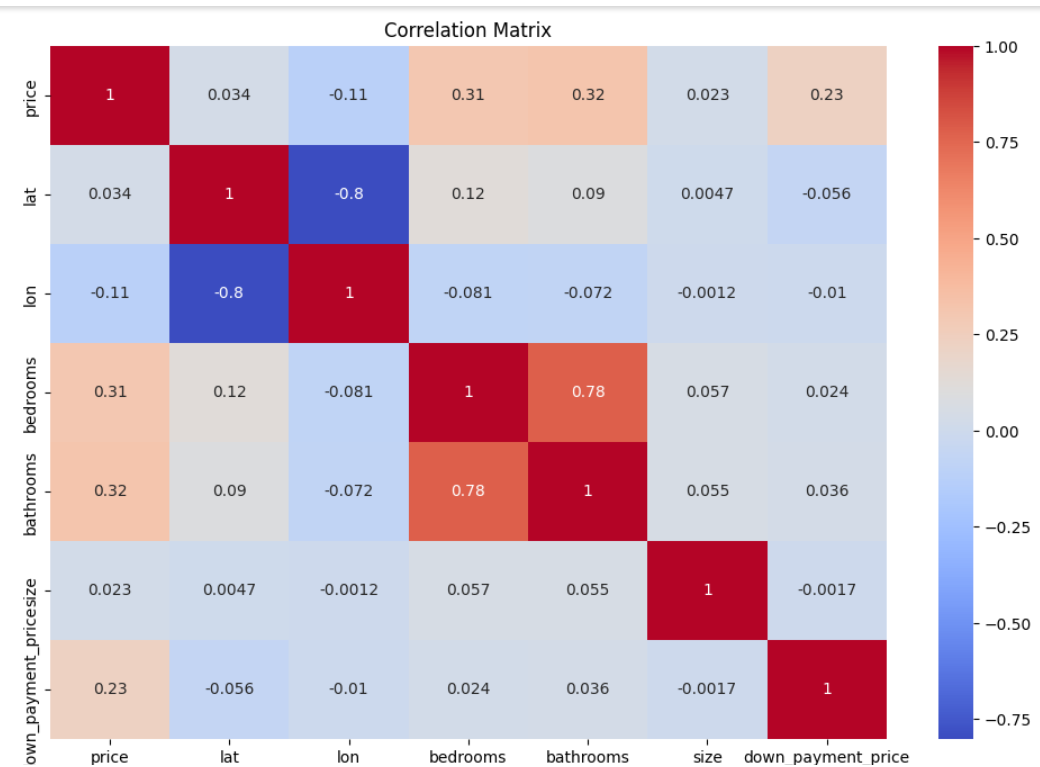


Figure 15 Correlation Heatmap of Numerical Features

**4.4 Summary of Insights**

* Majority of the properties are apartments located in Cairo and Giza.
* Furnished properties tend to have higher completion status.
* Pricing correlates strongly with size and number of rooms.
* Preprocessing greatly reduced missing data, leading to cleaner modeling input.

# CHAPTER 5

## Modeling and design

**5.0 Introduction**

To develop a smart real estate recommendation system, we implemented a multi-model machine learning approach aimed at addressing different user needs: property discovery, filtering, and price prediction. Our modeling phase included three main components:

* A **K-Nearest Neighbors (KNN)** engine for personalized recommendations based on mixed-attribute similarity.
* A **search interface** that enables both dynamic and fuzzy filtering using user-defined or approximate inputs.
* A **price prediction ensemble** using a **Histogram-Based Gradient Boosting Regressor** alongside bagging regressors for robust and accurate price estimation.

These techniques were chosen for their flexibility with structured and mixed data, interpretability, and support for real-time, user-driven decision-making.

