HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

CAPSTONE PROJECT

Introduction to Data Science - IT4142E

Apartment Price Prediction in Hanoi

Group 6
Nguyen Trong Huy - 20210451
Trinh Giang Nam - 20215229
Nguyen Chinh Minh - 20215224

Supervisor: PhD. Nguyen Thi Oanh _____

Signature

Assoc. Professor Than Quang Khoat _____

Signature

Department: Computer Science

School: School of Information and Communications Technology

HANOI, 12/2024

ACKNOWLEDGMENT

First and foremost, we express our sincere thanks to our project supervisor, PhD Nguyen Thi Oanh, along with Assoc. Professor Than Quang Khoat, PhD Tran Viet Trung and PhD Bui Thi Mai Anh, whose guidance, expertise, and patience were instrumental in steering this project towards its completion.

Special appreciation goes to our classmates and the teaching assistant whose advice helped us a lot in writing this report. Their willingness to contribute their time and thoughts added depth and authenticity to our work.

Lastly, we extend our gratitude to each other, the members of Group 6. This project was a collaborative effort that required dedication, compromise and teamwork. We have grown individually and collectively through this experience, gaining not just knowledge but also friendships that we treasure.

This project report is not only a reflection of our hard work but also a testament to the support and guidance we received from all those mentioned above. Thank you for making this journey memorable and our project a success.

ABSTRACT

The real estate market in Hanoi is growing rapidly, with apartment prices fluctuating due to various factors such as location, amenities, proximity to transportation, and market demand. With the demand of buying and selling apartment increasing rapidly, we want to create a model to predict the price of apartments based on the factors mentioned above so sellers and customers can either set the price or choose the suitable apartment. Therefore, we chose the topic of predicting apartment price for this problem with the hope of finding a highly accurate model. In this problem, we extracted data from three different websites, processed a total of 16,914 samples and conducted research on traditional machine learning models: Linear Regression, SVR, XGBoost and LSTM. From the experimental results, we found that XGBoost provided the best RMSE, down to 0.7347, and LSTM gave the best score on MAE at 0.5087.

Student

(Group 6 - Introduction to Data Science)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
SVM	Support Vector Machine
SVR	Support Vector Regressor

CHAPTER 1. INTRODUCTION

1.1 Problem Statement

The real estate market in Hanoi is dynamic and influenced by a variety of factors, making accurate price prediction a critical task for buyers, sellers, and policymakers. Apartment prices in Hanoi are determined by a complex interplay of variables, including location, proximity to amenities, size, age of the building, and market trends. Understanding these factors and accurately predicting apartment prices can facilitate informed decision-making, promote transparency, and enhance market efficiency.

Traditional methods of apartment price prediction often rely on linear regression models and limited datasets, which may fail to capture the intricate relationships between variables. This can lead to less precise predictions, hindering stakeholders from making optimal decisions.

With the emergence of data science techniques, there is an opportunity to leverage advanced statistical methods and machine learning algorithms to improve the accuracy of apartment price predictions. Data science approaches allow the integration and analysis of large, multidimensional datasets, uncovering patterns and relationships that traditional methods may overlook.

However, predicting apartment prices in Hanoi presents several challenges. These include handling diverse and potentially incomplete datasets, accounting for regional and temporal variations in the market, and ensuring that models are both accurate and interpretable for users. Additionally, integrating external factors such as economic conditions or urban development plans into the predictive models can further complicate the task.

This project addresses these challenges by applying data science methodologies to develop and evaluate models for apartment price prediction in Hanoi. The goal is to enhance predictive accuracy while ensuring that the models provide actionable insights to stakeholders.

1.2 Capstone Project Objectives

The primary objective of this research is to collect, process data and develop an effective machine learning-based system for predicting apartment price. The focus is on extracting data from various sources, processing the data and leveraging state-of-the-art machine learning techniques to analyze and interpret complex datasets, thereby improving the prediction accuracy compared to traditional methods. The

conceptual framework involves the following key components:

1. Data Collection and Processing: Gathering a comprehensive dataset that includes demographic, behavioral, and genetic factors associated with apartment. The data will be cleaned, analyzed and preprocessed to ensure quality and relevance. This step involves domain knowledge and exploratory data analysis to enhance the model's performance.
2. Feature Engineering: Identifying and engineering relevant features that significantly contribute to apartment price prediction.
3. Model Development and Training: Developing various ML models, including but not limited to logistic regression, support vector machines, and gradient boosting. Each model will be trained and validated using the preprocessed dataset.
4. Model Evaluation and Interpretation: Evaluating the models based on Mean Absolute Error relevant metrics.
5. Implementation and Validation: Implementing the best-performing model and validating its performance on an independent dataset. This step ensures that the model generalizes well and can be reliably used in real-world scenarios.

The research aims to contribute to the field by providing a robust, interpretable, and accurate apartment price prediction model.

1.3 Organization of Report

There are five main chapters in our report:

CHAPTER 1. INTRODUCTION represents the overview of the problem and proposes the objectives and sketch the outlines of solution.

CHAPTER 2. LITERATURE REVIEW provides the fundamental concepts and foundation theory including the selected datasets, pre-trained large language model, and metrics.

CHAPTER 3. METHODOLOGY includes the main scenario applied to solve problem. For each case, we propose the full pipeline to process the data, train the model and evaluate.

CHAPTER 4. EXPERIMENTAL RESULT shows the result for our model, comparison among all the cases. This also compares the results to the current researches.

CHAPTER 5. CONCLUSIONS sums up all the relevant results and proposes the future works on this project.

CHAPTER 2. LITERATURE REVIEW

2.1 Baseline Regressors

2.1.1 Linear Regression

In order to establish a linear relationship between the input elements and the target variable, one of the most well-known and often used regression techniques is linear regression. This method is also straightforward and effective. It makes the assumption that the independent and target variables have a linear connection and looks for the line of best fit to reduce the discrepancy between the expected and actual values. It assumes a linear equation of the form.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{2.1}$$

where Y is the dependent variable, $x_1, x_2, ..., x_n$ are the independent variables, $\beta_0, \beta_1, \beta_2, ..., \beta_n$ are the coefficients to be estimated. It does this by estimating coefficients for each input element. These coefficients represent the effect of each element on the target variable. Linear regression is known for its interpretability and simplicity, as it provides coefficients indicating the effect of each characteristic on the target variable. Using the apartment price dataset, use linear regression and evaluate its performance in terms of predictability and interpretability. In the context of predicting apartment prices, linear regression can be used to better understand the relationship between factors like crime rate, typical room size, and proximity to employment centers and how they affect home prices. By analysing the coefficient values, we can determine which characteristics have a significant impact on predicted prices.

Discussing the mathematical implications of regression in terms of a unitary linear equation it can be found that if there is k observations for (x_i, Y_i) and want the equation to coincide as closely as possible with the observed data.

Suppose the fitted equation is $y = a*x+b$. Substituting x from the observed data gives $y_1 = a * x + b$ and the error in y from the observed data is $error = y - y_1 = y - (a * x + b)$.

Let the objective function be:

$$c = \sum_{i=1}^{k} error_i^2 = \sum_{i=1}^{k} [y_i - (a * x_i + b)]^2 \tag{2.2}$$

In order to make error smaller, i.e., when C takes a very small value, the bias

derivative for a and b is 0:

$$\begin{cases} \frac{\partial_c}{\partial_a} = -2 * \sum_{i=1}^{k} (y_i - a * x_i - b) * x_i = 0 \\ \frac{\partial_c}{\partial_b} = -2 * \sum_{i=1}^{k} (y_i - a * x_i - b) = 0 \end{cases} \tag{2.3}$$

Solve for:

$$\begin{cases} a = \frac{\sum_{i=1}^{k} (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{k} (x_i - \overline{x})^2} \\ b = \overline{y} - a * \overline{x} \end{cases} \tag{2.4}$$

This gives the regression equation:

$$y = a * x + b \tag{2.5}$$

Further, it can be also derived a multiple regression equation using multiple independent variables.

The advantages of linear regression are threefold. The first point is simplicity. Linear regression is easy to understand and interpret. The coefficients shed light on the strength and slant of the investigated relationship between the independent and dependent variables. The second benefit is that it moves quickly. Because linear regression is computationally effective, it can handle massive data sets. The ability to interpret is the third point. Characteristic importance analysis is made possible by the model's provision of characteristic coefficients.

Because it presupposes a linear relationship between variables, linear regression has the drawback that it may not hold true in complex real-world circumstances. Furthermore, linear regression has a low level of complexity. The non-linear correlations between variables may be difficult for linear regression to capture. Finally, outliers might affect linear regression results. The model's performance might be impacted by this.

2.1.2 SVM regressor

A potent method utilized mostly for classification tasks, the Support Vector Machine (SVM) regressor can also be used to solve regression issues. It is also applicable to regression issues, where the SVM regressor translates the data to a high-dimensional feature space with the goal of locating the optimum hyperplane that maximizes the distance between data points. Being used in the context of apartment price forecasting, the SVM regressor can identify the best hyperplane that best fits the data and minimises error. It is particularly effective in dealing with

high-dimensional data, allowing linear and non-linear relationships to be captured using kernel functions. the SVM regressor can provide valuable insights into non-linear relationships in apartment data and may be effective in identifying outliers.

The data was first pre-processed by dealing with missing values and normalized attributes, and then it was analyzed to determine the apartment price forecast. A training set and a test set were created from the dataset. The training set will be used to train each of the four models (linear regression, SVM regressor and XGBoost regressor), which will then be tested using the test set and the relevant evaluation metrics (such as Root Mean Squared Error (RMSE) or Mean Absolute Error (MAE)).

The advantages of the Support Vector Machine regressor are threefold. The first is versatility. SVMs can use different kernel functions to deal with linear and non-linear relationships. The second point is robustness to overfitting. SVMs use regularisation parameters to prevent overfitting and to handle noisy data. The third point is its effectiveness in high-dimensional spaces. sVM performs well in high-dimensional spaces, making it suitable for datasets with many features.

The downside is the computational complexity; SVMs can be computationally expensive, especially for large datasets. In addition, there is the choice of kernel. Choosing the right kernel function and tuning the relevant parameters can be challenging. And interpretability. In contrast to linear regression, SVM models may not provide intuitive explanations.

2.2 Ensemble Learning

2.2.1 XGBoost regressor

The XGBoost regressor is an optimised gradient boosting algorithm that has become popular in machine learning competitions due to its superior performance. It is an integrated learning technique that builds an ensemble of weak decision tree models in a sequential manner and has proven to be very effective in a variety of regression tasks. The application of XGBoost to the apartment price dataset and evaluate its predictive performance compared to other models. The impact of hyperparameter tuning on model accuracy have been discussed. XGBoost works by initially fitting a single decision tree to the data and then sequentially adding more trees to correct for errors made by previous models. It uses gradient descent optimisation to minimise the loss function and improve the predictions of the model. xGBoost is known for its efficiency, scalability and ability to handle complex functional interactions. It also incorporates regularisation techniques to prevent over-fitting.

