SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Data Science

COURSE: IT4142E

Real Estate Market

Trend Analysis

Author

Nguyen Tri Thanh	20225457
Nguyen Xuan Thanh	20225460
Pham Tuan Kiet	20214909
Le Nhat Quang	20225522
Phan Tran Viet Bach	20225435

Mentor

THAN QUANG KHOAT

November 2024

1. Project Overview

The real estate market is a cornerstone of modern economies, influencing both macroeconomic stability and individual wealth. Housing, as a fundamental human need, is particularly sensitive to market dynamics, and fluctuations in property prices can have profound social and economic consequences. In recent years, the real estate market has become a focal point of debate due to several persistent challenges, notably the **housing affordability crisis**, **housing price shocks**, and the complex process of **housing market recovery**. These issues are not only central to the well-being of individuals and families but also have significant implications for broader economic stability and urban development.

The **housing affordability crisis** has emerged as one of the most pressing issues in many global cities, particularly in regions characterized by rapid urbanization and population growth. The disparity between rising property prices and stagnant or declining wages has exacerbated the difficulty for middle- and low-income households to access affordable housing. In major metropolitan areas, the growing gap between income levels and home prices has led to increased rates of housing insecurity, social inequality, and displacement. This crisis has intensified the need for comprehensive analysis and predictive modeling of housing market trends, with the aim of understanding the underlying forces driving affordability challenges and identifying potential solutions.

Simultaneously, the real estate market is prone to sudden **housing price shocks**, which are often triggered by external economic factors such as interest rate hikes, inflationary pressures, or financial crises. These shocks can cause significant disruptions in the housing market, leading to sharp declines in property values and leaving homeowners and investors vulnerable to substantial financial losses. The COVID-19 pandemic, for instance, triggered a global economic shock that had profound effects on housing markets, with some regions experiencing unprecedented surges in property prices while others faced dramatic declines. The volatile nature of housing prices underscores the need for more sophisticated methods of forecasting and risk assessment to anticipate such shocks and mitigate their impact.

On the other hand, the process of **housing market recovery** after such shocks is complex and multifaceted. In the aftermath of a crisis, housing markets often go through extended periods of stagnation or slow recovery before returning to pre-crisis price levels. The pace and nature of this recovery are influenced by a variety of factors, including government intervention, economic growth, and shifts in demographic patterns. Understanding the dynamics of market recovery is crucial for policymakers and investors seeking to capitalize on opportunities while avoiding the risks associated with market volatility.

The objective of this study is to address these critical challenges by applying data science techniques to analyze and predict trends in the real estate market. By examining historical price data, economic indicators, and demographic trends, this research aims to develop a deeper understanding of the complex interactions that drive housing affordability, price volatility, and

recovery cycles. In particular, the study seeks to identify the primary factors contributing to housing price shocks, evaluate the impact of these shocks on affordability, and predict the trajectory of market recovery.

Given the cyclical and often unpredictable nature of the real estate market, traditional methods of analysis may not fully capture the dynamic forces at play. Therefore, this study will leverage advanced machine learning algorithms, time-series analysis, and econometric modeling to uncover hidden patterns and relationships within the data. The ultimate goal is to develop predictive models that can forecast housing market trends, offering valuable insights for policymakers, investors, and urban planners.

The key research questions driving this study are as follows:

1. **What are the key trends and patterns in the real estate market, particularly in relation to housing affordability and price volatility?**
2. **What are the primary factors influencing housing affordability and market price fluctuations?**
3. **How can machine learning models be used to forecast housing price shocks and recovery cycles?**
4. **What implications do these findings have for investment strategies?**

This study will employ a robust dataset comprising historical property prices, demographic data, and housing supply metrics. By applying advanced statistical and machine learning techniques, the research will explore the relationships between these variables, with a particular focus on identifying the drivers of housing price shocks and affordability issues.

The results of this analysis are expected to contribute significantly to the ongoing discourse on housing market stability and affordability. By using data-driven insights, this study will offer a more nuanced understanding of the factors that lead to housing market volatility and provide practical recommendations for addressing the affordability crisis and managing recovery phases more effectively.

The challenges of housing affordability, price volatility, and market recovery are central to the ongoing debates on urban development and economic policy. This study aims to provide a scientific and data-driven framework for understanding and forecasting real estate market trends, offering valuable insights that can guide decision-making and policy formulation in a time of unprecedented global economic uncertainty.

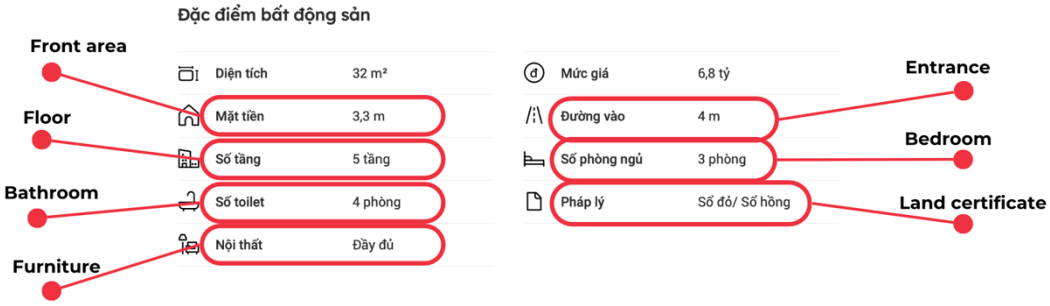
2. Data Collection

2.1. Data Crawling

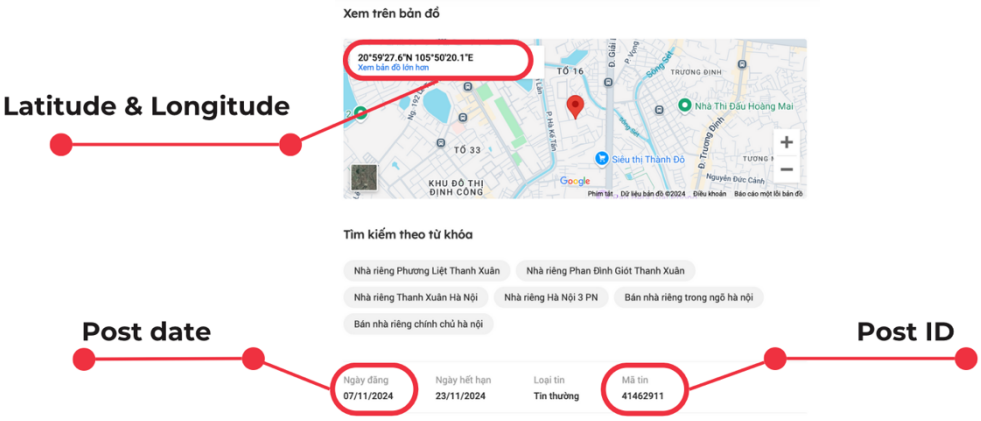
A property listing should include essential attributes that are described in the Figure below. Key attributes include Price, Area, and Address, with Price being the main variable for prediction. Listings missing these key details will be removed. Additionally, we collect supplementary

attributes such as ‘bedrooms’, ‘bathroom’, … to support the model development, along with several attributes like ‘latitude’, ‘longitude’, … for further analysis.

Essential Attributes



Supplementary Attributes



Additional Attributes

After research and survey, we identified eight reputable websites that offer a significant number of listing with the required attributes. We decide to utilize crawling and scraping on these sites. Each listing is stored as a record in CSV or JSON format.

In this project, we are using Selenium and Scrapy. They are both powerful tools for web crawling and scraping with their own pros and cons. Selenium is suitable for dynamic websites, easy to set up, and can simulate human actions. However, it process time is slow and not ideal for large-scale data extraction. Scrapy, on the other hand, is efficient for large data volumes and has

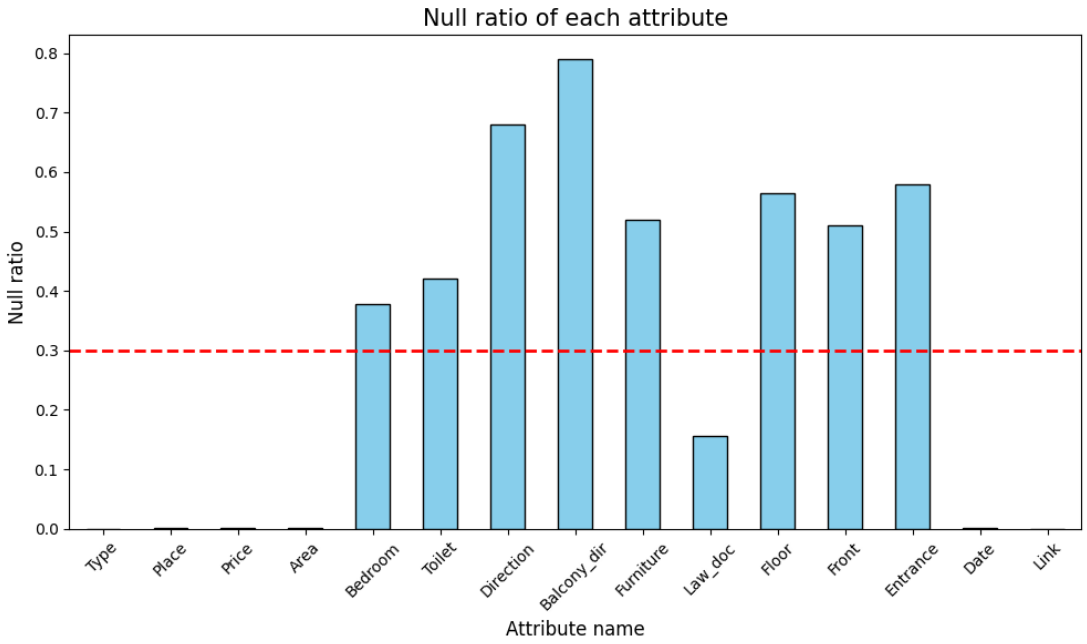
faster processing speeds, but it is more likely to be blocked by websites and more suitable for static sites.