**5.1 Our Models**

We utilized the following models and techniques:

* K-Nearest Neighbors (KNN)
* Filtering and Fuzzy Search (via rule-based logic and fuzzy string matching)
* Ensemble Regressors:
  + Bagging with Decision Trees
  + Histogram-Based Gradient Boosting Regressor (HistGradientBoosting)

All models were implemented using Python libraries including scikit-learn, RapidFuzz, NumPy, and Pandas.

**5.1.1 K-Nearest Neighbors (KNN)**

**Main Idea**

K-Nearest Neighbors (KNN) is a **non-parametric, instance-based** learning algorithm that makes predictions or recommendations based on the proximity of data points in a multidimensional feature space. It relies on a **distance or similarity metric** to determine the "closeness" between a given input and existing data samples.

In the context of recommendation systems, KNN identifies the **k most similar items** (neighbors) to a given input item. These neighbors are used to infer similarity, relevance, or predict unknown attributes. In our system, we adopted **cosine similarity** as the distance measure, which compares the angle between feature vectors rather than their magnitudes—making it ideal for our normalized, mixed-feature data.

**Application in Our Project**

In our real estate recommendation system, the KNN algorithm plays a central role in **retrieving and ranking similar properties** based on user interaction or property selection. The implementation followed a structured data preparation process:

1. **Feature Engineering**:
   * We selected a combination of **numerical features** (e.g., price, size, bedrooms, bathrooms, latitude, longitude) and **categorical features** (e.g., property type, city, town, furnishing status, offering type, compound).
   * These features were chosen to reflect user intent and property similarity.
2. **Data Preprocessing**:
   * **Categorical features** were transformed into numerical values using **Label Encoding**, where each unique category is assigned a distinct integer.
   * **Numerical features** were scaled using **Min-Max Normalization** to bring all values into the range [0, 1]. This step ensures fair comparison across features, especially when using cosine similarity which is sensitive to vector direction.
3. **Feature Vector Construction**:
   * All encoded and scaled features were **horizontally stacked** to form a unified numerical matrix representing the entire property dataset.
   * Each row in the matrix corresponds to a property, and each column represents a normalized or encoded feature.
4. **KNN Model Training**:
   * We used NearestNeighbors from **scikit-learn**, with metric='cosine' and algorithm='brute'.
   * The model was **fitted on the preprocessed matrix** (fit(prep\_data)), allowing it to index and compare all property vectors for similarity lookup.
   * Cosine similarity ensures that recommendations focus on the **relative pattern of features**, rather than absolute values, which is ideal when comparing mixed real estate attributes.

**Advantages**

* **High Personalization**: KNN recommends listings that closely match the selected property’s profile, offering tailored and contextually relevant suggestions.
* **No Training Phase**: The algorithm is **lazy-learning**, requiring no explicit model training—this allows for instant updates when new properties are added.
* **Effective for Mixed Data**: Works well when combining **encoded categorical** and **normalized numerical** features.
* **Intuitive and Transparent**: Easy to interpret as it directly uses feature similarity to make decisions.

**Disadvantages**

* **Scalability**: KNN can be computationally expensive with large datasets, as it computes distances to all samples at prediction time.
* **Sensitivity to Irrelevant Features**: Without proper feature selection or weighting, irrelevant data can negatively impact similarity calculations.
* **Cold Start Problem**: For users without a selected listing or historical interaction, KNN alone cannot recommend items (unless combined with other methods).
* **Memory Usage**: Requires keeping all instances in memory, which may increase memory consumption with large property datasets.

**5.1.2 Filtering and Fuzzy Search**

In order to offer users **more control and flexibility** when searching for real estate listings, our system integrates two major complementary techniques: **Dynamic Rule-Based Filtering** and **Fuzzy Filtering via RapidFuzz**. Together, they enhance the recommendation process by narrowing down candidates before applying KNN.

**A. Dynamic Filtering (Rule-Based Search)**

**Main Idea**

Dynamic filtering is a **rule-based search mechanism** that allows users to filter the dataset based on specified conditions such as **location, price range, and property attributes** (e.g., number of bedrooms or bathrooms). It works similarly to traditional database querying but in an interactive way.