The main features and benefits of XGBoost are mainly the advantages offered, such as handling missing values, regularisation techniques to prevent over-fitting, and support for parallel processing. It also provides built-in evaluation metrics and feature importance analysis. In terms of parameter tuning and model optimisation, XGBoost involves tuning hyperparameters such as learning rate, tree depth and the number of estimators to optimise model performance. Techniques like cross-validation and grid search can be used to find optimal parameter values.

The formula of XGBoost regressor uses gradient boosting, and the prediction formula can be represented as:

$$Y = \Sigma w_i * f_i(x) \tag{2.6}$$

where Y is the predicted output. w_i is the weight assigned to each weak learner. $f_i(x)$ is the prediction from each weak learner.

Gradient boosting, which sequentially combines weak learners to produce a powerful predictive model, is enhanced in XGBoost. A new weak learner is fitted to the residuals (errors) of the preceding model at each step via the procedure. The total of all the weak learners' guesses, weighted by importance, forms the final forecast. The advantages of the XGBoost regressor are fourfold. The first is high performance. XGBoost is known for its scalability and efficiency, making it suitable for large data sets. The second point is the handling of complex relationships. It can capture complex non-linear relationships between variables. The third point is regularization. XGBoost includes regularization techniques to prevent overfitting and improve generalization. The fourth point is the built-in evaluation metric. Evaluation metrics such as the Mean Square Error (MSE) and the Root Mean Square Error (RMSE) are provided during training.

It has two drawbacks. The first is the computational resources. XGBoost requires more computational resources than simpler models, such as linear regression. The second is the complexity of the tuning. Tuning the hyperparameters of XGBoost requires careful optimization and can be very time-consuming.

2.3 LSTM

Long Short-Term Memory (LSTM) networks, a specialized type of Recurrent Neural Network (RNN), were introduced by Hochreiter and Schmidhuber in 1997 [1]. LSTMs address the vanishing and exploding gradient problems commonly encountered in traditional RNNs, enabling them to learn and remember long-term dependencies in sequential data.

The LSTM architecture is built around a memory cell, which maintains information over extended time periods. This memory cell is controlled by three main gates:

- **Forget Gate:** Decides which information from the previous cell state should be discarded. It is mathematically expressed as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.7}$$

  where f_t is the forget gate activation, x_t is the input vector, h_{t-1} is the previous hidden state, and W_f and b_f are the weights and biases, respectively.

- **Input Gate:** Determines which new information should be added to the memory cell. This is represented as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.8}$$

  and the candidate cell state is calculated as:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.9}$$

- **Output Gate:** Decides the output of the LSTM cell, which is based on the memory cell and the input. This is defined as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.10}$$

  and the hidden state is updated as:

$$h_t = o_t \cdot \tanh(C_t) \tag{2.11}$$

- **Cell State Update:** The cell state is updated as follows:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{2.12}$$

The primary advantages of LSTMs include: ability to model long-term dependencies, resistance to the vanishing gradient problem and flexibility in handling sequences of variable length.

Despite these strengths, LSTMs can be computationally intensive and require careful tuning of hyperparameters for optimal performance.

2.4 Hyperparameter Tuning with Grid Search

Grid search is an optimization technique used to brute force through all possible combinations of a set of variables. Fundamentally it is one of the most basic and simple optimization algorithms, but it's still quite powerful and guarantees finding the most optimal solution to your problem.

It works by creating a grid of all possible combinations of parameter values and testing each combination to find the best one. This grid of parameters is defined before the optimization/search step, hence the name grid search.

One application of grid search is hyperparameter tuning, a commonly used technique to optimize machine learning models. Machine learning models have several parameters that can be adjusted, known as hyperparameters. These hyperparameters have a significant impact on model performance, making it crucial to find an optimized combination so that we can build good models.

In our problem, we will use Grid Search to fine-tune the hyperparameters.

2.5 Model Assessment

2.5.1 Mean Absolute Error (MAE)

Mean Absolute Error measures the average magnitude of errors in a set of predictions without considering their direction. It is the average of the absolute differences between the predicted values and the actual values.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}| \tag{2.13}$$

where:

- n is the number of data points.
- y_i is the actual value.
- $\widehat{y_i}$ is the predicted value.

Significance in Model Training:

+ Robustness to Outliers: MAE is less sensitive to outliers compared to MSE and RMSE because it treats all errors equally without squaring them. This means that during training, the model will aim to minimize the average error without disproportionately focusing on larger errors.
+ Linear Penalty: The linear nature of MAE means that the impact of each error on the model's learning process is directly proportional to the magnitude of

that error.

+ Interpretability: MAE is in the same units as the original data, making it easier to interpret. If the MAE is 5, it means that on average, the model's predictions are off by 5 units.

MAE works well with simple linear models where the goal is to minimize the average deviation without worrying too much about outliers. MAE ideal for applications where you need to avoid the influence of outliers or when the cost of an error is linear, such as estimating delivery times or predicting scores.

2.5.2 Root Mean Squared Error (RMSE)

Root Mean Squared Error is the square root of the MSE. It brings the metric back to the original scale of the data, making it easier to interpret than MSE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2} \tag{2.14}$$

where:

- n is the number of data points.
- y_i is the actual value.
- $\widehat{y_i}$ is the predicted value.

Significance in Model Training:

+ Balance between MAE and MSE: RMSE retains the sensitivity to outliers like MSE but brings the error metric back to the original scale of the data, making it more interpretable than MSE.
+ Penalizes Large Errors: Similar to MSE, RMSE also penalizes larger errors more due to the squaring process, but because it takes the square root, it doesn't exaggerate them as much as MSE does.
+ Interpretable Units: Since RMSE is on the same scale as the original data, it's easier to understand in the context of the problem. For instance, an RMSE of 5 means that on average, the model's prediction errors are about 5 units away from the actual values.
+ Optimization in Complex Models: RMSE is often used in models where the distribution of errors is important, such as in complex regression models or neural networks.

RMSE is commonly used in more complex models because they provide a smoother

gradient, which is essential for optimization techniques like gradient descent. The penalization of larger errors can also help in fine-tuning complex models. RMSE best for situations where large errors are particularly undesirable and where the application demands a higher penalty for these errors, such as in high-stakes financial predictions, safety-critical systems, or when optimizing models in competitive environments.

CHAPTER 3. METHODOLOGY

3.1 Data crawling

Data crawling is a process of automatically extracting information from websites by making use of computer programs. In this project, we crawl data mainly using HTTP requests and Beautiful Soup library. Beautiful Soup were utilized to retrieve and extract structured data from websites, such as text and images, without the need for an interactive browser. We decided to choose 3 websites: `alonhadat.com.vn`, `homedy.com` and `bds68.com.vn` for their simple structures and sufficiency of the number of samples.

Crawling data using BeautifulSoup involves fetching the HTML content of a webpage and extracting desired information by parsing its structure. First, we use the requests library to send an HTTP request to the target website and retrieve the HTML content. This content is then passed to BeautifulSoup, which parses it into a navigable tree structure. Specific elements, such as tags (<h1>, <a>, etc.), attributes (like href or class), or text, can be located using methods like *find()*, *find_all()*, or CSS selectors. The extracted data is processed as needed, such as by iterating over lists of elements. For multi-page scraping, pagination logic is added by modifying the URL dynamically and looping through pages. Proper error handling and compliance with the site's terms of service are crucial during this process to ensure ethical and effective scraping. In this project, we mainly aim to extract data relevant to apartments such as price, area, direction or number of bedroom.

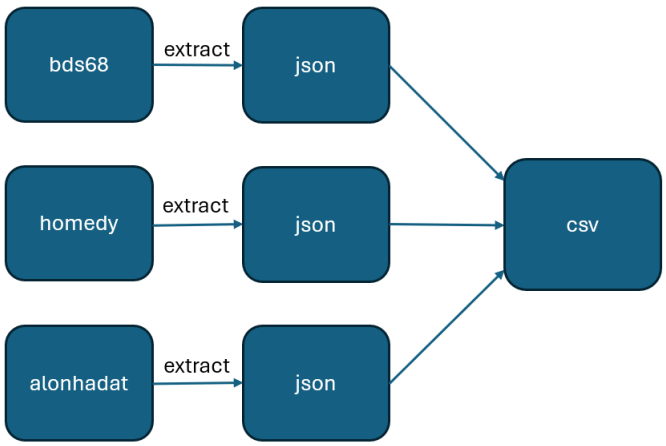

Figure 3.1: Data crawling pipeline

We also need to clean the data in this step. It involves identifying and rectifying

errors, inconsistencies, or ambiguities in the data. First thing is to remove any sample whose target feature is null due to the misinformation from data source. One key aspect of data cleaning is grouping similar terms, such as standardizing variations like "Mua bán Căn hộ chung cư" and "Mua bán Căn hộ" into a single format for uniformity. Additionally, cleaning tasks often include extracting specific information from text, such as isolating location-related terms like "Đường", "Phường" or "Quận" to create structured columns for better analysis.

After careful consideration, we combined the results crawled from three websites mentioned above into one dataset which is a table-type dataset, including 14 columns and 24942 rows. Full description of this dataset is outlined as below:

Feature	**Datatype**	**Description**
duAn	Categorical	The project that the apartment belongs to
huong	Categorical	The direction of the apartment
phapLy	Categorical	Type of legal document to claim for possession of the apartment
noBed	Numerical	Number of Bedroom
soLau	Numerical	Number of Floor
Street	Categorical	Street
Precinct	Categorical	Precinct
District	Categorical	District
City	Categorical	City
acreage_value	Numerical	Area of use
acreage_unit	Categorical	Area unit
price_value	Numerical	The price of the apartment
price_unit	Categorical	Price unit
noBathroom	Numerical	Number of Bathroom

Table 3.1: Description of each column in the dataset

The data contains 14 attributes, 12 of which are categorical. The target feature is price_value.