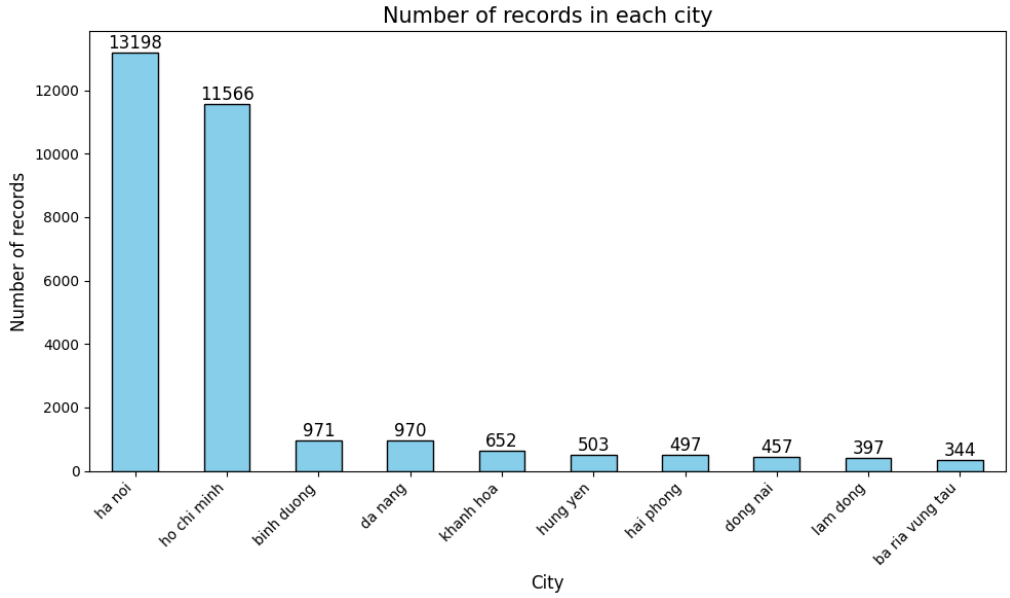
We have implemented Scrapy across all 8 websites initially due to its fast processing speed. After testing, we found that two websites allowed scraping a large amount of data with fully detailed listings. On the other hand, two websites did not allow Scrapy to function, two websites required CAPTCHA verification after scraping some records, and two did not provide attributes adequately. The detailed results are shown in Table below.

Website	#pages crawled	Notes
cafeland.vn	36063	
rever.com	11693	
muaban.net	2067	Not satisfy requirements
nhadat24h.net	1147	Not satisfy requirements
sosanhnha.com	632	Captcha
alonhadat.com	458	Captcha
batdongsan.com.vn	0	Error 403
nhatot.com	0	Error 403

Among the websites we surveyed, batdongsan.com.vn is considered one of the largest real estate websites in Vietnam, with approximately 200,000 available records. However, as mentioned above, this website does not allow Scrapy, and also requires verification via a checkbox CAPTCHA, as well as additional dynamic attributes such as location. Therefore, we continued to implement Selenium, another tool better suited for this website.

Collecting a single listing with Selenium takes about 10 seconds, which speed is much slower than Scrapy. We successfully gathered 35,000 records from this website. A preliminary analysis revealed that the dataset had a high rate of missing values, an uneven distribution of data across provinces/cities, short period of time, and inconsistent data formats, which could lead to significant challenges in future stages.



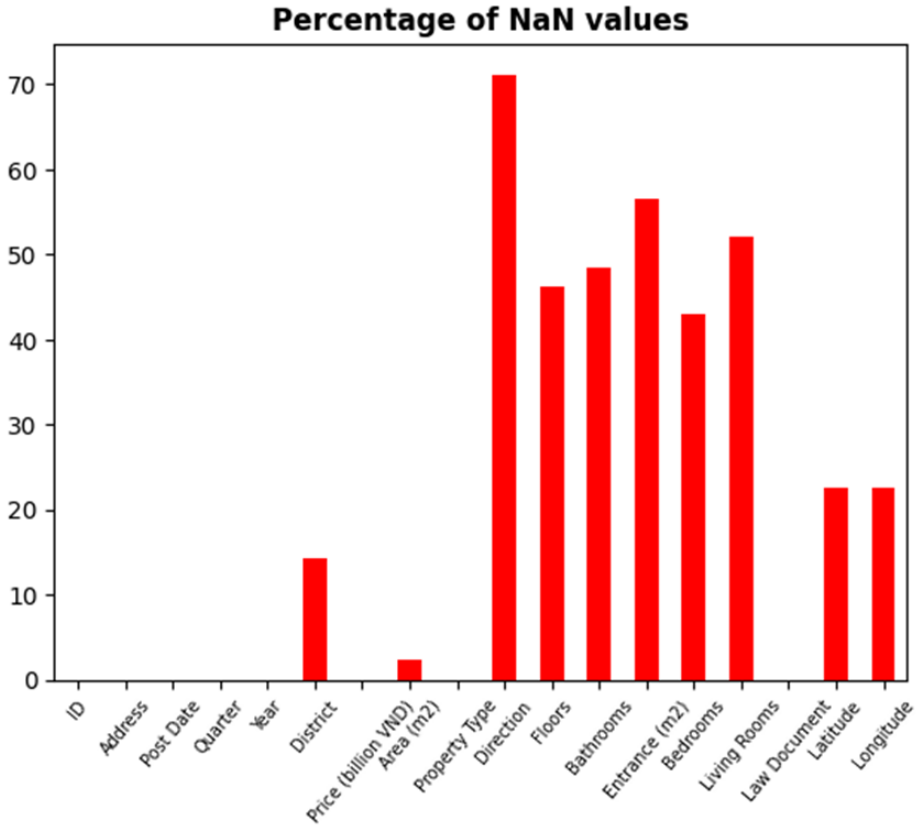


So in this project, we only use the data that collected with Scrapy from 2 websites ***cafeland.vn*** and ***rever.com***. After completing the data collection process, we obtained a dataset with over 40,000 records with sufficient attributes, focusing on two major cities in Vietnam: Hanoi and Ho Chi Minh City for further analysis.

2.2. Data Imputation and Preprocessing

As previously mentioned, our data is scraped and crawled through over 1000 pages from 2 popular real estate websites in Hanoi and Ho Chi Minh City (CafeLand and Rever). With duplicated data points removed, there is a total of 47,230 real estate of many kinds. The lion's share of this data comes from CafeLand, representing 76.3%.

Although paling in comparison, Rever's data is complete. On the other hand, CafeLand has several missing values. Most notably, the **Direction** field is later on dropped as 70% of the data is missing.



The original data dimensionalities range from 14 to 31. Since we merge data from the two sources, only common fields are chosen, namely **Price, Area, Bedrooms, Bathrooms, Address, Law Document, and Post Date, Latitude, and Longitude.** Out of 9 features, 7 need missing data handling, except for **Post Date and Law Document**. Any imputation or handling techniques must be implemented solely based on CafeLand's data, as we don't want noise from Rever to affect the distribution and nature of data in Hanoi. While exploring, we also discovered properties not in Hanoi but are purchased online in the Hanoi-only forum on the crawled web page. Those data are also removed.

One of the main purposes of this project is to design an ML/DL architecture for house price prediction. We decided to drop all data points not having their prices.

Thanks to free APIs from Google and MapQuest, information between a house address and its coordinates is interchangeable. We utilize them to make **Address, Latitude, and Longitude** complete. The other 3 fields **(Area, Bathrooms, and Bedrooms)** are imputed using Multivariate Imputation by Chained Equations (MICE) and Predictive Mean Matching (PMM). The following subsections will discuss these topics in more in-depth.

2.2.1. Types of Missing Data

Missing data mechanisms are typically classified as one of the following *(Rubin 1976)*:

- **MCAR:** Missing Completely At Random
- **MAR:** Missing At Random
- **MNAR:** Missing Not At Random

2.2.1.1. Missing Completely At Random

- Missing data are MCAR if the probability of missing is independent of the data.
- The reason for missing values in the outcome or predictors has nothing to do with the data values themselves.

→ This is often regarded as a strong and often unrealistic assumption.

2.2.1.2. Missing At Random

- The fact that the data is missing is systematically related to the observed but not the unobserved data.
- Under MAR, how likely a value is to be missing can be estimated based on the non-missing data.

Example: *A registry examining depression may encounter data that are MAR if male participants are less likely to complete a survey about depression severity than female participants. That is, if the probability of the completion of the survey is related to their sex (which is fully observed) but not the severity of the depression, then the data may be regarded as MAR.*

→ The complete case analysis is biased. Still, proper accounting for the known factors (in the above example, sex) can produce unbiased results in analysis.

2.2.1.3. Missing Not At Random

- Missing data are MNAR if, even given all the observed information, the probability of missingness depends on the unobserved values themselves.
- The missingness is related to events or factors which are not measured by the researcher.

Example: *To extend the previous example, the depression registry may encounter data that are MNAR if participants with severe depression are more likely to refuse to complete the survey about depression severity.*

→ As with MAR data, complete case analysis of a dataset containing MNAR data may or may not result in bias. If the complete case analysis is biased, however, the fact that the sources of missing data are themselves unmeasured means that (in general), this issue cannot be addressed in analysis and the estimate of effect will likely be biased.