This filtering method is useful for users who know exactly what they're looking for and want **precise control over the results**. It operates before applying the KNN algorithm, serving as a pre-selection stage.

**Application in Our Project**

The user is prompted to input one or more filter criteria (such as city, town, price range, etc.). The system dynamically filters the main dataset using these inputs and returns only matching rows. These filtered results are then passed to the **KNN recommendation engine**, allowing for more relevant and context-aware suggestions.

Here is the pseudocode and functionality breakdown:

**Advantages**

* **Precise Filtering**: Users can control their search by explicitly selecting desired features like city, price, etc.
* **Integrates Seamlessly with KNN**: The filtered subset becomes the input space for more accurate neighborhood comparison.
* **Flexible Input**: Users can skip any field to perform partial filtering.

**Disadvantages**

* **Rigid Matching**: Requires exact matches for strings (e.g., city names), leading to failed matches on typos or formatting issues.
* **User Knowledge Required**: Relies on users knowing exact available values in the dataset.

**B. Fuzzy Filtering (via RapidFuzz)**

**Main Idea**

Fuzzy filtering enhances usability by allowing **approximate string matching**. Instead of requiring exact text matches, it compares user input against known values using a **similarity score**. This is especially helpful when the user **misspells** location names or provides **incomplete keywords**.

The system uses **RapidFuzz’s process.extractOne()** method to find the **most similar entry** from a list of valid categories (e.g., cities or compounds) given an arbitrary string input.

**Application in Our Project**

The fuzzy filter takes user inputs like city or compound names and matches them against valid entries using **string similarity thresholds**. If a close match is found (e.g., score > 80%), the matched value is used instead of the raw input.

The fuzzy-matched values are then passed into the **filtering pipeline** and used in further narrowing down the search before passing to KNN.

**Advantages**

* **Error-Tolerant**: Handles typos, misspellings, and partial inputs.
* **Natural Language Friendly**: Improves user experience with flexible, forgiving input processing.
* **No Exact Match Needed**: Makes it easier for users unfamiliar with dataset formatting.

**Disadvantages**

* **False Positives**: If the similarity threshold is too low, unrelated matches may occur.
* **Limited Vocabulary**: Can only match against existing known values in the dataset (e.g., existing cities).

By combining **rule-based dynamic filtering** and **fuzzy matching**, our system gives users multiple intuitive ways to narrow down search results. These filtered results are then sent to the **KNN engine** for similarity-based recommendations. This hybrid approach allows for both **precision and flexibility**, offering a significantly enhanced property discovery experience.

**5.1.3 Ensemble Regression Models**

To accurately predict real estate prices from a combination of numerical and categorical features, we implemented two robust **ensemble-based regression models**: **Bagging Regressor** and **Histogram-Based Gradient Boosting Regressor**. These models were chosen for their ability to handle **nonlinear relationships**, **high-dimensional data**, and to generalize well without extensive manual tuning.

**A. Bagging Regressor**

**Main Idea**

**Bagging** (Bootstrap Aggregating) is an ensemble technique that builds multiple independent base estimators (typically decision trees), each trained on random subsets of the training data. The outputs of these models are then averaged to reduce variance and overfitting, resulting in a more stable prediction.

**Application in Our Project**

We used the BaggingRegressor from sklearn.ensemble, wrapping a DecisionTreeRegressor as the base estimator. A total of **500 estimators** were trained using **80% bootstrapped samples** from the training set, with **OOB (Out-of-Bag) scoring** enabled for internal validation.

The input features include **normalized numerical data** and **one-hot encoded categorical attributes**, combined through a ColumnTransformer. The full process is encapsulated in a Pipeline.

**Advantages**

* **Variance Reduction**: Aggregating over many trees helps prevent overfitting.
* **Built-in Validation**: OOB samples offer unbiased internal performance estimation.
* **Simplicity**: Easy to implement with minimal parameter tuning.