3.2 Data Preprocessing

Full data collection and processing stage is shown in Figure 3.2 and presented in each section below:

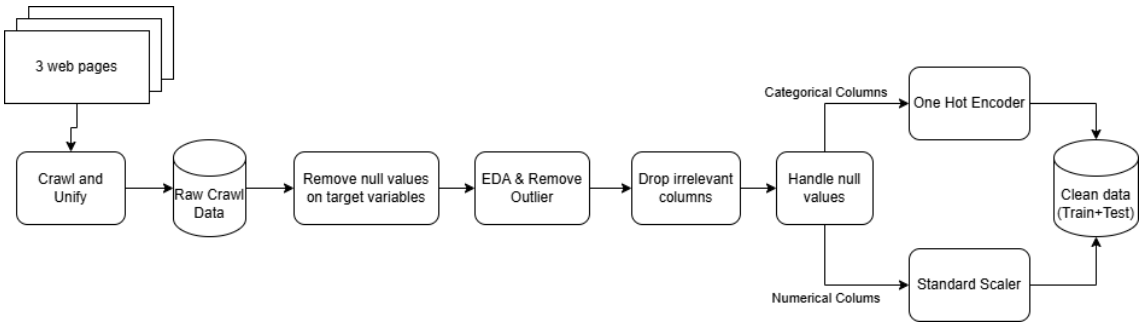

Figure 3.2: Data collection and processing

3.2.1 Preprocessing

Preprocessing is a critical step in the data processing pipeline that ensures the dataset is clean, consistent, and suitable for analysis or modeling. This stage involves systematically preparing raw data by handling missing values, identifying and removing outliers, and transforming the dataset into a structured format that aligns with the objectives of the analysis. The purpose of preprocessing is to improve the quality of the data, minimize noise, and maximize the accuracy and efficiency of subsequent steps, such as exploratory data analysis (EDA) and feature engineering.

In this part, we focus on converting attribute values, removing redundant features and outliers. For example, *price_unit* and *acreage_unit* only contains one or two unique values, so we convert them into one unique value and drop these features. We also remove outliers by using box plot to detect noises and IQR to define the range.

The Interquartile Range (IQR) is a measure of statistical dispersion, representing the range within which the central 50% of the data lies. It is defined as the difference between the third quartile (Q_3) and the first quartile (Q_1):

$$IQR = Q_3 - Q_1 \tag{3.1}$$

Where:

- Q_1: The first quartile, the value below which 25% of the data falls.
- Q_3: The third quartile, the value below which 75% of the data falls.

To detect outliers, the following criteria are often used:

$$\text{Lower Bound} = Q_1 - 1.5 \times \text{IQR}$$

$$\text{Upper Bound} = Q_3 + 1.5 \times \text{IQR}$$

Any data point outside the range $[\text{Lower Bound}, \text{Upper Bound}]$ is considered an outlier.

Additionally, relevant features such as *price_value* and *acreage_value* are also considered to be combined into a new feature so that we can get a better insight of the dataset. The *price_per_area* is the combination feature of the two features mentioned above, which is also used to detect and remove outliers.

3.2.2 Exploratory Data Analysis

For our Exploratory Data Analysis (EDA), we take it in two main steps:

1. Univariate Analysis: We focus on one feature at a time to understand its distribution and range.
2. Bivariate Analysis: In this step, we explore the relationship between each feature and the target variable. This helps us figure out the importance and influence of each feature on the target outcome.

With these two steps, we aim to gain insights into the individual characteristics of the data and also how each feature relates to our main goal: predicting the target variable. The results of this stage is shown in Appendix A.

3.3 Feature Engineering

In this step, the primary focus was on preparing the dataset for machine learning model training through feature engineering. The process began by loading the dataset from the specified file path and removing duplicate records to ensure data integrity. Some exploratory cleaning steps, such as concatenating street and precinct information or dropping unnecessary columns like *duAn*, were prepared for future inclusion but were not applied in this iteration.

The dataset contained a mix of categorical and numerical features. Categorical features included *District*, *phapLy*, *huong*, *Precinct*, and *duAn*, while numerical features included *acreage_value*, *noBed*, *noBathroom*, and *soLau*. Missing values in the numerical columns *noBed*, *noBathroom*, and *soLau* were replaced with -1 to represent unknown values. For categorical features, missing values were filled with the constant placeholder "NO INFO" to ensure consistent handling during the transformation phase.

To process the features effectively, a pipeline-based approach was employed. For categorical data, a pipeline was constructed using the SimpleImputer to fill missing values and the **OneHotEncoder** to convert categorical variables into a

format suitable for machine learning models. The encoder was configured to handle unknown categories gracefully. For numerical data, another pipeline was implemented, which involved imputing missing values using the median strategy followed by scaling the data using **StandardScaler** to standardize numerical distributions.

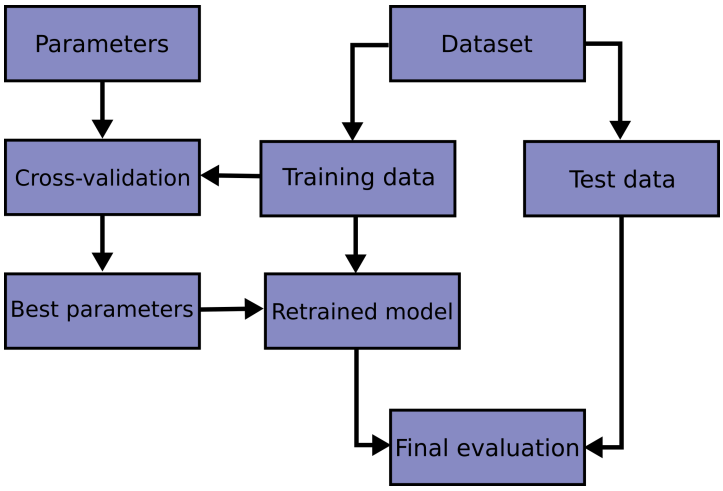

Figure 3.3: Training model with Grid Search workflow [1]

The two pipelines were then integrated into a Column Transformer, which allowed for the simultaneous transformation of categorical and numerical features. The dataset was split into feature variables (X) and the target variable (*price_value*). Additional columns such as Street and *price_per_area* were excluded from the feature set to focus on the primary predictive features. The dataset was then divided into training and testing sets using an 80-20 split, ensuring a fair evaluation process following Figure 3.3 during model development.

Finally, the preprocessing pipeline and the resulting data splits were saved for future use. The pipeline was serialized using **joblib** and saved to the specified path, while the training and testing splits were exported to CSV files for easy accessibility. This ensures reproducibility and simplifies integration with the subsequent model training stages. The completion of this step established a robust foundation for the machine learning workflow, providing clean, well-processed data and a reusable feature transformation pipeline.

3.4 Training Regressor Models

The model training phase focused on developing predictive models for apartment prices using the preprocessed dataset. This process included data preparation, model selection, hyperparameter tuning, evaluation, and model saving.

[1] `https://scikit-learn.org/stable/modules/cross_validation.html`

3.4.1 Data Preparation

The training data, comprising the features (`X_train`) and target (`y_train`), was loaded from preprocessed CSV files. The saved preprocessing pipeline was reloaded and applied to transform the feature set into a suitable format for model training. This ensured consistency in the data preparation and feature engineering process.

3.4.2 Model Selection and Tuning

Model	Hyperparameters
LinearSVR	C: 10
XGBoost	max_depth: 9, learning_rate: 0.2, n_estimators: 800

Table 3.2: Hyperparameters for each model

Three models were employed for this task:

- **Linear Regression**: Used as a baseline model without hyperparameter tuning.
- **Support Vector Regressor (SVR)**: A non-linear model tuned using grid search over the regularization parameter (`C`)).
- **XGBoost Regressor** [2]: A gradient boosting algorithm, tuned for the number of estimators (`n_estimators`), maximum tree depth (`max_depth`), and learning rate (`learning_rate`).
- **LSTM** [1]: The details of LSTM training is shown later in the next subsection.

Hyperparameter tuning for SVR and XGBoost was conducted using `GridSearchCV` with 5-fold cross-validation. The hyperparameters selected were shown in Table 3.2 The optimization criterion was the negative mean absolute error (MAE), ensuring the best-performing hyperparameter combinations for these models.

3.4.3 Model Evaluation

Each model was evaluated using predictions on the training set. The evaluation metrics included:

- **Mean Absolute Error (MAE)**: Calculated as the average absolute difference between predicted and actual values, providing an intuitive measure of prediction accuracy.
- **Root Mean Squared Error (RMSE)**: Penalizes larger prediction errors more heavily, offering insight into the model's sensitivity to outliers.

Performance metrics, along with the best hyperparameters, were logged into a JSON file for further analysis.

3.4.4 Model Saving

The trained models, including the tuned SVR and XGBoost, were serialized and saved as `.pkl` files. This ensured that the models could be reused for predictions or further evaluation in subsequent stages. The preprocessing pipeline was also resaved to maintain compatibility with the trained models.

3.4.5 Summary

This phase resulted in the successful training and saving of three machine learning models: Linear Regression, SVR, and XGBoost. For SVR and XGBoost, hyperparameter tuning significantly improved performance. All models, along with their configurations and evaluation metrics, were systematically logged. This setup provides a robust foundation for deployment and future improvements.

3.5 Training LSTM model

This section describes the process of training a Long Short-Term Memory (LSTM) model for predicting apartment prices. The LSTM network was employed due to its capability to capture sequential dependencies in data, which could potentially enhance the predictive performance for complex relationships in real estate features.

3.5.1 Data Preparation

The first step involved preparing the feature set for the LSTM model. Since LSTM networks require a three-dimensional input structure (*samples, timesteps, features*), the preprocessed data was transformed as follows:

1. **Conversion to Dense Matrix**: The sparse representation of the training feature matrix was converted to a dense matrix using the `toarray()` method.
2. **Normalization**: While normalization using a `MinMaxScaler` was considered, this step was temporarily commented out to experiment with raw feature scaling.
3. **Reshaping for LSTM Input**: The dense matrix was reshaped into a 3D tensor with dimensions (*samples, timesteps, features*), where *timesteps* was set to 1. This ensures that each sample represents a single timestep in the sequence, maintaining compatibility with the LSTM input requirements.

The resulting input tensor had a shape of $(n\_sequences, 1, n\_features)$, where *n_sequences* is the total number of samples divided by the timestep.

3.5.2 Model Architecture

The LSTM network was constructed using Keras' `Sequential` API, with the following architecture:

- **Input Layer**: The input shape was defined as $(1, n\_features)$, where the timestep value was 1.
- **Hidden Layers**:
  - Four stacked LSTM layers with hidden units of 128, 64, 64, and 32, respectively. All LSTM layers used the ReLU activation function to introduce non-linearity and improve convergence.
  - Dropout layers with a rate of 0.2 were interleaved after each LSTM layer to reduce overfitting by randomly deactivating neurons during training.
- **Output Layer**: A Dense layer with a single neuron was used for predicting the apartment price.

Our LSTM includes up to 670,369 parameters. The model summary is presented in Table 3.3.