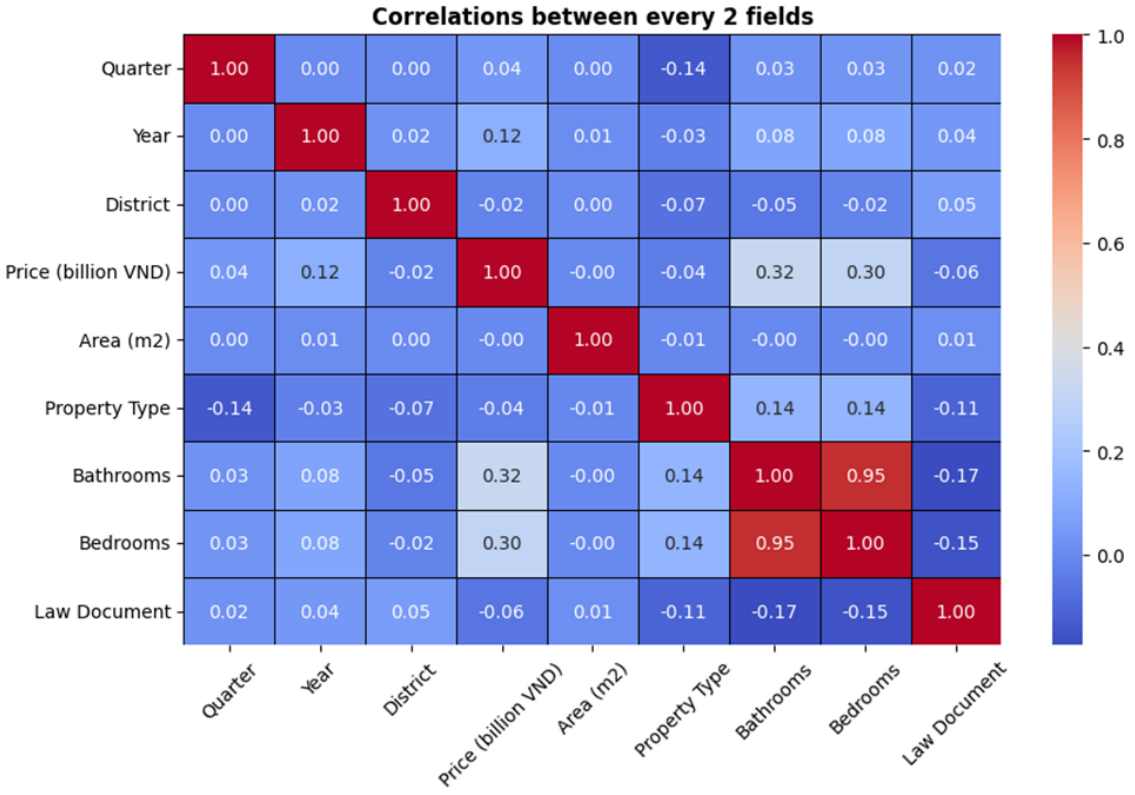
2.2.2. Imputations using Autoimpute

Autoimpute is a Python package for handling missing data with numerous imputation methods for continuous, categorical, and time series data by applying Single, Multiple, and MICE imputer classes. Below are the supported imputation techniques:

strategy	series-imputer
least squares	LeastSquaresImputer
stochastic	StochasticImputer
binary logistic	BinaryLogisticImputer
multinomial logistic	MultinomialLogisticImputer
bayesian least squares	BayesianLeastSquaresImputer
bayesian binary logistic	BayesianBinaryLogisticImputer
pmm	PMMImputer
lrd	LRDImputer
mean	MeanImputer
median	MedianImputer
mode	ModeImputer
random	RandomImputer
norm	NormImputer
categorical	CategoricalImputer
interpolate	InterpolateImputer
locf	LOCFImputer
nocb	NOCBImputer

The goal of imputation is to preserve the data structure (the correlation and covariance between features), not correctly guess the true value of the missing points. Therefore, it is only sensible if we consider each missing data to be a random variable and not a value that can be predicted. Multiple imputations are implemented in our project to be in line with this view. Now, this objective is under the assumption that our data is at least **Missing At Random (MAR).** If our data is **Missing Not At Random (MNAR),** then preservation may actually indicate poor imputation.

Before we move on to the details of each imputation method, here is the correlation matrix between features in our complete case dataset (a subset of the CafeLand's data). After imputation, we are going to review the structure of our data and see how our imputation works.



I have also calculated the VIF values for each field in the complete case dataset to detect multicollinearity between features and for later comparison.

VIF values (Before imputation – complete case dataset)	
Fields	**VIF**
Quarter	1.01
Year	1.03
District	1.01
Price (billion VND)	1.15
Area (m2)	1.00
Property Type	1.06
Bathrooms	11.29
Bedrooms	11.03
Law Document	1.06

Focus solely on the 3 fields to impute **(Area (m2), Bathrooms, and Bedrooms),** there is an extremely high correlation between **Bathrooms** and **Bedrooms.** This is also reflected in the severe multicollinearity present in these two fields (11.29 and 11.03, respectively).

For **Area (m2),** it is quite unusual for the correlations between it and fields like **Bathrooms** and **Bedrooms** to be close to zero since larger houses tend to have more bathrooms and bedrooms. This could be explained by the diverse types of properties and the variability in design standards across regions, and time periods.

2.2.2.1. Multivariate Imputation by Chained Equation (MICE)

In this procedure, a series of regression models are run whereby each variable with missing data is modeled conditional upon other variables in the data.

The chained equation process can be broken down into four general steps. For ease of understanding, an example is provided along with the steps described whenever necessary. Here, we have three features in our dataset: **Area (m2), Bathrooms,** and **Bedrooms**, all of which have missing values.

- **Step 1:** A simple imputation (mean imputation for example) is performed for every missing value in the dataset. These imputed values can be thought of as "placeholders". **Area (m2), Bathrooms,** and **Bedrooms** are mean imputed.

- **Step 2:** The "placeholders" for one variable are set back to missing. The imputed mean values of **Area (m2)** would be set back to missing.

- **Step 3:** The observed values from the variable in **Step 2** are regressed on the other variables in the imputation model, which may or may not consist of all of the variables in the dataset. These regression models operate under the same assumption made when one performs linear, logistic, or Poison regression models outside of the context of imputing missing data. A linear regression of **Area (m2)** predicted by **Bathrooms** and **Bedrooms** would be run using all cases where **Area** was observed.

- **Step 4:** The missing values of that variable are then replaced by the predictions from the regression model. Predictions of the missing **Area (m2)** values would be obtained from that regression equation and imputed. At this point, **Area (m2)** doesn't have any missingness.

- **Step 5: Steps 2-4** are then repeated for each variable that has missing data. The cycling through each of the variables constitutes one iteration or "cycle". At the end of one cycle, all of the missing values have been replaced with predictions from regressions that reflect the relationships observed in the data. **Steps 2-4** are repeated for **Bathrooms,** the only difference here is that both previously imputed and observed values of **Area (m2)** are inputs to the regression model.

- **Step 6: Steps 2-4** are repeated for a number of cycles, with the imputations being updated at each cycle. **Steps 2-4** would then again be repeated for the entire process of iterating through the three fields would be repeated until convergence.

Now we know how MICE works, we are fully equipped to understand why it fails at imputing **Area (m2), Bathrooms**¸and **Bedrooms** at the same time. MICE runs a series of regression models with assumptions about the data. One of the assumptions is the linear independence of the predictors. This is straight-out violated because **Bedrooms** and **Bathrooms** are highly correlated (with a 0.95 correlation), which explains our imputation strategy: using MICE to impute **Area (m2)** and **Bathrooms** first, then use PMM to imputed **Bedrooms** based on the complete **Bathrooms.**

2.2.2.2. Predictive Mean Matching (PMM)

Predictive Mean Matching borrows ideas from linear models to capture the correlation between features as well as strategies from hot-deck methods like K-Nearest Neighbour. This is a brief description of how PMM works:

- **Step 1:** Fit a linear regression on observed data. Keep the predictions and the regression coefficients handy.

- **Step 2:** Fit a Bayesian model on the observed data, using the coefficients from **Step 1** as inputs to the priors of the Bayesian model. Passing coefficients to priors is optional. We can use uninformative priors instead, but convergence may be much slower. If the linear model is poor, however, the coefficients may do more harm than good.

- **Step 3:** Make a random draw from the posterior predictive distribution of the coefficients to produce a new set of coefficients. This is a random draw using the means and the covariance matrix of the regression coefficients. The random draw creates sufficient variability and is the first step to generalize to multiple imputations.

- **Step 4:** Using the coefficients produced from the random draw, predict missing values for**y** from the corresponding features **x**.

- **Step 5:** For each predicted **y**, find the n closest predictions **y_pred**from the original observed linear regression

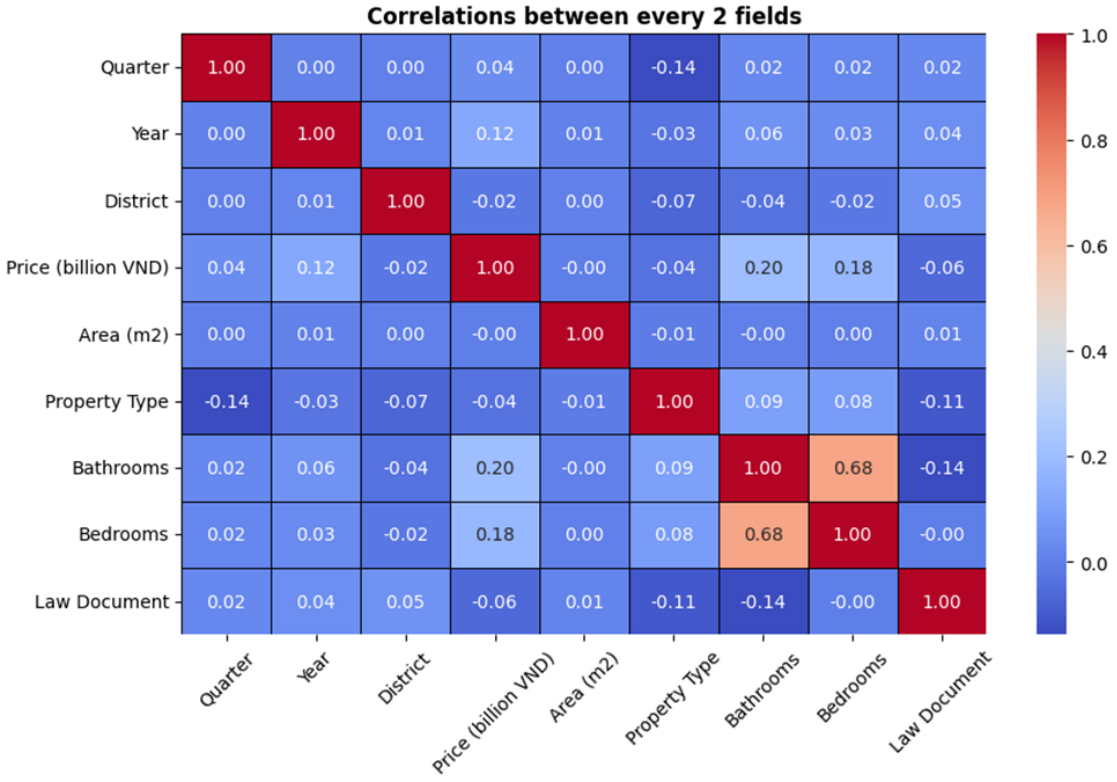
- **Step 6:** Take a random draw from the actual, observed**y** values that correspond to each **y_pred**, and impute.