**Disadvantages**

* **Less Interpretable**: Individual predictions are harder to explain compared to single models.
* **Memory Intensive**: Storing and evaluating 500 trees can be resource demanding.

**B. Histogram-Based Gradient Boosting (HistGradientBoosting)**

**Main Idea**

**HistGradientBoostingRegressor (HGBR)** is a highly optimized, tree-based boosting algorithm. It discretizes (bins) continuous features into histograms, enabling fast training and efficient memory usage. The model is built iteratively, where each tree corrects the errors of the previous ones using **gradient descent** over the loss function.

**Application in Our Project**

We used HistGradientBoostingRegressor within a separate pipeline, applied **after transforming the data with ColumnTransformer** and converting it to dense arrays (as HGBR does not accept sparse matrices).

The model was configured with:

* **1000 iterations (trees)**
* **Max depth = 10**
* **Early stopping** based on validation error
* **Learning rate = 0.1**

**Evaluation Metrics**

* **R² Score**: Indicates goodness of fit.
* **MAE (Mean Absolute Error)**: Measures average error in prediction.
* **RMSE (Root Mean Squared Error)**: Penalizes large errors more than MAE.

**Advantages**

* **High Performance**: Handles both numerical and categorical data efficiently.
* **Handles Missing Values**: Natively supports missing data during training.
* **Fast Training**: Histogram-based binning speeds up large dataset processing.
* **Accurate**: Yields high prediction accuracy with robust generalization.

**Disadvantages**

* **Complexity**: More sensitive to parameter tuning than basic models.
* **Black-Box Nature**: Predictions are difficult to explain intuitively.

**5.2 Required Technologies and Libraries**

Our system is built using a modern Python ecosystem, with libraries specifically selected to support machine learning, data preprocessing, model evaluation, and user-friendly interfaces. Below is an overview of the key libraries and their respective roles:

| **Library / Tool** | **Purpose and Functionality** |
| --- | --- |
| **scikit-learn** | The core machine learning library used for training, validating, and deploying models. It provides tools for splitting data, fitting estimators, evaluating performance, and building preprocessing pipelines. Models such as KNN, BaggingRegressor, and HistGradientBoostingRegressor were implemented from this package. |
| **RapidFuzz** | A powerful fuzzy string matching library, used to handle approximate user inputs in search queries. It computes string similarity using Levenshtein distance and allows the system to suggest close matches even when user input is misspelled or incomplete. |
| **NumPy** | A foundational library for numerical computing. It handles the creation and manipulation of arrays and vectors. Used throughout for matrix operations, reshaping input data for models, and computing metrics such as cosine similarity. |
| **Pandas** | A high-level library for working with structured tabular data. It supports reading, cleaning, transforming, and filtering datasets using DataFrames. All property listings and features were processed with Pandas for compatibility with ML models. |
| **Matplotlib** | Used for visualizing actual vs. predicted values in scatter plots, helping to assess model accuracy. This library enables quick generation of meaningful visual diagnostics of model performance. |
| **LabelEncoder** | A utility from sklearn.preprocessing that converts categorical string labels into numerical integers. Each category is assigned a unique integer value, allowing them to be used in numerical models such as KNN or regression estimators. |
| **MinMaxScaler** | Also from sklearn.preprocessing, this scaler normalizes numerical features to a 0–1 range. This ensures that features with different scales (e.g., price vs. number of bedrooms) contribute equally to distance-based models like KNN. |
| **FunctionTransformer** | A transformer wrapper that allows custom data conversions to be applied within a Pipeline. In our project, it was used to convert sparse matrices (resulting from one-hot encoding) into dense arrays required by HistGradientBoostingRegressor. |

# CHAPTER 6

# CONCLUSION AND FUTUURE WORK

**6.1 Project Conclusion**

In conclusion, our real estate recommendation system is a complete intelligent platform designed to enhance the real estate experience for both property seekers and sellers. By combining machine learning techniques with smart filtering mechanisms, the system delivers a highly personalized and efficient solution for property recommendation and price estimation.