Layer	Output Shape	Number of Parameters
LSTM (128 units)	(None, 1, 128)	*575488*
Dropout (0.2)	(None, 1, 128)	0
LSTM (64 units)	(None, 1, 64)	*49408*
Dropout (0.2)	(None, 1, 64)	0
LSTM (64 units)	(None, 1, 64)	*33024*
Dropout (0.2)	(None, 1, 64)	0
LSTM (32 units)	(None, 32)	*12416*
Dropout (0.2)	(None, 32)	0
Dense (1 unit)	(None, 1)	*33*

Table 3.3: Summary of the LSTM model architecture.

3.5.3 Training Procedure

The model was compiled using the Adam optimizer and the Mean Absolute Error (MAE) loss function. Adam was chosen for its ability to adapt the learning rate dynamically during training, which enhances optimization performance.

The model was trained using the transformed training set (X_{lstm} and y_{train}) over 200 epochs with a batch size of 64. Validation was conducted on the test set (X_{lstm_test} and y_{test}) to monitor performance. An Early Stopping callback was implemented to terminate training if the validation loss did not improve for 12 consecutive epochs, ensuring the best model weights were restored.

The training history, including loss and validation loss over epochs, was recorded for analysis.

The LSTM model successfully completed training, demonstrating convergence

in both training and validation loss metrics. The final model and training history have been saved for further evaluation and deployment. Future steps will involve comparing the LSTM model's performance against traditional regression models to assess its effectiveness in capturing the complex patterns in the dataset.

4.1 Data Collection

After the entire data collection process, which involved scraping data from websites, followed by data merging and processing, including handling outliers, we obtained a dataset with the following summary statistics in Table 4.1:

Statistics	Quantity
Number of Samples	15,604
Train set	12,483
Test set	3,121
Number of Features	9
Number of Categorical Features	5
Number of Numerical Features	4

Table 4.1: Data statistics

4.2 Modeling Results

With the collected and processed dataset of apartments in Hanoi, we trained machine learning models to predict housing prices using three selected models: Linear Regression, SVM Regressor, and XGBoost. After hyperparameter tuning with K-Fold Cross-Validation, we evaluated the final model on the test dataset.

Model	MAE	RMSE
Linear Regression	0.5430	0.7464
SVR	0.3282	0.5312
XGBoost	0.2915	0.4207
LSTM	0.3689	0.5646

Table 4.2: Model training results on train set

The results of models on train dataset is shown in Table 4.2. As it can show that, XGBoost achieved the best performance overall, with 0.2915 and 0.4207 on MAE and RMSE scores respectively. Meanwhile, SVR got 0.3282 and 0.5312, and Linear seemed to be undefit on the train set. This can be due to the large number of features and the outliers in the train dataset. At the same time, Linear Regression model is not complicated enough and too sensitive to the outliers.

The Table 4.3 shows the results of model evaluation on test set. It can easily seen that the SVR, LSTM and XGBoost model performed the same on the test set, in which LSTM got 0.5087 on MAE and XGBoost got 0.7347 on RMSE scores. While, Linear Regression model achieved the lowest performance overall.

Model	MAE	RMSE
Linear Regression	0.6025	0.8152
SVR	0.5191	0.7465
XGBoost	0.5133	0.7347
LSTM	0.5087	0.7352

Table 4.3: Model training results on test set

These results suggest that the data contains non-linear patterns that advanced models like XGBoost and SVR can capture effectively. The observed overfitting in XGBoost and SVR may indicate noise or insufficient training data, which could be mitigated through better preprocessing, regularization, or increased data availability. Overall, while XGBoost demonstrates the strongest performance, improving data quality and addressing overfitting can further enhance the models' generalization capabilities.

4.3 Feature Extraction

The comparison of feature importance between XGBoost and SVR models reveals significant differences in how these algorithms prioritize features for predicting apartment prices. Figure 4.1 reports highlights the characteristics of each model's feature selection process and provides insights into their strengths and focus.

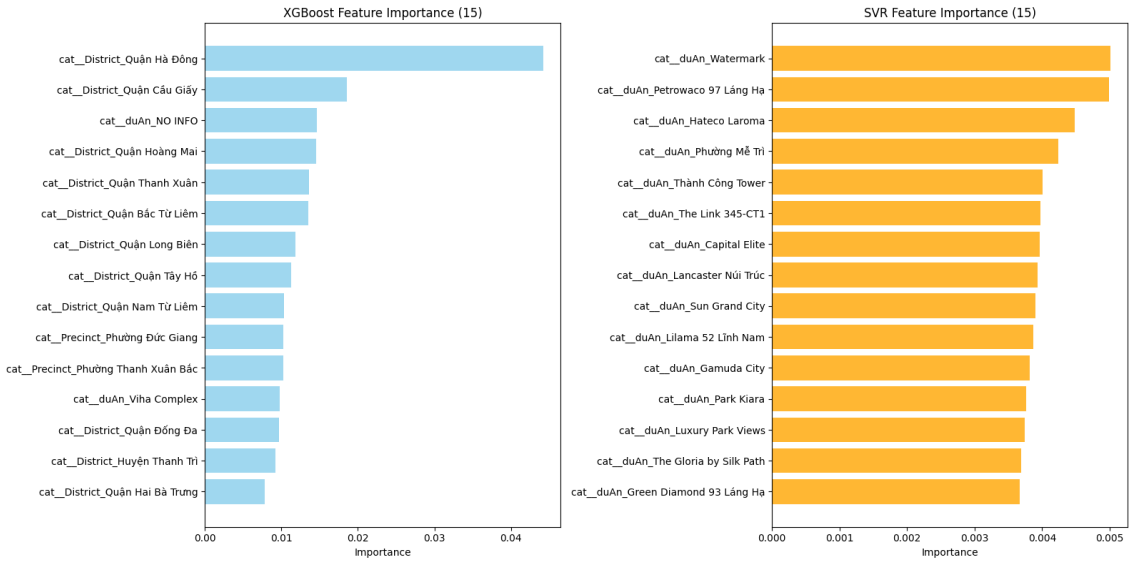

Figure 4.1: Feature Importance evaluated by SVR and XGBoost

The distinction between XGBoost and SVR in feature selection highlights their differing approaches to learning from data. XGBoost, as a tree-based model, excels at identifying broad patterns across hierarchical data, making it effective in scenar-

ios where features like districts or wards play a dominant role. This makes XGBoost well-suited for capturing the macro-level relationships that influence property prices.

On the other hand, SVR, a kernel-based model, prioritizes specific and detailed features, such as project names, reflecting its ability to perform well in scenarios with high dimensionality and fine-grained data. By leveraging project-specific information, SVR can capture subtle variations in pricing that are not immediately apparent from broader geographic attributes.

This divergence indicates that XGBoost's approach is more generalized and interpretable, while SVR provides a more targeted focus on the micro-features of individual data points.

In summary, while XGBoost excels at identifying key macro-level features, SVR brings attention to specific, project-level attributes. These findings underscore the importance of tailoring feature selection strategies to align with the predictive capabilities of each model and the nature of the dataset.

4.4 Demo Interface

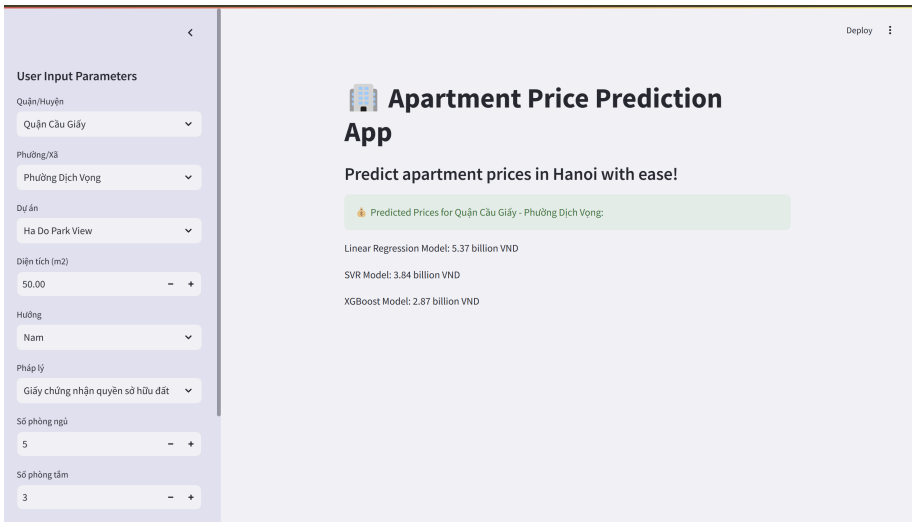

Figure 4.2: Hanoi Apartment Price Prediction Interface

The interface is built using Streamlit for an "Apartment Price Prediction App." It allows users to predict the price of apartments in Hanoi using various models such as Linear Regression, SVR, and XGBoost. The interface includes the following features:

1. Input Methods
   (a) Dropdowns for Categorical Inputs:

- Quận/Huyện (District): A dropdown menu where users select the district in Hanoi. This is a required field, as the application cannot provide predictions without a district.
- Phường/Xã (Precinct): A dropdown for selecting the precinct. This field can be left empty, as it is not marked mandatory. It follows the District provided above.
- Dự án (Project): A dropdown for selecting the real estate project. This is optional; if left blank, the prediction will not be tied to a specific project. It follows the District provided above.

(b) Number Input Fields:

- Diện tích (m²) - Apartment Size: Users can input the size of the apartment in square meters using a number input field. A default value of 50.00 is provided for convenience.
- Số phòng ngủ (Number of Bedrooms): Users can increment or decrement the value using plus (+) and minus (-) buttons. If the value is -1, it indicates that the number of bedrooms is unknown.
- Số phòng tắm (Number of Bathrooms): Similar to bedrooms, the default value is -1, meaning the number of bathrooms is unknown.
- Số lầu (Number of Floors): Users can also use increment/decrement buttons. A value of -1 implies the number of floors is not known.

(c) Dropdown for Additional Information:

- Hướng (Direction): Users can select the direction the apartment faces (e.g., North, South). This is optional, and leaving it blank implies no preference or unknown.
- Pháp lý (Legal Status): Allows users to specify the legal status of the apartment. This is also optional.

2. Results Section

- Displays the predicted price for the selected district.
- Shows predictions from three models: Linear Regression, SVR, and XGBoost.

CHAPTER 5. CONCLUSIONS

5.1 Summary

In summary, with the given topic, we have implemented methods to build a data science pipeline applicable to real-world problems. Specifically, the problem of predicting apartment prices in Hanoi has been addressed using foundational data science techniques and machine learning models. Data preprocessing, exploratory data analysis, feature engineering and hyperparameter tuning were applied to build a model with high predictive accuracy. In our analysis, the XGBoost model and LSTM model achieved the best performances on each metrics, demonstrating that data science methods can effectively solve practical problems like apartment price prediction.