Since the imputed values are real values that are “borrowed” from individuals with real data, we get realistic imputations, and we obey the covariance structure of the dataset. Additionally, we preserve the heteroskedasticity of the data. On the other hand, if the original variable is skewed, the imputed values will also be skewed.

The obvious danger of predictive mean matching is the duplication of the same donor value many times. This problem is more likely to occur if the sample is small, or if there are many more

missing data than observed data in a particular region of the predicted value. Such unbalanced regions are more likely if the proportion of incomplete cases is high, or if the imputation model contains variables that are very strongly related to the missingness.

2.2.3. Results



VIF values (After imputation)

Fields	**VIF**
Quarter	1.02
Year	1.02
District	1.01
Price (billion VND)	1.08
Area (m2)	1.00
Property Type	1.05
Bathrooms	1.96
Bedrooms	1.90
Law Document	1.05

Back to the correlation matrix and the VIF values of each field after imputations, there are 2 notable changes:

- The correlation between **Bathrooms** and **Bedrooms** decreases from 0.95 to 0.68. Missing values in **Area (m2)** and **Bathrooms** are imputed iteratively using MICE imputer. Therefore the imputed values for **Bathrooms** and **Area (m2)** depend partially on each other than ever before, which means that variability is introduced to the imputed data points. PMM uses the now completed **Bathrooms** as the predictor to impute **Bedrooms**, which propagates variability into the imputed bedrooms.
- Multicollinearity no longer exists for **Bathrooms** and **Bedrooms**. Since the correlation between **Bathrooms** and **Bedrooms** reduces, multicollinearity is also negated.

These changes can be considered to be positive as the relationship between **Bedrooms** and **Bathrooms** is more realistic and more interpretable. Multicollinearity is also reduced. Still, the usefulness of these features ultimately depends on their relationship with **Price (billion VND).**

3. EDA

3.1 Overview of the Dataset

The dataset comprises **36,116 records** with **16 attributes**, including critical features such as `Price (billion VND)`, `Area (m²)`, `Bedrooms`, `Bathrooms`, and geographical information. Following preprocessing steps, the dataset was reduced to **36,095 records** after outlier removal and handling of missing values.

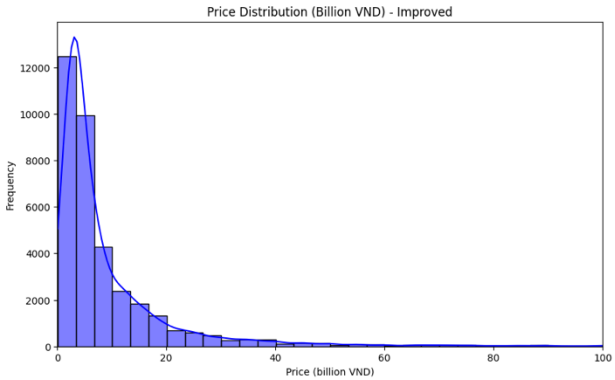
- **Key Features:**
  - Numerical: `Price (billion VND)`, `Area (m²)`, `Bedrooms`, `Bathrooms`, `Importance`, `Place Rank`.
  - Categorical: `Property Type`, `Law Document`, `City`.

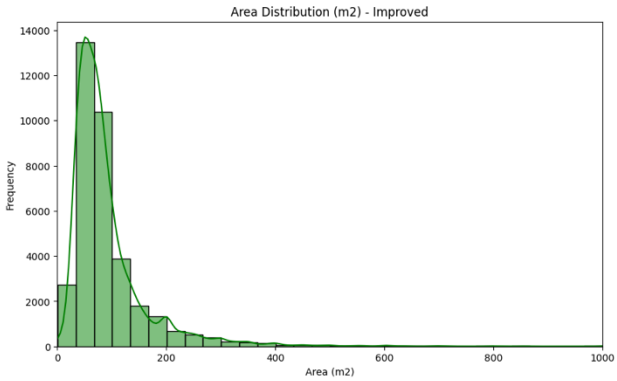
3.2 Data Distribution and Outlier Detection

The distributions of key numerical attributes (`Price` and `Area`) were explored:

- **Price Distribution:** Most properties are priced below **100 billion VND**, with a highly right-skewed distribution reflecting rare but high-value properties.
- **Area Distribution:** Majority of properties have areas below **1,000 m²**, also exhibiting a right-skewed trend.

Outlier detection and removal refined the dataset, ensuring representativeness for modeling purposes.

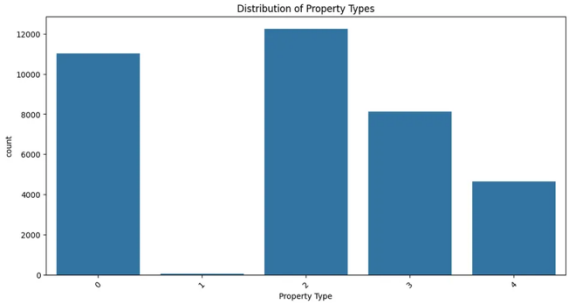


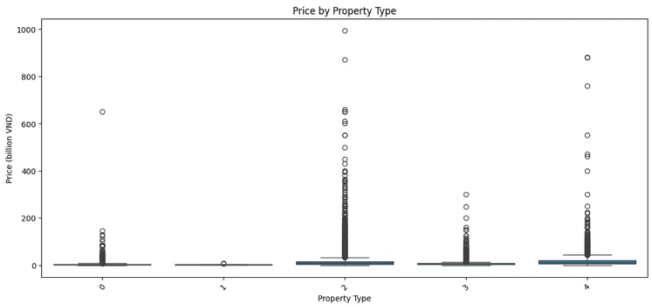


3.3 Property Type Analysis

The dataset includes several property types. Their distributions and relationships with prices were examined:

- **Distribution:** Property types 0 and 2 dominate the dataset, reflecting their prevalence in the real estate market.
- **Price by Property Type:** Boxplot analysis revealed significant variability in prices across different property types, with higher median prices observed for certain types.



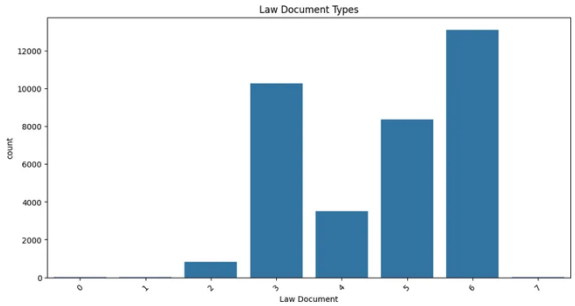


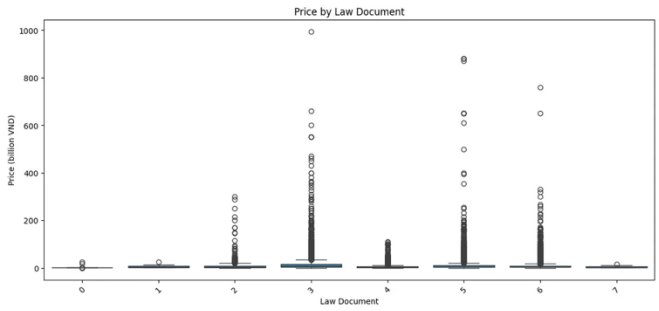
3.4 Pairwise Relationships with Price

3.4.1 Law Document Analysis

The `Law Document` feature, which reflects the legal status of properties, was analyzed:

- **Distribution:** The dataset is dominated by two categories (`5` and `6`), indicating common types of legal documents in Vietnam's real estate market.
- **Price by Law Document:** Boxplots indicate variation in prices, with properties associated with certain document types tending to have higher values.

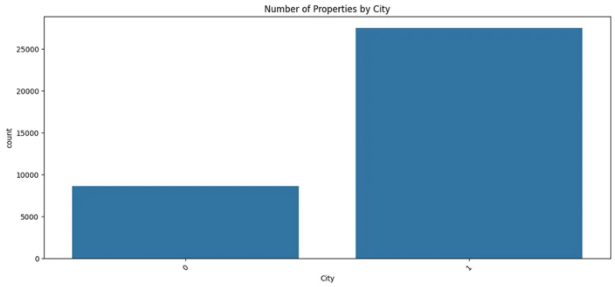


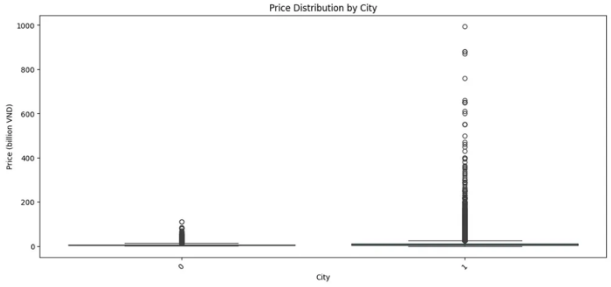


3.4.2 City-Level Insights

The dataset focuses on two major cities: Hanoi and Ho Chi Minh City:

- **Property Counts:** Ho Chi Minh City has significantly more listings, reflecting its larger market size.
- **Price Distribution:** While both cities exhibit right-skewed price distributions, Ho Chi Minh City displays greater variability, likely due to its diverse property market.