The system integrates several core components: a K-Nearest Neighbors (KNN) model for buyer-item matching, search filtering with fuzzy logic to improve discoverability, and a powerful ensemble-based regression engine using HistGradientBoosting to predict property prices with high accuracy. Together, these components allow users to discover properties that match their preferences while receiving reliable price guidance based on market patterns.

We began by thoroughly cleaning and analyzing a real estate dataset collected through web scraping from the Property Finder platform. Through rigorous preprocessing, feature transformation, and visualization, we gained deep insights into the market trends, which shaped our modeling choices.

Our solution supports two major user roles: sellers can submit property listings that are integrated into the system in real-time, and buyers can filter properties by various features (type, location, price range, etc.), get similar recommendations, and understand fair pricing using our predictive models.

While this system achieves a strong level of functionality and accuracy, we are committed to continuous improvement by expanding its features, optimizing performance, and enriching user interaction.

**6.2 Project Limitations**

Despite the strengths and promising results of our system, a few limitations remain:

* **Data Quality and Scope**  
  The performance of the recommendation and prediction modules relies heavily on the quality and diversity of the data. Any missing or outdated property listings can reduce the effectiveness of the system. Additionally, the dataset was limited to properties available at a specific point in time and region.
* **Personalization Depth**  
  Although the system provides recommendations based on clustering and similarity, it does not yet incorporate long-term user behavior, previous searches, or implicit interests to personalize results more deeply.
* **Cold Start for New Users and Sellers**  
  For new users without prior interaction data or new sellers with no historical property records, the system may initially struggle to deliver highly tailored results.
* **Computational Cost**  
  Techniques like KNN and boosting models require non-trivial computational resources, especially when real-time recommendations or predictions are needed for large datasets.
* **Interface Limitations**  
  The current system is presented as a functional backend pipeline. While it supports dynamic filtering and returns recommendations, the user interface layer is not integrated, which limits interaction in a live, user-friendly application.
* **Security and Privacy**  
  While no sensitive personal data is currently collected, any future extensions involving user profiles or account data would require secure handling, encryption, and adherence to data protection laws.

**6.3 Future Work**

Looking forward, we plan several enhancements to strengthen the capabilities and scalability of our real estate recommendation system:

* **Production Model Deployment**  
  Our machine learning models, especially KNN and HistGradientBoosting, will be deployed through scalable APIs using platforms like FastAPI or Flask. This will allow real-time interaction, model retraining with new data, and reliable integration into full-stack applications.
* **Advanced Filtering and Smart Ranking**  
  We aim to combine multiple recommendation signals (search filters, similarity scores, user interaction) into a hybrid ranking system. This will prioritize results not only by similarity but also by relevance and predicted value.
* **User Feedback Loop**  
  Introducing user ratings, preferences, and click-tracking will help us improve the personalization engine. Feedback will be used to retrain or reweight recommendation outputs dynamically.
* **Interactive Frontend with Mapping and Filters**  
  A future UI developed in frameworks like Streamlit or React will offer interactive filtering, real-time map-based search, and graphical display of pricing and recommendation results, enhancing user engagement and accessibility.
* **Automated Property Categorization**  
  With further model enhancements, we plan to classify properties into market segments (luxury, affordable, investment-friendly, etc.) and offer insights to users and realtors based on pricing trends and property attributes.
* **Geospatial Intelligence**  
  By integrating real-time geolocation services and external data like amenities, crime rates, and transportation, we can further improve the context of property recommendations.
* **Cloud Storage and Real-time Syncing**  
  Using cloud platforms like AWS or Firebase, we will enable live updates, seller-buyer synchronization, and robust backend storage for real-time property listing ingestion.
* **Community Features and Social Integration**  
  As the platform grows, we plan to implement forums or review sections where buyers can discuss properties or experiences, and sellers can showcase verified credentials and testimonials.