Using the XGBoost method combined with hyperparameter tuning via Grid Search, we developed a model with RMSE down to 0.7347. While, LSTM got the best score on MAE at 0.5087. However, it is evident that this performance can be further enhanced with more sophisticated techniques, in both data processing and models.

Additionally, we integrated these predictive models into a user-friendly platform or application that can assist real estate agents, buyers, and investors in making data-driven decisions. This tool could include a GUI for users to input parameters and receive instant price predictions, thereby democratizing access to insights from data science and enhancing the real estate market's efficiency.

5.2 Suggestions for Future Work

In the future, we plan to explore more advanced methods, such as crawling more data from different sources, handling a large number of features and deep learning models with architectures specifically designed for regression tasks, to improve the prediction accuracy of apartment prices.

Besides, we aim at better handling the outliers and anomalies to gain a better results that can be generalized as well. At the same time, complicated feature selection and dimensionality reduction methods should be used to reduce the computational complexity and costs.

REFERENCE

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[2] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016. arXiv: 1603.02754. [Online]. Available: http://arxiv.org/abs/1603.02754.

APPENDIX

A. EXPLORATORY DATA ANALYSIS

A.1 Remove Outliers

The method used to remove outliers is plotting box plot to detect the outliers, then 3.1 is applied to set the range in which any point outside is considered as noise. Note that sometimes we manually set the range based on plot to detect outliers if the result of 3.1 is not practical.

List of IQR range of each feature that had outliers removed:

- price_value: [0.1, 10.1425]
- acreage_value: [12.0, 156.0]
- no_bed: {1..5}
- soLau: {1..35}
- no_bath: {1..3}
- price_per_area: [14.505, 95.483]

Additionally, we apply the IQR method to identify and remove outliers in each feature, based on their corresponding price_value.

A.2 Univariate Analysis

```
...                                      duAn  Frequency  Percentage
0              Vinhomes Ocean Park Gia Lâm        349    1.399246
1                     Vinhomes Smart City        188    0.753749
2                           Goldmark City        140    0.561302
3                         Ecolife Capitol        125    0.501163
4                     Trung Hòa Nhân Chính        119    0.477107
..                                     ...        ...         ...
813  Nhà ở xã hội Kiến Hưng - Lucky House          1    0.004009
814            Khu đô thị ParkCity Hà Nội          1    0.004009
815                      Sun Grand City            1    0.004009
816                              Phố 8/3           1    0.004009
817                   B4 - B14 Kim Liên            1    0.004009

[818 rows x 3 columns]
```

Figure A.1: Distribution of duAn

	Street	Frequency	Percentage
0	Đường Tố Hữu	943	3.780771
1	Đường Phạm Văn Đồng	296	1.186753
2	Đường Phạm Hùng	287	1.150670
3	Đường Lê Văn Lương	240	0.962232
4	Đường Nguyễn Trãi	234	0.938177
..	...	...	...
575	Đường Nguyễn Thái Học	1	0.004009
576	Đường Bùi Thiện Ngộ	1	0.004009
577	Đường Quyết Thắng	1	0.004009
578	Đường An Hưng 3	1	0.004009
579	Đường Quốc lộ 6	1	0.004009

[580 rows x 3 columns]

Figure A.2: Distribution of Street

	Precinct	Frequency	Percentage
0	Phường Mỹ Đình 2	641	2.569962
1	Phường Hoàng Liệt	537	2.152995
2	Phường Tây Mỗ	532	2.132948
3	Phường Trung Hòa	531	2.128939
4	Phường Trung Văn	526	2.108893
..	...	...	...
186	Phường Thanh Lương	1	0.004009
187	Phường quang trung	1	0.004009
188	Phường Nhật Tân	1	0.004009
189	Phường Láng Hạ	1	0.004009
190	Phường Liên Mạc	1	0.004009

[191 rows x 3 columns]

Figure A.3: Distribution of Precinct

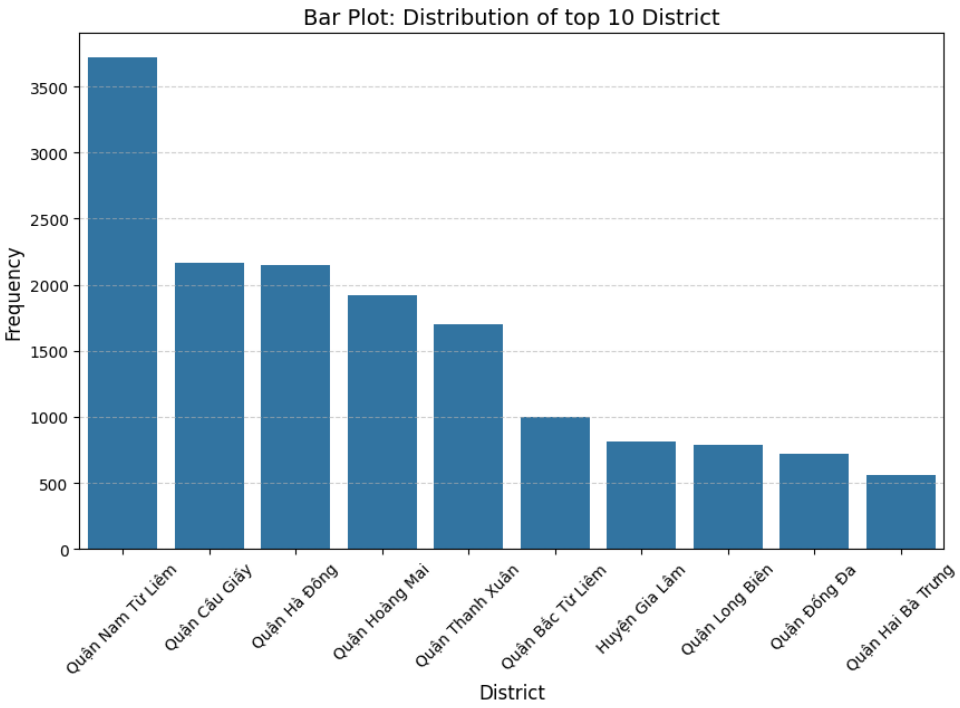

Figure A.4: Distribution of top 10 District

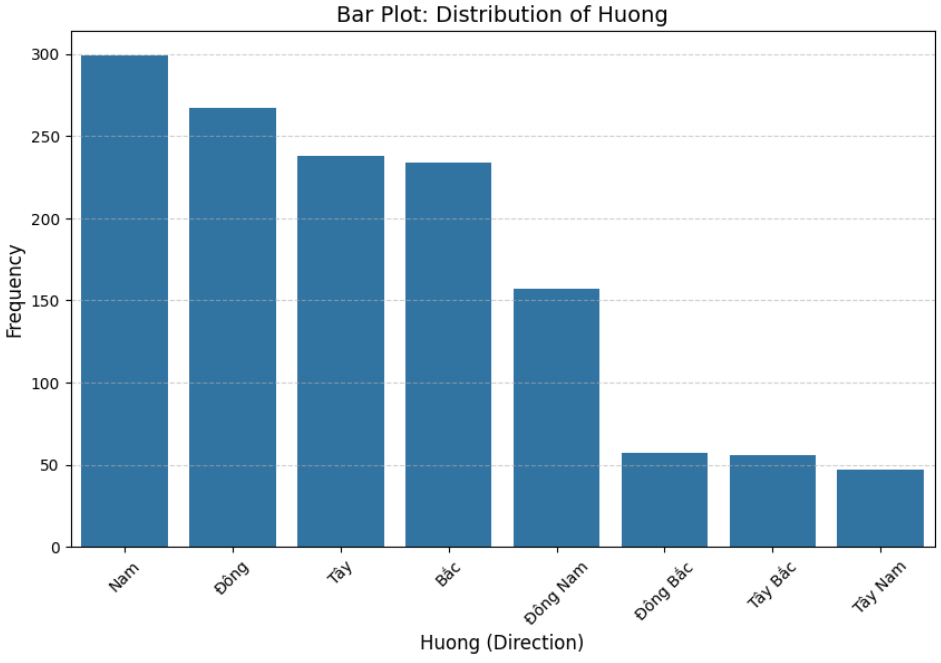

Figure A.5: Distribution of huong (Direction)