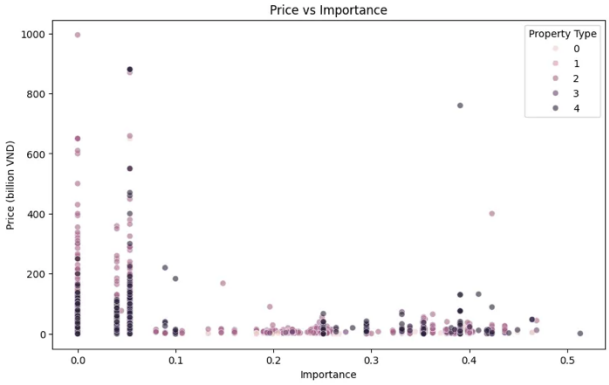
3.4.3 Relationship Between Importance and Place Rank

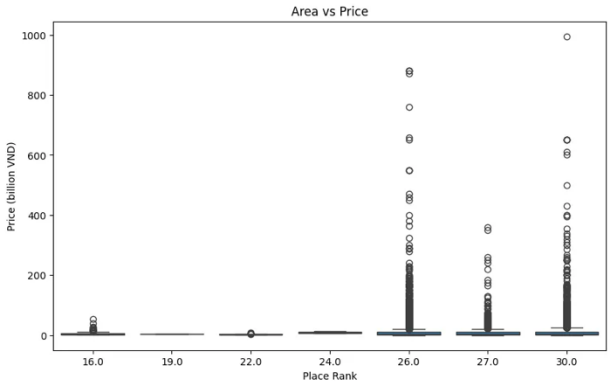
Two significant features, `Importance` and `Place Rank`, were analyzed:

- **Price vs. Importance:** Scatterplots show no strong linear relationship between `Importance` and `Price`. However, some property types show higher `Importance` values.
- **Place Rank vs. Price:** Boxplots reveal properties with higher `Place Rank` values tend to have greater price variability, emphasizing location as a key factor in pricing.

The relationship between importance scores and price is less pronounced. Although higher importance scores are occasionally associated with premium properties, the overall trend is weak, with most properties clustering around low importance scores. This suggests that while importance contributes to property value, its impact is overshadowed by more tangible features like area and location.

Place rank, on the other hand, demonstrates a clearer influence. Properties with higher place ranks tend to show greater price variability, underscoring the critical role of location in real estate valuation. Premium locations often command higher prices regardless of other property attributes, reinforcing the adage that "location is everything" in real estate.



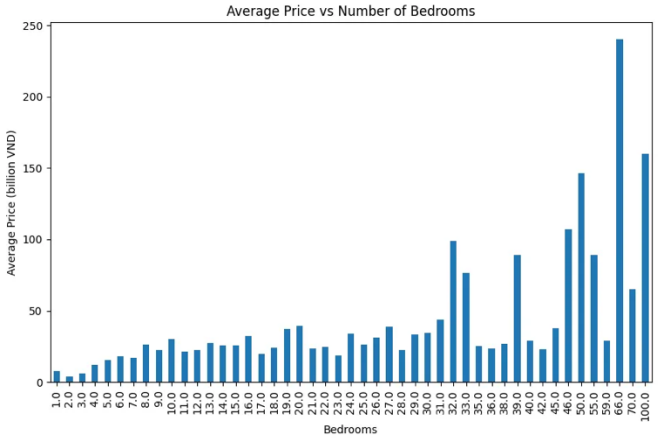


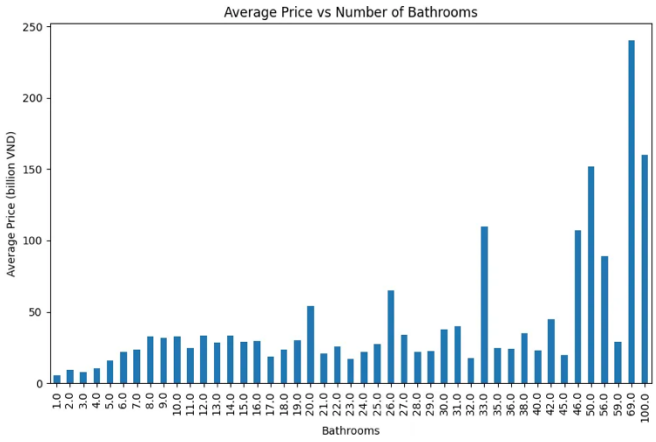
3.4.4 Bedrooms and Bathrooms Insights

The impact of property features such as `Bedrooms` and `Bathrooms` on prices was explored:

- **Average Price vs. Bedrooms:** As the number of bedrooms increases, the average price rises, although extreme values are observed for properties with unusually high bedroom counts.
- **Average Price vs. Bathrooms:** A similar trend is noted, with properties featuring more bathrooms generally commanding higher prices.

The number of bedrooms and bathrooms are strong indicators of property value. Properties with more bedrooms and bathrooms tend to have higher prices, reflecting their larger size and superior amenities. This trend is particularly noticeable in the median price progression for these features. However, the data also reveals properties with exceptionally high bedroom or bathroom counts, often exceeding 30, which likely belong to luxury or specialized real estate categories. These outliers highlight the existence of niche markets within the dataset.





3.4.5 Interaction Between Area and Price

The relationship between property area and price is a fundamental aspect of real estate analysis. In this dataset, the interaction between `Area (m²)` and `Price (billion VND)` reveals expected yet nuanced trends. Larger properties generally command higher prices, as evidenced by the positive correlation observed in the scatterplot. However, this relationship is not linear.

The data shows a concentration of properties with smaller areas, typically below 1,000 m², and prices under 200 billion VND. This clustering suggests that the market is dominated by smaller, more affordable properties, catering to a wider audience. Beyond this threshold, the rate of price increase diminishes, indicating diminishing returns for additional area. High-value outliers exist across all property types, representing luxury or commercial properties that diverge significantly from the typical market behavior. These findings highlight the segmentation of the real estate market into distinct categories, including affordable, mid-range, and premium segments.

3.5 Key Insights

The analysis of pairwise relationships reveals several important trends. Property size, represented by area, as well as the number of bedrooms and bathrooms, are the most consistent drivers of price. These features show strong positive relationships with property value, underscoring their importance in predictive modeling. Location-related features, such as place rank, further emphasize the importance of qualitative factors in real estate pricing, though their direct correlation is less pronounced.

Additionally, the presence of high-value outliers across all features points to unique market segments, including luxury and commercial properties, that deviate from the broader market trends. These insights provide a comprehensive understanding of the factors influencing property prices, serving as a foundation for the development of robust predictive models.

4. Machine Learning Models

4.1. Feature Engineering

Feature engineering is a critical component of the data preprocessing pipeline that aims to create new, informative features from raw data. This process enhances the predictive power of

machine learning models by transforming the available data into variables that better capture the underlying patterns influencing real estate prices. In this study, several key feature engineering techniques were employed to derive new features, each contributing to a more comprehensive representation of the property market. The following describes the specific features created, their rationale, and the formulas used to compute them.

4.1.1. Price per Square Meter

The **Price per Square Meter** is one of the most informative features in real estate market analysis. It normalizes the property price by the size of the property, making it easier to compare properties of varying sizes. Larger properties generally have higher total prices, but this metric helps us understand the price efficiency relative to size.

The formula for calculating **Price per Square Meter** is:

$$\text{Price\_per\_m2} = \frac{\text{Price (billion VND)}}{\text{Area (m}^2\text{)}}$$

Where:

- **Price (billion VND)** is the price of the property in billions of Vietnamese Dong.
- **Area (m²)** is the size of the property in square meters.

This feature is especially valuable when comparing properties of different sizes within the same location or property type. It helps determine whether the price of a property is reasonable relative to its size and can be useful for identifying market outliers.

4.1.2. Total Rooms and Room Ratio

The **Total Rooms** feature is derived from the sum of **Bedrooms** and **Bathrooms**. This feature reflects the overall scale of a property, which is a key determinant of its value. Larger properties with more rooms are generally more expensive due to their greater living capacity.

The formula for calculating **Total Rooms** is:

$$\text{Total\_Rooms} = \text{Bedrooms} + \text{Bathrooms}$$

Where:

- **Bedrooms** is the number of bedrooms in the property.
- **Bathrooms** is the number of bathrooms in the property.

In addition to **Total Rooms**, the **Room Ratio** was calculated as the ratio of **Bedrooms** to **Bathrooms**. This feature is important because it captures the balance between living space and functional space. A higher ratio of bedrooms to bathrooms may indicate an imbalance in the property's design, potentially reducing its desirability and, in turn, its price.

The formula for calculating the **Room Ratio** is:

$$\text{Room\_Ratio} = \frac{\text{Bedrooms}}{\text{Bathrooms}}$$

Where:

- **Bedrooms** is the number of bedrooms.
- **Bathrooms** is the number of bathrooms.

This ratio can highlight properties with an unusual number of bedrooms relative to bathrooms, which may be less attractive to buyers and could affect pricing trends.

4.1.3. Distance from Center

The **Distance from Center** feature quantifies the spatial relationship between a property and the approximate center of the geographic area under consideration. Location is one of the most important factors affecting real estate prices, as properties near urban centers or commercial hubs tend to have higher demand and therefore higher prices.

To calculate **Distance from Center**, the mean latitude and longitude of the properties in the dataset are used as the reference point for the center. The formula for calculating the **Distance from Center** (in degrees of latitude and longitude) is:

$$\text{Distance\_from\_center} = \sqrt{(\text{Latitude} - \text{mean\_lat})^2 + (\text{Longitude} - \text{mean\_lon})^2}$$

Where:

- **Latitude** and **Longitude** are the geographic coordinates of the property.
- **mean_lat** and **mean_lon** are the mean latitude and longitude of all properties in the dataset, representing the approximate center of the geographic area.

This feature captures the relative distance to the center of urban or commercial activity, which is often correlated with property prices. Properties located closer to urban centers generally exhibit higher prices due to their accessibility to amenities, employment opportunities, and transportation.

4.1.4. District Extraction from Address

The **District** feature provides a geographic categorization of each property based on its address. Districts are often strong indicators of property value, with properties located in more

affluent or high-demand districts typically commanding higher prices. To extract this information, the district name is parsed from the address string, assuming that the address contains this information.

If the address format includes the district, the district is extracted as follows:

$$\text{District} = \text{Address.split}(",")[1].\text{strip}()$$

Where:

- **Address** is the full address of the property.

The district feature enables segmentation of the dataset by location, allowing for the analysis of price variations across different regions. This can be particularly useful for identifying regional market trends and understanding how specific locations influence real estate prices.

4.1.5. One-Hot Encoding of Categorical Features

Categorical variables, such as **Property Type**, **Law Document**, and **City**, need to be transformed into a numerical format to be used in machine learning models. One-Hot Encoding is a widely used technique that converts each category into a binary vector, where each column represents a unique category of the variable. If a property belongs to a particular category, the corresponding binary value is set to 1; otherwise, it is set to 0.