**References**

1. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd ed. O'Reilly Media.
2. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, pp.2825–2830.
3. Natekin, A., & Knoll, A. (2013). *Gradient boosting machines, a tutorial*. Frontiers in Neurorobotics, 7, Article 21.
4. Real Estate Market Analysis Datasets - Property Finder (Scraped and compiled internally).
5. Official Streamlit Docs. https://docs.streamlit.io
6. FastAPI Framework. https://fastapi.tiangolo.com
7. XGBoost Documentation. <https://xgboost.readthedocs.io>
8. Using ensemble methods of machine learning to predict real estate prices <https://arxiv.org/pdf/2504.04303>
9. House Price Prediction and Recommendation Using CBF and KNN <https://ijirt.org/publishedpaper/IJIRT178609_PAPER.pdf>
10. From Single Trees to Forests: Enhancing Real Estate Predictions with Ensembles <https://machinelearningmastery.com/from-single-trees-to-forests-enhancing-real-estate-predictions-with-ensembles/>
11. LightGBM Documentation. <https://lightgbm.readthedocs.io>

## APPENDICES

**Appendix i: Terminologies and Definitions**

| **Term** | **Definition** |
| --- | --- |
| KNN (K-Nearest Neighbors) | A machine learning algorithm that recommends items based on their similarity to the user input. |
| Cosine Similarity | A metric used to measure similarity between two vectors, used in recommendation. |
| Fuzzy Search | A technique that finds approximate matches to input queries (e.g., misspelled city names). |
| Filtering | A process of narrowing down listings based on user-selected criteria. |
| HistGradientBoosting | An efficient and scalable boosting algorithm used for regression and prediction tasks. |
| Encoding | Transforming categorical features into numerical format (e.g., label encoding). |
| Scaling | Normalizing numerical values to improve model performance. |
| Ensemble Model | A machine learning model that combines the predictions of several models to improve accuracy. |

**Appendix ii: Sample Recommendation Output**

**Input**  
User selects:  
• property\_type = Apartment  
• city = Cairo  
• bedrooms = 3  
• budget = 1,000,000 EGP

**Output (Top 3 Recommendations)**

| **Property ID** | **Property Type** | **City** | **Town** | **Price** | **Bedrooms** | **Bathrooms** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 231 | Apartment | Cairo | Nasr City | 980,000 EGP | 3 | 2 |
| 312 | Apartment | Cairo | Maadi | 990,000 EGP | 3 | 1 |
| 178 | Apartment | Cairo | New Cairo | 1,020,000 EGP | 3 | 2 |

**Appendix iii: Source Code Summary**

| **Component** | **Details** |
| --- | --- |
| Data Preprocessing | - Null value handling - Label encoding (e.g., city, town) - MinMaxScaler for continuous columns |
| Recommendation Engine | - Implemented using NearestNeighbors with cosine distance - Returns property details from nearest neighbors |
| Filtering & Fuzzy Search | - Dynamic user filters (city, price, etc.) - Fuzzy matching with RapidFuzz for misspelled inputs |
| Price Prediction | - HistGradientBoostingRegressor pipeline - Trained on encoded + scaled features, predicting price |
| Source File | real\_estate\_system\_final.ipynb (submitted with project) |

**Appendix iv: User Manual**

| **Mode** | **Description** |
| --- | --- |
| **Buyer Mode** | • Inputs: city, budget, bedrooms, etc. • Output: ranked KNN recommendations and predicted prices |
| **Seller Mode** | • Inputs: add new property data • Output: listing becomes part of future recommendations |
| **Search Types** | • **Exact Filter**: uses dropdowns for specific filtering • **Fuzzy Search**: accepts typed input, matches closest options |