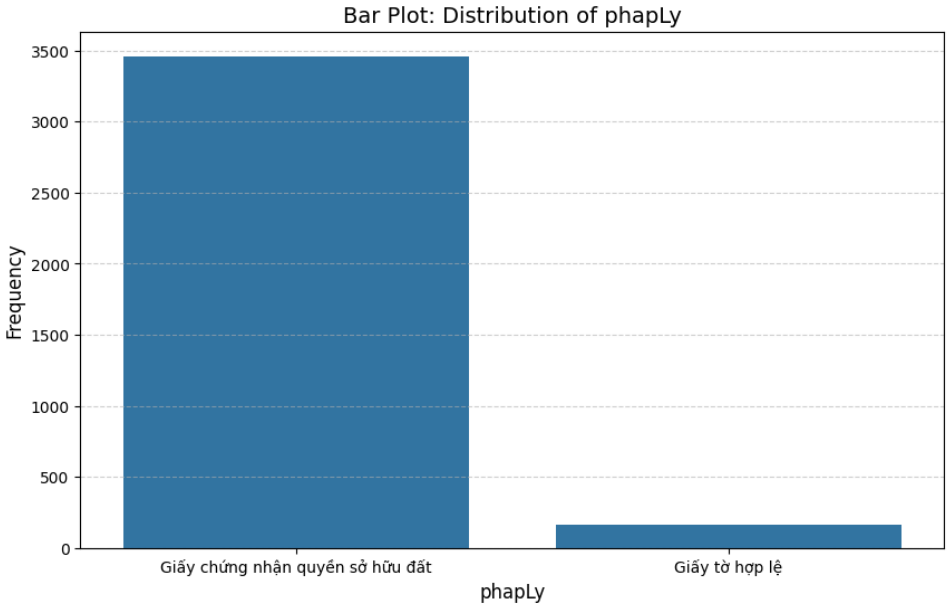

Figure A.6: Distribution of phapLy

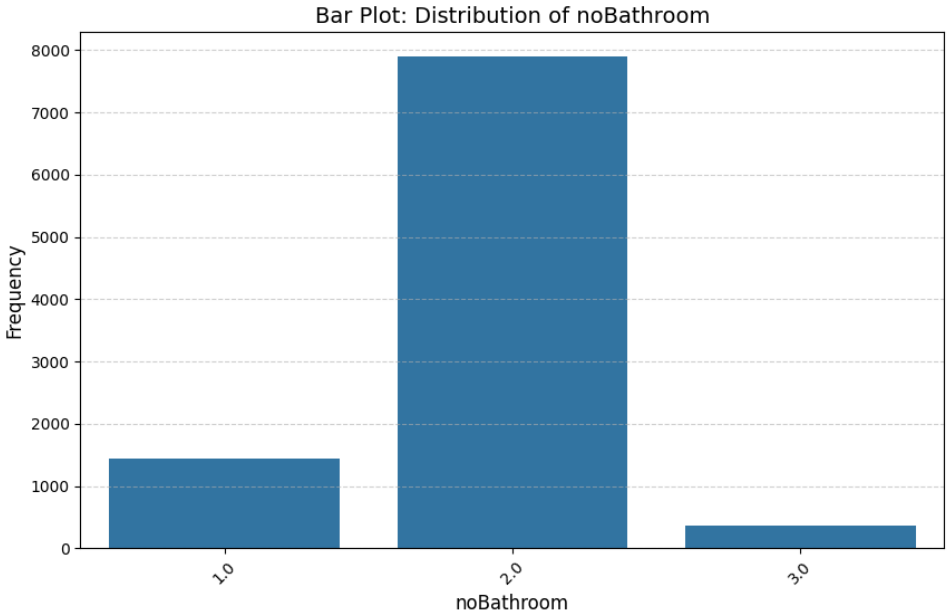

Figure A.7: Distribution of noBed

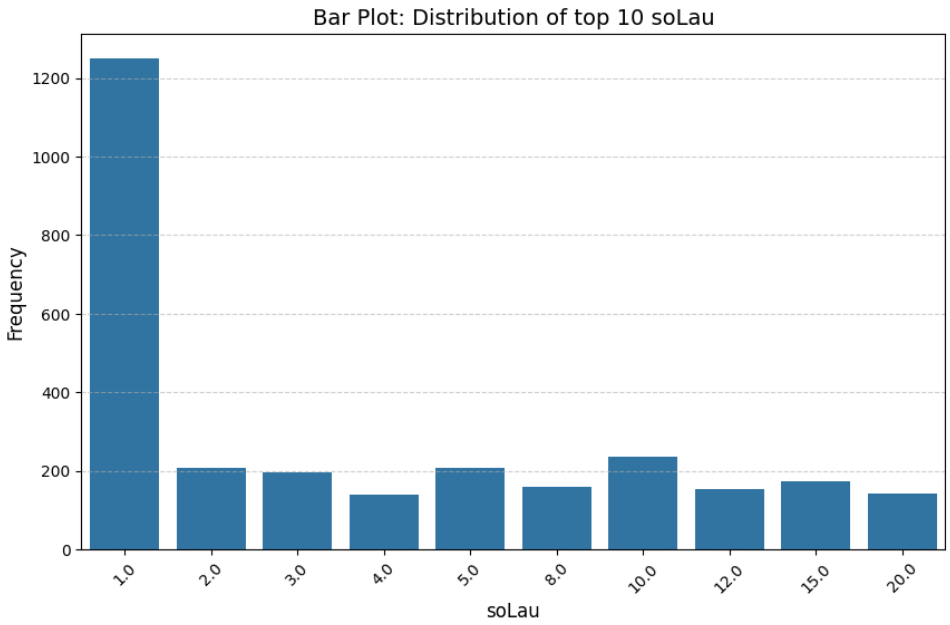

Figure A.8: Distribution of top 10 Number of Floor

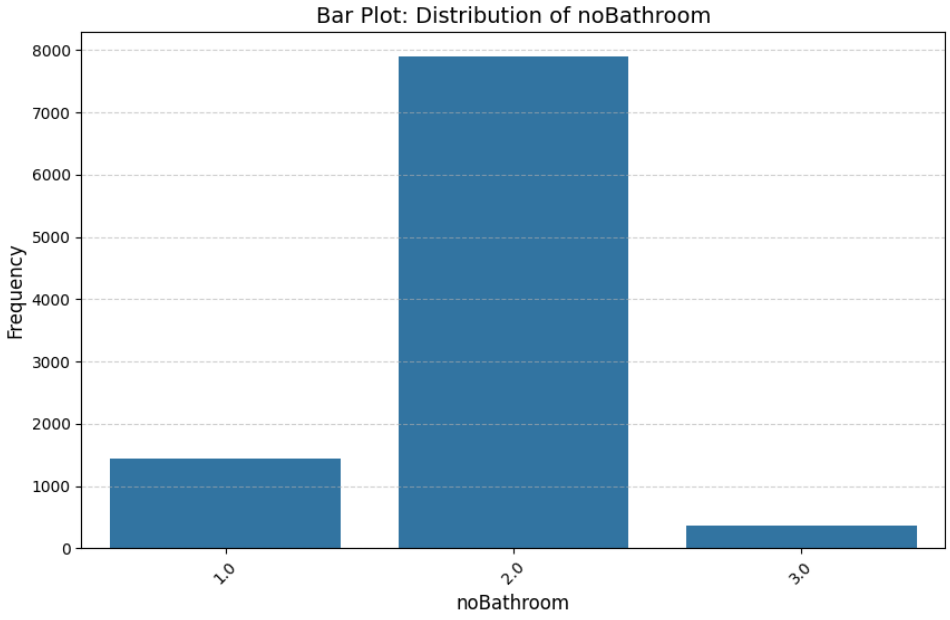

Figure A.9: Distribution of noBathroom

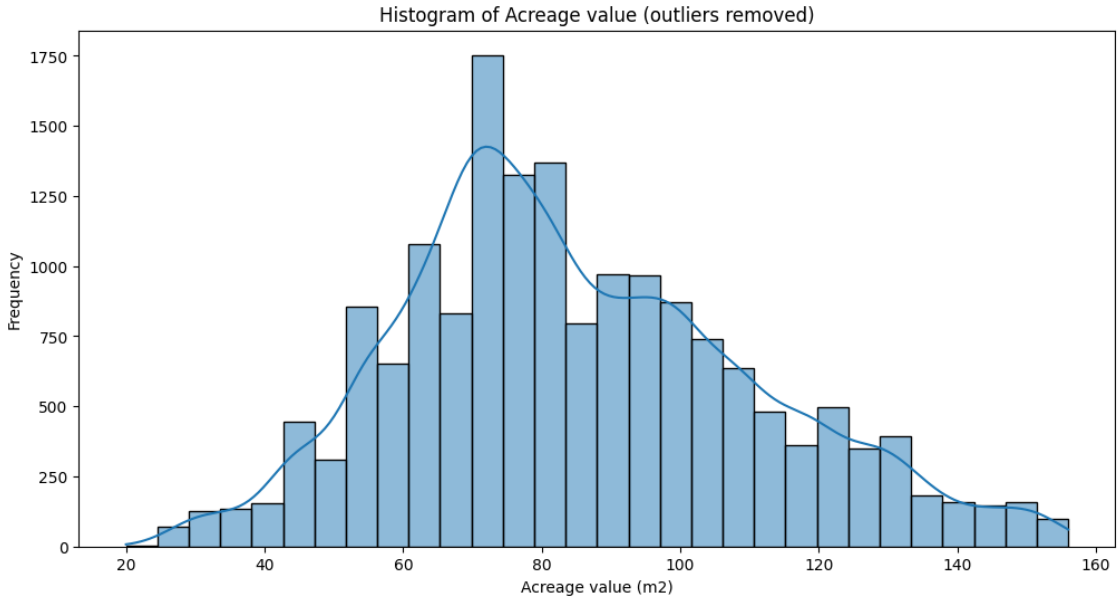

Figure A.10: Distribution of acreage_value

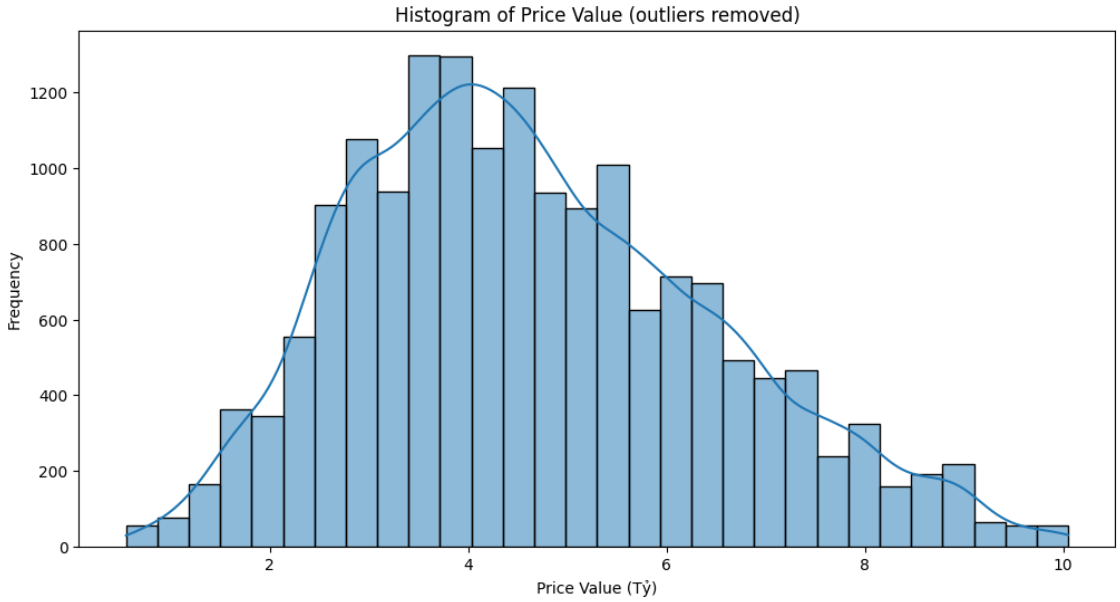

Figure A.11: Distribution of price_value

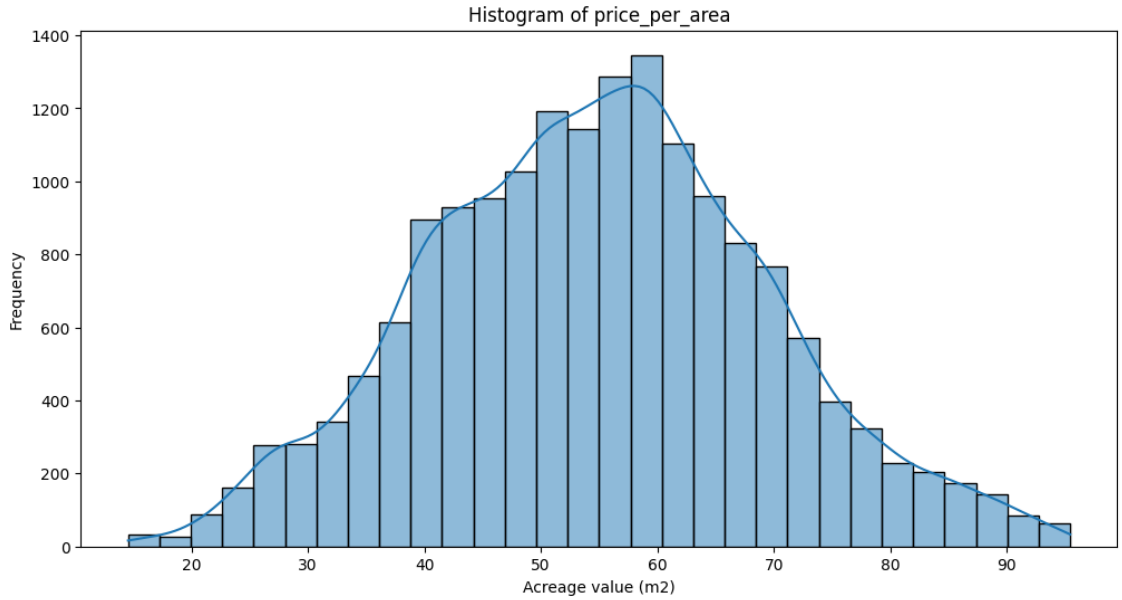

Figure A.12: Distribution of price_per_area

A.3 Bivariate Analysis

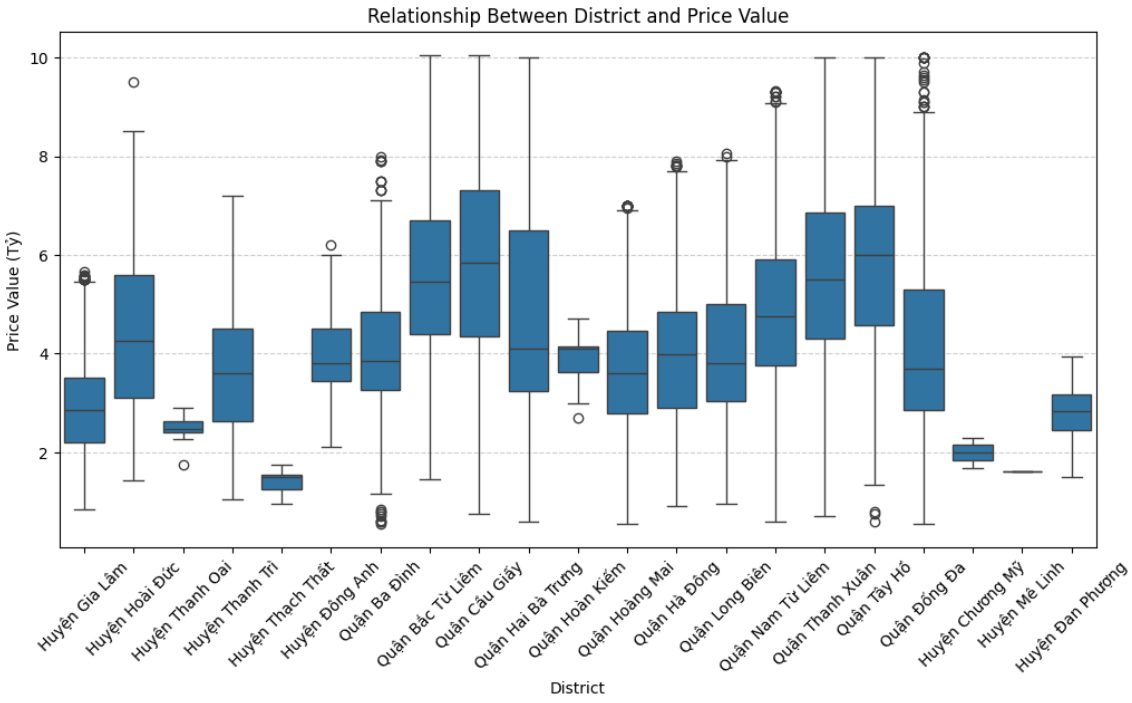

Figure A.13: Relationship between District and price_value

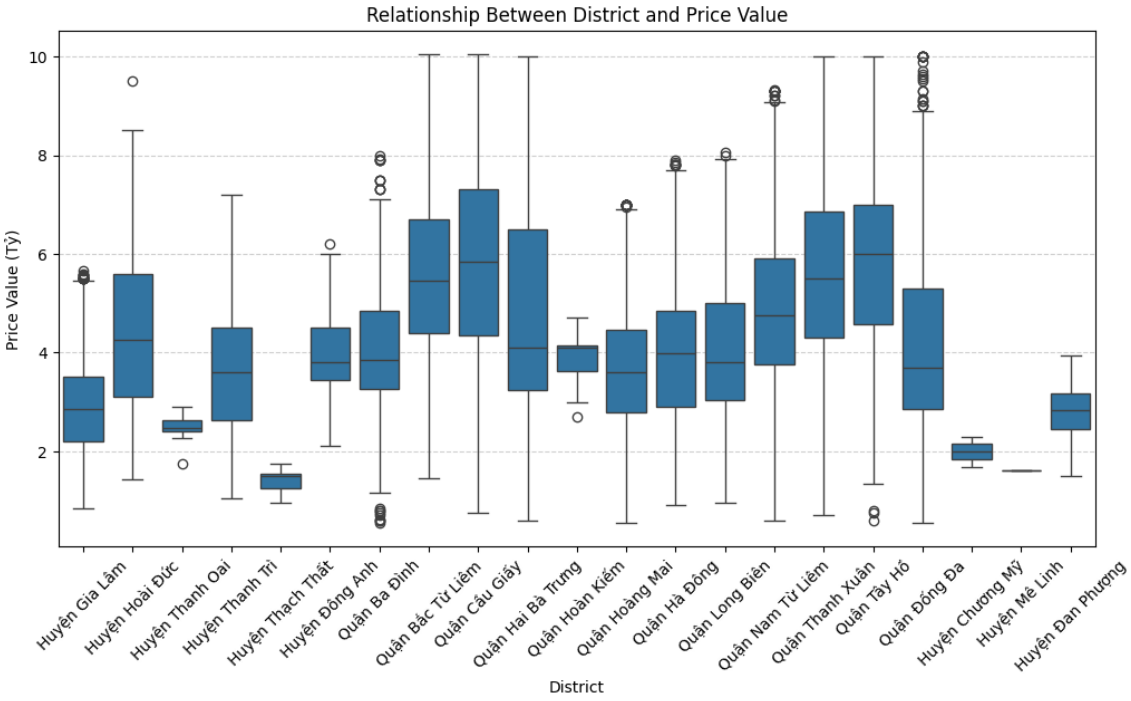

Figure A.14: Relationship between District and price_value

Figure A.15: Relationship between huong and price_value

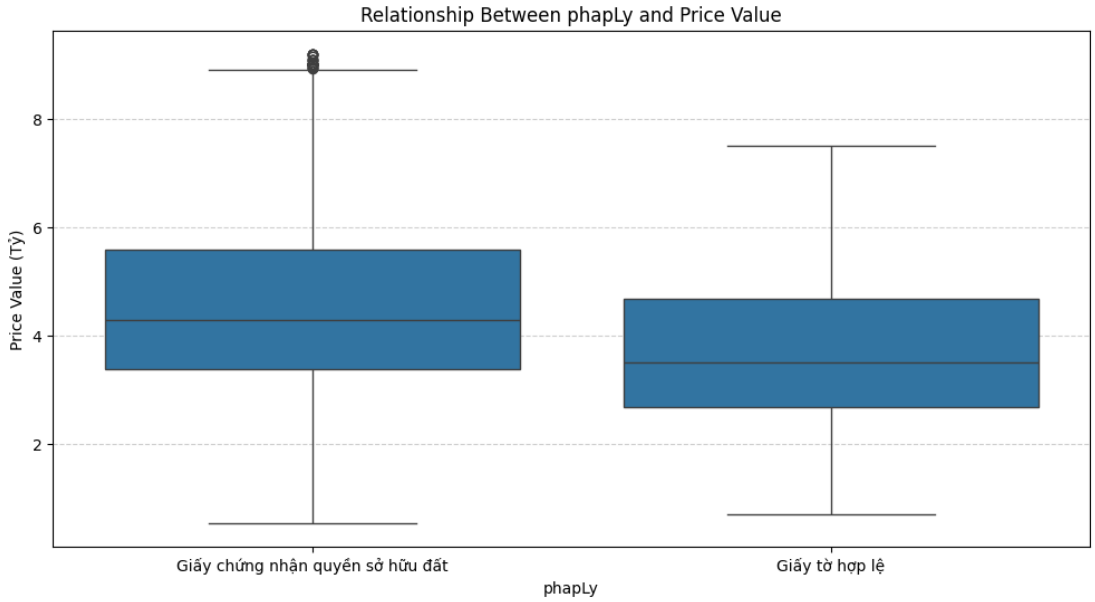

Figure A.16: Relationship between phapLy and price_value

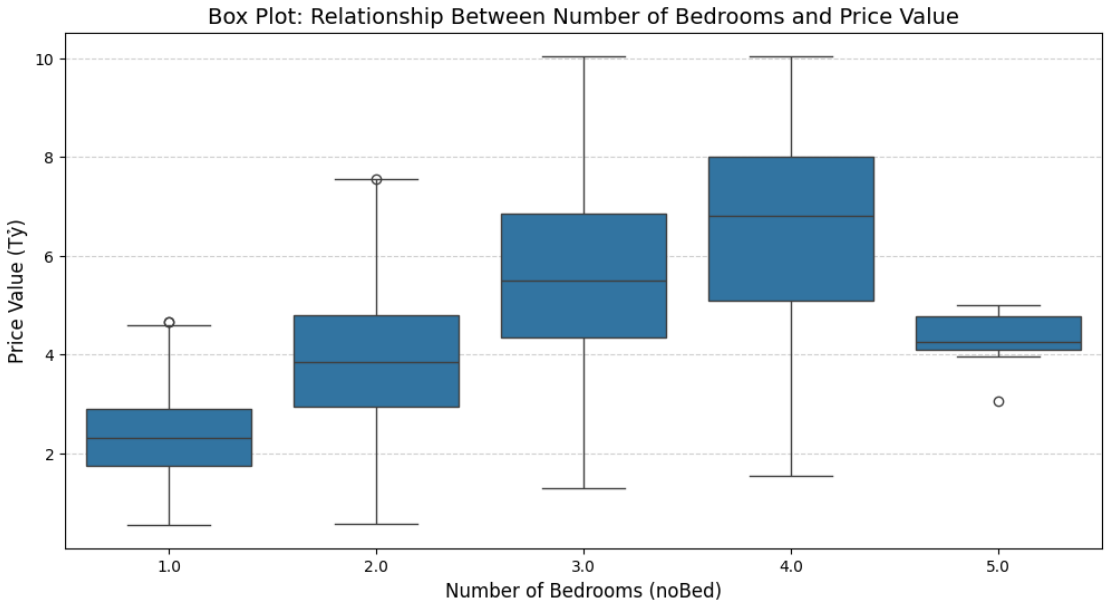

Figure A.17: Relationship between Number of Bedroom and price_value

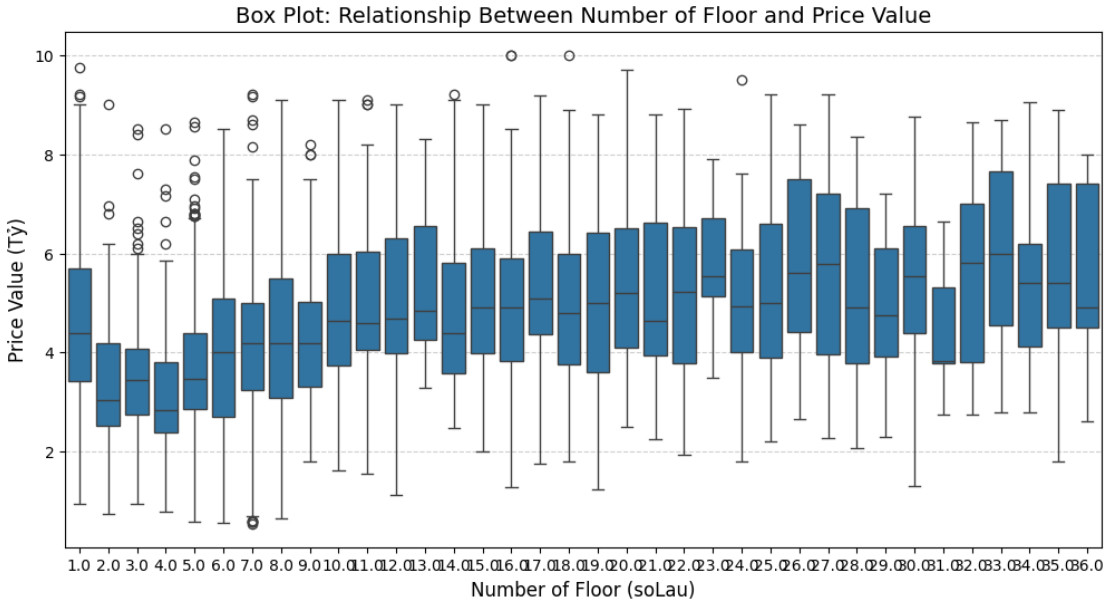