For example, for the **Property Type** feature, if there are three categories ("Apartment," "House," and "Condo"), the encoding would produce the following columns:

- **Property Type_Apartment**
- **Property Type_House**
- **Property Type_Condo**

Each of these columns would contain a 1 or 0, depending on the property type. This transformation ensures that categorical variables can be used in machine learning models without introducing any ordinal relationships, which may not exist between the categories.

4.1.6. Outlier Treatment

Outliers can significantly affect the accuracy and robustness of predictive models, particularly when the data distribution is highly skewed. In this study, outlier treatment was performed using the **Interquartile Range (IQR)** method. This approach identifies and handles extreme values that deviate substantially from the central distribution of the data. The IQR is defined as the difference between the 75th percentile (Q3) and the 25th percentile (Q1) of the data:

$$IQR = Q3 - Q1$$

Values that fall outside the bounds defined by the following thresholds are considered outliers:

$$\text{Lower Bound} = Q1 - 1.5 \times IQR$$

$$\text{Upper Bound} = Q3 + 1.5 \times IQR$$

Properties with values outside of these bounds are treated either by capping (clipping) or by removal, depending on the context. This step ensures that extreme values do not disproportionately affect the model's ability to generalize.

Features above provide a more granular and insightful representation of the data, enabling more accurate predictive modeling:

- **Price per Square Meter** normalizes prices by size, offering a more comparable metric across properties of varying sizes.
- **Total Rooms** and **Room Ratio** reflect the functional aspects of a property, which can impact both its desirability and price.
- **Distance from Center** quantifies the importance of location, reflecting how proximity to urban or commercial hubs affects property value.
- **District** extraction helps capture the geographical variation in property values, segmenting properties by location-based pricing trends.
- **One-Hot Encoding** converts categorical variables into a machine-readable format, ensuring that attributes such as property type and location are appropriately considered in predictive models.
- **Outlier Treatment** ensures that extreme values do not skew the dataset, providing a cleaner input for machine learning algorithms.

4.2. Time Series

Forecasting real estate prices is a critical task for investors, policymakers, and financial analysts, requiring accurate predictive models that can handle various challenges inherent in time series data. These challenges include non-stationarity, seasonality, and complex trends. Traditional time series models like ARIMA and SARIMA are widely used for their ability to capture these patterns, while machine learning methods such as XGBoost and Prophet offer greater flexibility in modeling non-linear relationships and external factors.

This study develops a comprehensive framework for time series modeling, encompassing data preprocessing, stationarity checks, feature engineering, model building, and evaluation. The objective is to forecast real estate prices using multiple models, comparing their performance, and selecting the most suitable approach for accurate price prediction.

The time series modeling process begins with data cleaning and preparation. The dataset consists of historical real estate prices, where the target variable is the price of properties (in

billions of VND), and the independent variable is the post date. For time series models like ARIMA and SARIMA to yield valid results, the data must be stationary, meaning its statistical properties (mean, variance) do not change over time. The stationarity of the time series is assessed using the **Augmented Dickey-Fuller (ADF) test**.

- **Stationarity Check**: The ADF test provides a p-value to determine if a unit root is present in the series, indicating non-stationarity. A p-value below a significance level (alpha = 0.05) suggests that the series is stationary. If the p-value is above 0.05, the series is differenced to remove trends and achieve stationarity.
- **Differencing**: In cases where the series is non-stationary, differencing is applied iteratively until stationarity is achieved, or a maximum of two differences is reached. The differencing process removes trends and seasonal patterns, making the data suitable for ARIMA-based models.

Feature engineering plays a vital role in improving the performance of machine learning models. For models such as XGBoost, which rely on historical features, additional features are created from the original time series data.

- **Lagged Features**: Lag features capture the temporal dependencies in the data by using past values of the target variable. For instance, the previous month's price (`lag_1`) or the price 12 months prior (`lag_12`) are included as features for predictive modeling.
- **Rolling Statistics**: Features such as rolling mean and rolling standard deviation are computed over a window (e.g., 3 months) to capture short-term trends and variability in the series.
- **Seasonal Decomposition**: The time series is decomposed using **STL (Seasonal-Trend decomposition using LOESS)** to extract the underlying trend, seasonal, and residual components. These components are used as additional features, capturing both long-term trends and seasonal patterns.
- **Time-Based Features**: To capture seasonal effects, the framework generates time-based features such as month, quarter, and year, which may correlate with price fluctuations over time.

4.2.1 ARIMA Model

The **AutoRegressive Integrated Moving Average (ARIMA)** model is a classic approach for time series forecasting that combines autoregression, differencing, and moving averages. ARIMA requires the time series to be stationary, a condition that is ensured through the differencing process discussed earlier.

- **Model Selection**: The ARIMA model's parameters (p, d, q) are automatically selected using the `auto_arima` function, which applies a grid search to identify the optimal combination of parameters. The differenced series is used as input to train the model.
- **Forecasting**: After model training, forecasts are generated for the test period, and confidence intervals are estimated to capture uncertainty.

4.2.2 SARIMA Model

The **Seasonal ARIMA (SARIMA)** model extends ARIMA by incorporating seasonal components, making it particularly suitable for time series with recurring patterns (e.g., monthly or yearly seasonality).

- **Seasonality**: The SARIMA model uses the seasonal period `m` (e.g., 12 for monthly data) to capture seasonal fluctuations in price. The model parameters (p, d, q) and seasonal parameters (P, D, Q) are also selected automatically using `auto_arima`.
- **Box-Cox Transformation**: To stabilize the variance, a Box-Cox transformation can be applied before training the model.
- **Forecasting**: Similar to ARIMA, forecasts are generated with associated confidence intervals, providing insights into the predicted price range.

4.2.3 Prophet Model

The **Prophet** model is a forecasting tool developed by Facebook that is designed for time series data with strong seasonal effects and missing data. Prophet allows for flexible seasonality modeling, which can be particularly useful in real estate price forecasting where periodic market fluctuations are common.

- **Growth and Seasonality**: The Prophet model can incorporate yearly, weekly, and daily seasonalities, making it adaptable to different time series data characteristics.
- **Forecasting**: The model generates forecasts along with uncertainty intervals. It is particularly effective when the data exhibits irregular patterns or sudden shifts, such as policy changes or economic events.

4.2.4 XGBoost Model

The **XGBoost (Extreme Gradient Boosting)** model is a powerful ensemble method that combines the predictions of multiple decision trees. XGBoost can handle large datasets and is highly effective for non-linear time series data with complex relationships between features.

- **Feature Engineering**: XGBoost relies heavily on feature engineering, and thus, lagged features, rolling statistics, and seasonal components extracted from STL decomposition are used as input features.
- **Hyperparameter Tuning**: The model is tuned using **GridSearchCV** with time series cross-validation (`TimeSeriesSplit`) to select the best hyperparameters. The grid search evaluates multiple combinations of hyperparameters, such as the number of estimators, learning rate, and tree depth.

4.2.5 Model Comparison

Once all models are trained and evaluated, they are compared using common performance metrics to determine the most effective model for forecasting real estate prices.

- **Evaluation Metrics**: The following metrics are used for comparison:

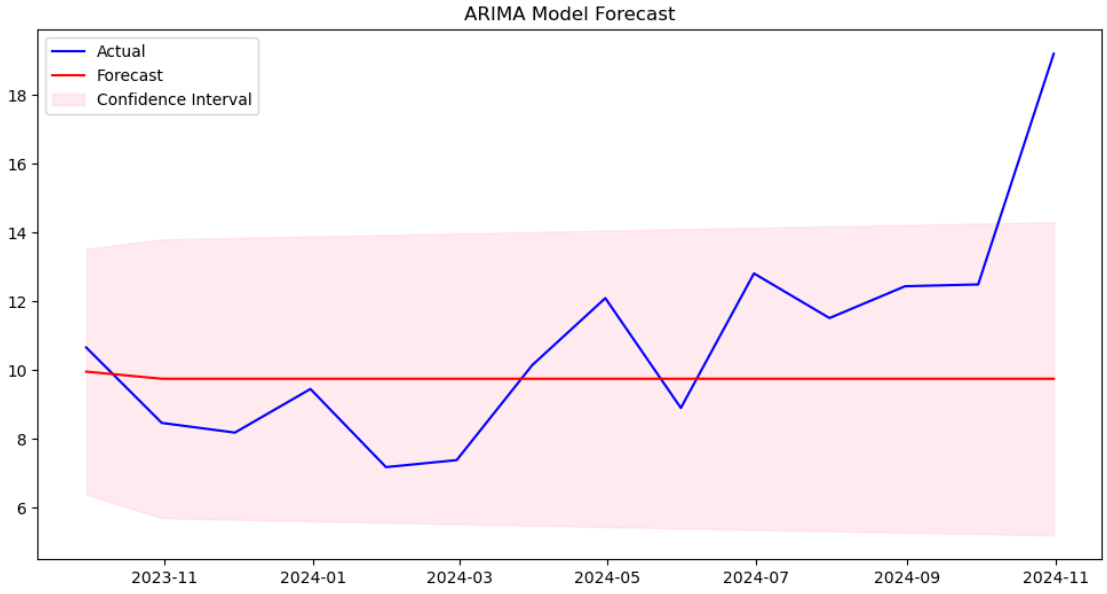
- **Mean Squared Error (MSE)**: Measures the average squared difference between actual and predicted values.
- **Root Mean Squared Error (RMSE)**: Provides the square root of MSE, representing the error in the same units as the target variable.
- **Mean Absolute Error (MAE)**: The average of absolute differences between actual and predicted values.
- **Mean Absolute Percentage Error (MAPE)**: Measures the accuracy of the predictions as a percentage.
- **R^2 (Coefficient of Determination)**: Represents the proportion of variance explained by the model.