Figure A.18: Relationship between Number of Floor and price_value

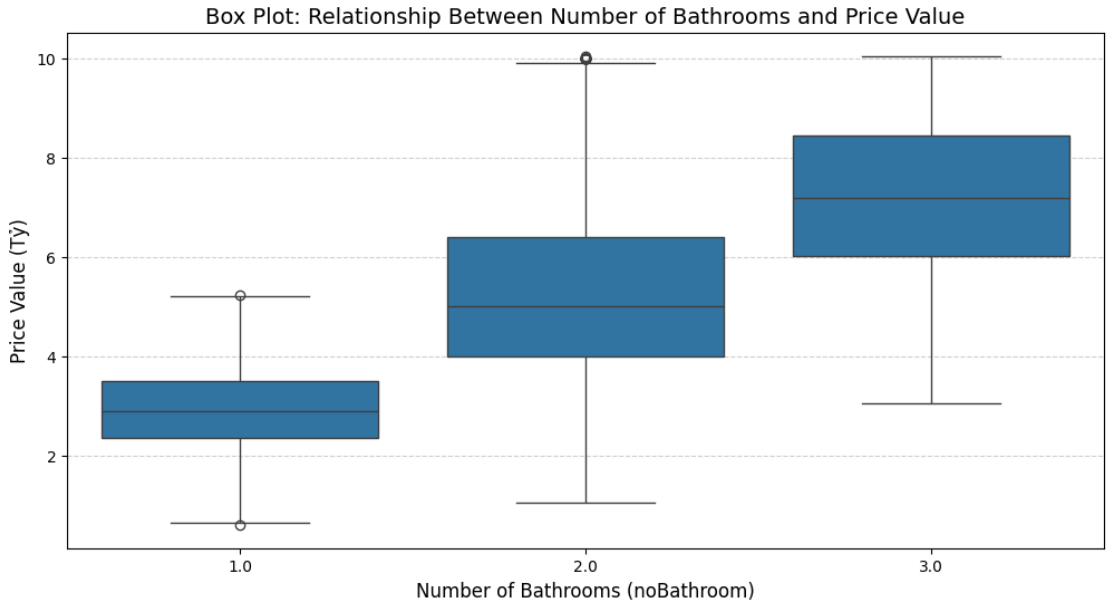

Figure A.19: Relationship between Number of Bathroom and price_value

Figure A.20: Relationship between huong and price_value

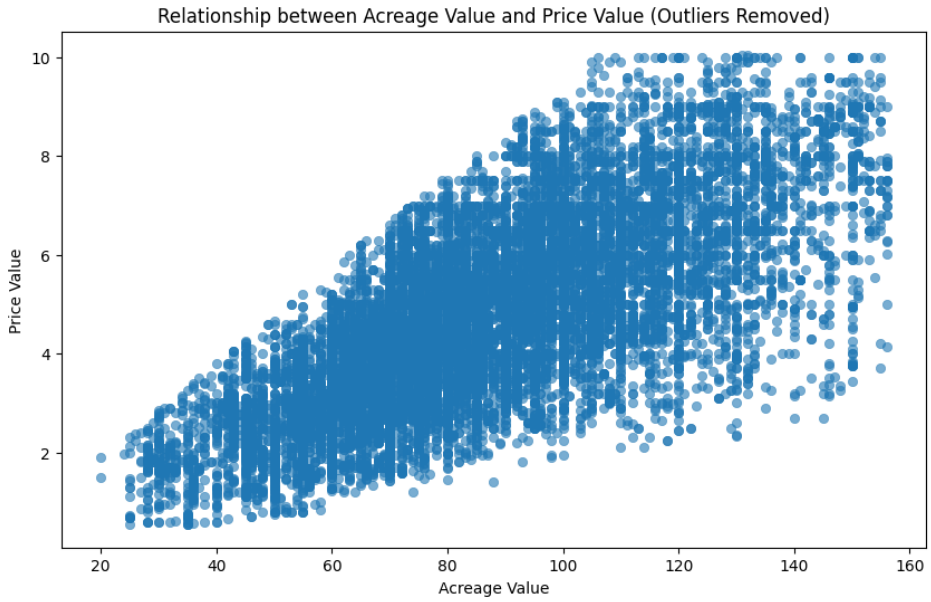

Figure A.21: Relationship between acreage_value and price_value

B. CONFIGURATION

The following Python libraries were utilized in this project to streamline the development process and enhance the performance of various components:

- **Beautiful Soup (bs4)**: Used for web scraping, particularly for parsing HTML and XML documents to extract relevant information.
- **Requests**: Facilitates HTTP requests for fetching data from web pages or APIs.
- **Streamlit (v1.41.1)**: A lightweight library for building interactive web applications for machine learning and data analysis.
- **Scikit-learn (v1.4.2)**: Provides a comprehensive suite of tools for machine learning, including models, preprocessing, and evaluation metrics.
- **XGBoost (v2.1.3)**: An efficient and scalable gradient-boosting framework optimized for both regression and classification tasks.