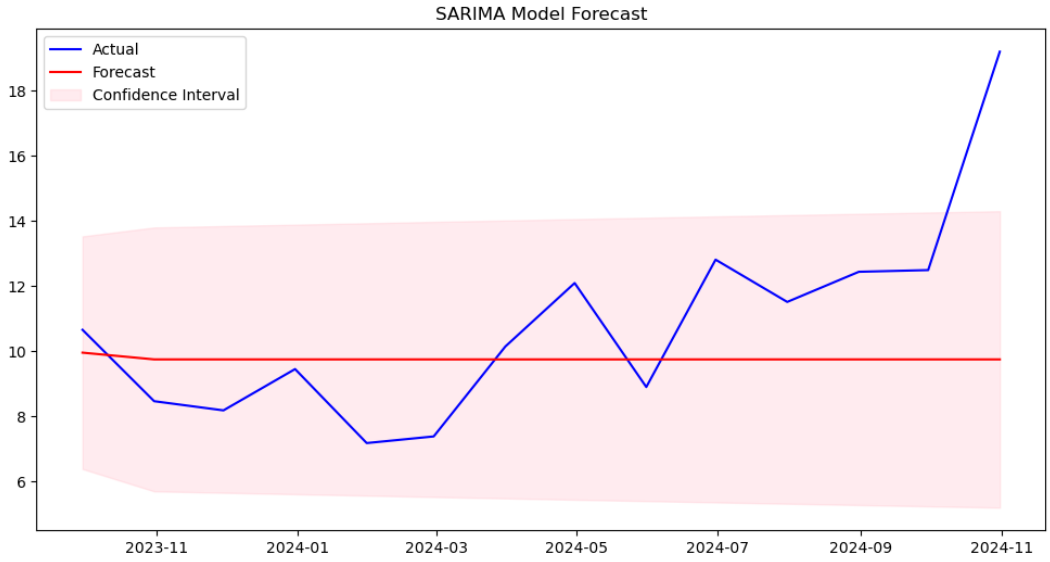
4.2.6 Performance

While ARIMA and SARIMA provide baseline forecasts, their inability to capture the sharp upward trend in real estate prices suggests that these models are insufficient for this dataset. To achieve better predictive accuracy, integrating advanced models and external variables should be considered.

- **Actual vs. Forecast**: The actual data (blue line) shows fluctuations over time, with a significant upward trend towards the end of the observed period. The forecast (red line) by the ARIMA and SARIMA model appears to be largely constant and fails to capture the upward movement in the actual data.
- **Confidence Interval**: The confidence interval (shaded pink region) is relatively wide, indicating high uncertainty in the ARIMA, SARIMA model's predictions. This wide interval suggests the model struggles to accurately estimate future price trends, likely due to inadequate handling of seasonality and recent volatility.
- **Mismatch with Actual Data**: The ARIMA, SARIMA forecast does not align well with the observed data. It seems to underestimate the upward trajectory of prices, which may be due to the model's assumption of stationarity and lack of explicit seasonal adjustments.

Neither ARIMA nor SARIMA successfully captures the sharp upward trend seen in the actual data. This indicates limitations in these models' ability to handle dynamic, non-linear trends or abrupt changes in the dataset. Both models exhibit wide confidence intervals, showing similar levels of uncertainty in their forecasts. Both models provide overly simplistic forecasts that do not reflect the complexity of the real estate price trends. The lack of responsiveness to recent fluctuations suggests that additional tuning or alternative approaches may be required.





- **Actual vs. Forecast**: The actual data (blue line) shows fluctuations over time, with a significant upward trend towards the end of the observed period. The forecast (red line) by the ARIMA and SARIMA model appears to be largely constant and fails to capture the upward movement in the actual data.
- **Confidence Interval**: The confidence interval (shaded pink region) is relatively wide, indicating high uncertainty in the ARIMA, SARIMA model's predictions. This wide interval suggests the model struggles to accurately estimate future price trends, likely due to inadequate handling of seasonality and recent volatility.
- **Mismatch with Actual Data**: The ARIMA, SARIMA forecast does not align well with the observed data. It seems to underestimate the upward trajectory of prices, which may be due to the model's assumption of stationarity and lack of explicit seasonal adjustments.

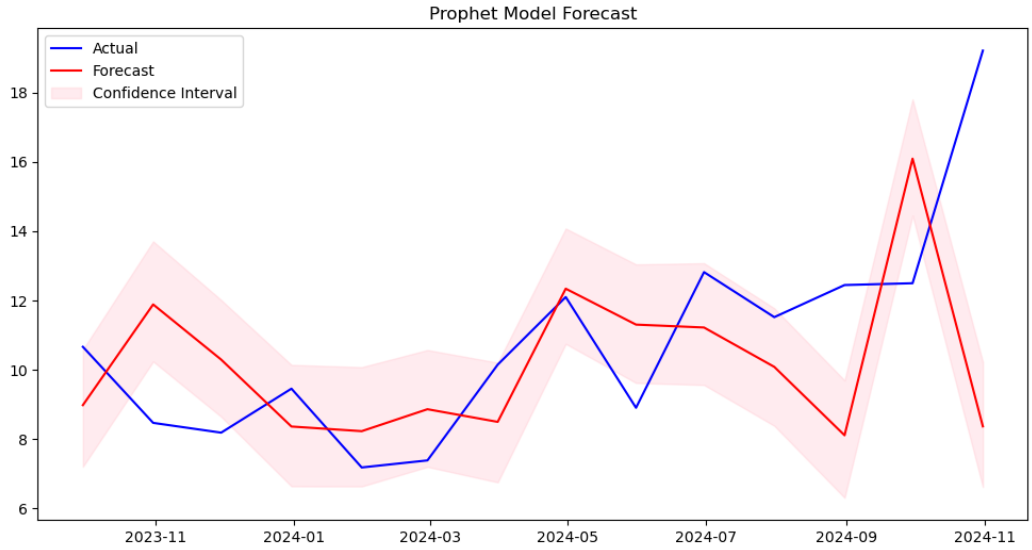
Neither ARIMA nor SARIMA successfully captures the sharp upward trend seen in the actual data. This indicates limitations in these models' ability to handle dynamic, non-linear trends or abrupt changes in the dataset. Both models exhibit wide confidence intervals, showing similar levels of uncertainty in their forecasts. Both models provide overly simplistic forecasts that do not reflect the complexity of the real estate price trends. The lack of responsiveness to recent fluctuations suggests that additional tuning or alternative approaches may be required.

- **Actual vs. Forecast**: The actual data (blue line) shows fluctuations over time, with a significant upward trend towards the end of the observed period. The forecast (red line) by the ARIMA and SARIMA model appears to be largely constant and fails to capture the upward movement in the actual data.
- **Confidence Interval**: The confidence interval (shaded pink region) is relatively wide, indicating high uncertainty in the ARIMA, SARIMA model's predictions. This wide interval suggests the model struggles to accurately estimate future price trends, likely due to inadequate handling of seasonality and recent volatility.
- **Mismatch with Actual Data**: The ARIMA, SARIMA forecast does not align well with the observed data. It seems to underestimate the upward trajectory of prices, which may be due to the model's assumption of stationarity and lack of explicit seasonal adjustments.

Neither ARIMA nor SARIMA successfully captures the sharp upward trend seen in the actual data. This indicates limitations in these models' ability to handle dynamic, non-linear trends or abrupt changes in the dataset. Both models exhibit wide confidence intervals, showing similar levels of uncertainty in their forecasts. Both models provide overly simplistic forecasts that do not reflect the complexity of the real estate price trends. The lack of responsiveness to recent fluctuations suggests that additional tuning or alternative approaches may be required.

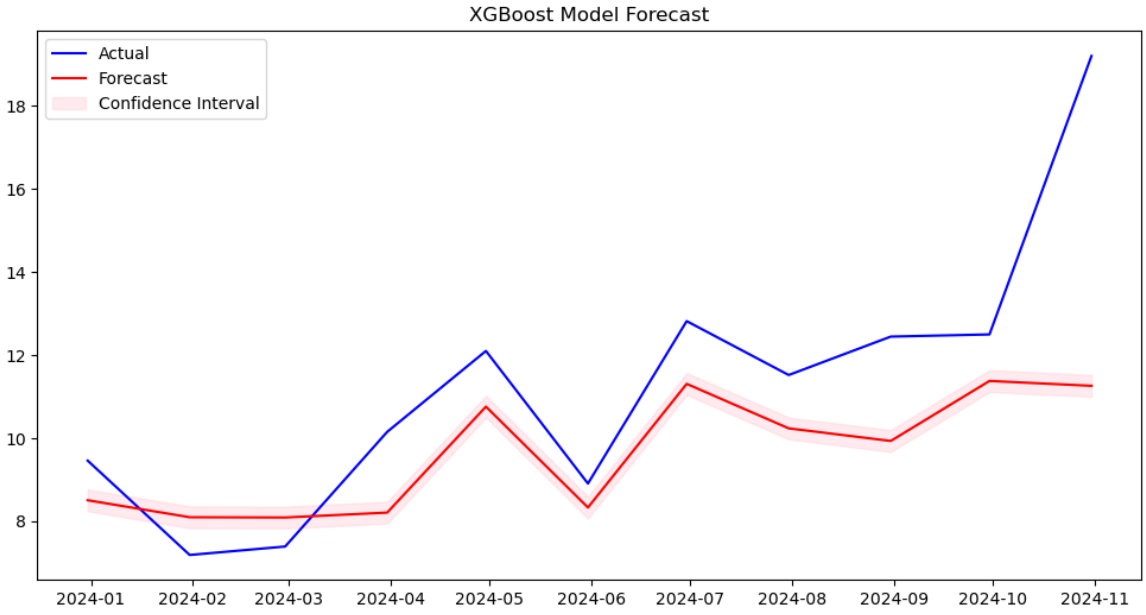
The Prophet model (red line) shows an ability to track seasonal patterns in the actual data (blue line), but it does not capture sudden spikes or sharp changes accurately, particularly toward the end of the observed period (e.g., the sharp increase in actual prices near November 2024). The forecast smooths over the actual fluctuations, indicating a preference for general seasonal trends rather than responding to abrupt shifts. The confidence interval (shaded pink region) is wide and expands significantly toward the end of the forecast period, reflecting increasing uncertainty in predictions over time. Despite capturing general seasonal trends, the wide intervals indicate the model's limited confidence in accurately predicting specific price values.

The Prophet model captures periodic fluctuations in the early parts of the time series but fails to anticipate the magnitude of rapid changes (e.g., steep rises or declines in the actual data). The model is better suited for long-term seasonal trends than short-term volatility.