- **Joblib (v1.4.2)**: A library for lightweight pipelining and efficient serialization of Python objects, particularly for handling machine learning models and data preprocessing pipelines.
- **Keras (v2.12.0)**: A high-level neural networks API for building and training deep learning models, integrated with TensorFlow.
- **NumPy (v1.24.3)**: A library for numerical computations in Python, offering support for arrays, matrices, and various mathematical operations.
- **Pandas (v2.2.3)**: Used for data manipulation and analysis, providing powerful data structures like DataFrames and Series.
- **TensorFlow (v2.12.1)**: A widely-used framework for machine learning and deep learning, enabling both model development and deployment.

These libraries collectively supported the project's key functionalities, from data collection and preprocessing to model training and deployment.

C. LSTM TRAINING RESULTS

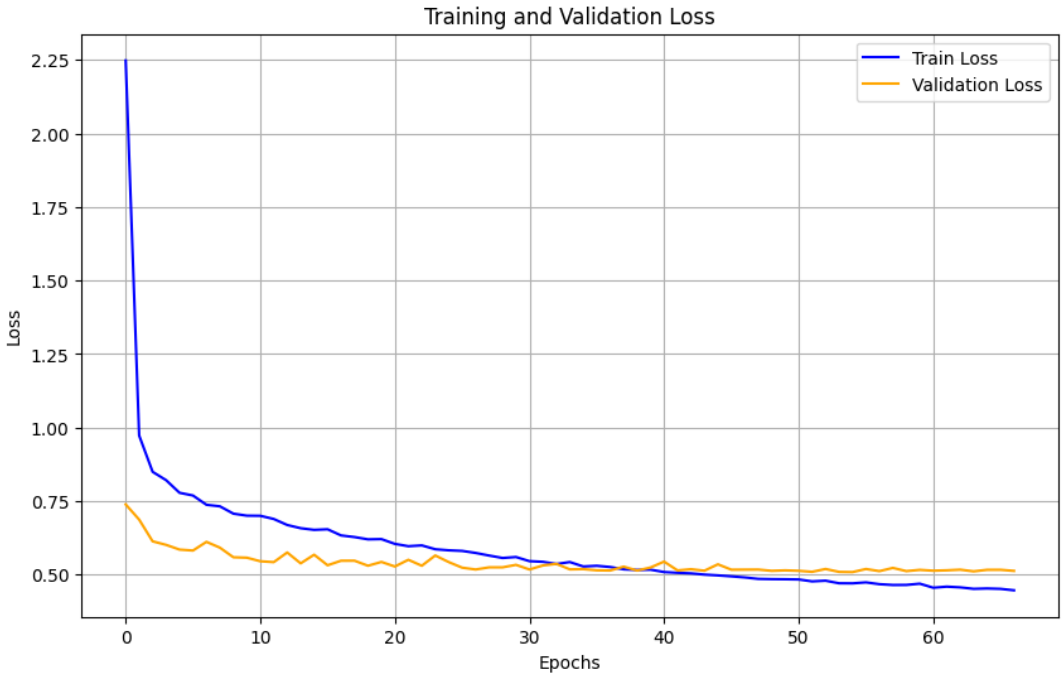

Figure C.1: LSTM Training Loss

Figure C.1 showed the LSTM training loss on train and validation sets. The model converged achieved its best validation loss after 65 epochs with early stopping at 12 patient epochs.