The XGBoost model (red line) shows a stronger alignment with the actual data (blue line) compared to the Prophet model, particularly in capturing localized fluctuations. The model performs well in tracking both minor and moderate price movements over time, but it still underestimates the magnitude of extreme changes (e.g., the sharp rise toward November 2024). The confidence interval (shaded pink region) is narrower than the Prophet model, suggesting higher confidence in its predictions. The relatively tight intervals indicate that the XGBoost model is better calibrated to the data and provides more precise forecasts.

The XGBoost model demonstrates flexibility in handling complex, non-linear patterns in the data, thanks to the engineered features (e.g., lagged values, rolling statistics). However, it still struggles to predict extreme changes in price values, likely due to limited training data for such events.



The comparative analysis of forecasting models—SARIMA, ARIMA, Prophet, and XGBoost—reveals distinct differences in their performance, with XGBoost emerging as the most effective model for real estate price prediction. XGBoost demonstrates superior accuracy, achieving the lowest MSE (7.53), RMSE (2.74), MAE (1.89), and MAPE (14.76%), along with the highest R^2 (0.25), indicating its ability to effectively capture complex, non-linear relationships in the data. In contrast, SARIMA and ARIMA, while comparable in performance (MSE: 9.99, RMSE: 3.16, MAE: 2.29, MAPE: 19.76%), show limitations in explanatory power with negative R^2 values (-0.12), suggesting their inability to model the dataset's dynamic variations. Prophet performs the weakest (MSE: 13.26, RMSE: 3.64, MAE: 2.64, MAPE: 22.71%, R^2: -0.48), reflecting its challenges in capturing short-term volatility and non-linear trends, despite its strength in modeling seasonality. These findings underscore the efficacy of machine learning approaches like XGBoost for complex, non-stationary time series data and highlight the need for hybrid approaches or enhanced feature engineering to improve the performance of traditional and seasonal models.

	MSE	RMSE	MAE	MAPE	R2
SARIMA	9.990144	3.160719	2.292061	19.756274	-0.116073
ARIMA	9.990144	3.160719	2.292061	19.756274	-0.116073
Prophet	13.261293	3.641606	2.635904	22.712037	-0.481518
XGBoost	7.527496	2.743628	1.891236	14.764750	0.253333

4.3. Supervised Learning

For this section, we utilized some Machine Learning models to attempt to learn from the dataset such as patterns within the data, feature importances, and some specific ML problems. There are 2 ML problems that we identified and targeted at in this project: regression of real estates' prices and classification of property types. We experimented with some well-known and effective learning algorithms, including Linear Regression (LR), Support Vector Machine (SVM), Random Forest (RF), and Gradient Boosting (GB).

4.3.1. Methodology

Linear Regression is a supervised learning algorithm used for regression problems, capturing linear relationships of independent variables (also called features) and the dependent variable to be predicted. It models this relationship by fitting a linear hyperplane to the data, learning the coefficients by minimizing differences between predicted and target values using methods like least squares. Regularization methods are also used by adding L1- (Lasso) or L2-norms (Ridge) of coefficients to limit the domain of coefficients to be searched, allowing the model to avoid overfitting. We used the Ordinary Least Squares (OLS) LR, the L2 regularized Ridge and the L1 regularized Lasso for some basic models to see if our dataset follows any linear relationships.

Support Vector Machine is another supervised machine learning algorithm used for classification and regression tasks. It works by finding the hyperplane in higher dimensions or decision boundary that best separates data of different classes, maximizing the margin between them. SVM is effective in high-dimensional spaces and is particularly powerful for datasets that are not linearly separable, as it can use kernel functions to transform the data into a higher-dimensional space where a linear separation is possible. SVM is known for its robustness, especially in situations with complex or non-linear data, so we used the linear version of SVM to compare its performance with LR methods and used SVM with the Gaussian radial basis function as kernel function to attempt to capture more complex patterns.

We also used ensemble learning methods, which are popularly used due to its high performance and robustness. An ensemble learning model works by training multiple weak learners and combining their results to obtain the final predictions. The most common weak learner to be used is the Decision Tree as it provides simplicity, flexibility, and efficiency during training. Moreover, these trees are suited to be used in ensembles since they learn from different subsets of data or focus on correcting errors of previous trees, creating a much stronger overall model. Here, we made use of two ensemble learning models: Random Forest and Gradient Boosting.

Random Forest builds multiple trees during training and merges their results to improve accuracy and prevent overfitting. Each tree is trained on a random subset of the data, and at each split, a random subset of features is considered. This randomness helps make the model more robust and reduces the variance compared to individual decision trees. The final prediction is made by aggregating the outputs of all trees, typically by majority voting for classification or averaging for regression.

Gradient Boosting builds a strong predictive model by sequentially training trees, where each new tree attempts to correct the errors made by the previous ones. The model is built in a way that minimizes the residual errors (the difference between the predicted and actual values) using gradient descent. Gradient Boosting is highly effective for both classification and regression tasks, offering high predictive accuracy, especially when tuned properly. One famous and well-performing distribution of GB is the XGBoost framework which allows the use of GPUs in parallel training of multiple trees, and we will be using this framework in later parts to acquire some well-trained models.

4.3.2. Training Process and Evaluation Metrics

Two main ML problems we selected to solve on our real estates dataset are price regression and property type classification. Real estates' price prediction is a common problem that are explored by many for hands-on experience on Machine Learning, using provided features of estates to predict their prices. In many cases, external features are also included, such as economic indicators, to further improve models' performance. On the other hand, we found estates' type classification an appealing problem: identifying type of estates based on their features can allow to learn patterns of different classes of housing properties.

For this project, we used Linear Regression and its regularization variants, SVM Regression, Random Forest Regression and Gradient Boosting Regression to learn the price prediction problem. Tree-based algorithms like RF and GB will use a histogram-based learning technique, separating the real-valued prices into "bins" (discrete ranges of values) instead of learning directly which inputs go to which outputs, allowing more generalization in the model. SVM, Random Forest, Gradient Boosting are also used to learn the property type classification problem, and models provided by the XGBoost framework are also used with the expectation of better performance than the mentioned models.

To evaluate each model, we measured some metrics and used the results for inference and comparison between models. Commonly used metrics for the regression problem include Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Median Absolute Error (MedAE). Beside these metrics, we also used statistics of absolute percentage errors, which is the absolute error divided by the actual value to obtain the percentage by which the predicted value is away from the actual one. The absolute percentage error for one instance of data can be formulated as:

$$AbsolutePercentageError = |R_t - P_t|/R_t$$

where Rt is the actual result and Pt is the predicted results. The Mean Absolute Percentage Error (MAPE) and Median Absolute Percentage Error (MedAPE), which are the mean and median values of all absolute percentage errors in the dataset, are obtained and included for evaluation. The classification problem is also evaluated on commonly used metrics: Accuracy, Precision, Recall, and F1 score. The aim is to look for models with high accuracy, while still having good precision and recall scores. Precision and recall values close to each other indicates that the model does not simply make predictions to maximize accuracy by selecting the class with highest count (leading to low precision) or failing to make many correct predictions (leading to low recall).

The workflow for training a model is as follows. The dataset is preprocessed to encode categorical features (including law document type, address, and city) to numerical values, and the post date feature is extracted to get the post year and quarter. Then, the data is split into a training set and a testing set with 20% of original data used for testing. For each model, we used the Grid Search module from the Scikit-learn library to find the best combination of parameters from a list of provided values. The search will then return the set of parameters that has the best metric result on a 5-fold cross-validation run. The metric used for regression models is MAPE and for classification models is accuracy. The best performing model will run through the 5-fold validation

once more to calculate metrics' results. This process of model selection is run once more on the standardized version of the dataset to compare performance of a model with and without data standardization. Finally, we will test the best performing models of each learning algorithm on the testing set and compare their results.

The workflow for training a model is as follows. The dataset is preprocessed to encode categorical features (including law document type, address, and city) to numerical values, and the post date feature is extracted to get the post year and quarter. Then, the data is split into a training set and a testing set with 20% of original data used for testing. For each model, we used the Grid Search module from the Scikit-learn library to find the best combination of parameters from a list of provided values. The search will then return the set of parameters that has the best metric result on a 5-fold cross-validation run. The metric used for regression models is MAPE and for classification models is accuracy. The best performing model will run through the 5-fold validation once more to calculate metrics' results. This process of model selection is run once more on the standardized version of the dataset to compare performance of a model with and without data standardization. Finally, we will test the best performing models of each learning algorithm on the testing set and compare their results.

4.3.3. Real Estates Price Regression

The 5-fold cross-validation results can be seen as below, with the best (lowest) values in bold:

Table 1: Cross-validation results of price regression models

	RMSE	**MAE**	**MedAE**	**MAPE**	**MedAPE**
Linear Regression	79.908	8.972	4.545	3.238	0.743
Ridge	79.904	8.971	4.546	3.238	0.744
Lasso	79.112	8.954	4.535	3.244	0.747
Linear SVM	86.061	7.757	2.092	**1.632**	0.452
SVM with RBF kernel	25.376	7.582	2.607	1.739	0.555
Random Forest	**19.684**	5.354	1.887	2.947	0.329
Gradient Boosting	23.724	5.939	1.593	1.913	0.321
XGBoost	21.925	**5.186**	**1.279**	2.497	**0.239**

Linear Regression models yielded the worst results among all models, with a RMSE of 79.9 and MAE of 8.97. Percentage errors are as high as 3.23 for MAPE and 0.74 for MedAPE, meaning the model can hardly predict the prices correctly. Using standardized data and regularizations barely improved performance as Ridge and Lasso regressions provided almost similar results, as well as with standardized data. However, the Lasso model, whose learning

strategy encourages sparse coefficients, acquired near-zero coefficients for address, postal code, and place importance, signaling that these features may be less important than others.

Linear SVM showed noticeably better results on absolute errors and absolute percentage errors; however, it has higher RMSE which can be interpreted as some larger errors appeared, yet percentage errors are lower meaning the model fitted better to the dataset than LR models. When combined with the Gaussian radial basis function as kernel function, its RMSE value further decreased to just 25.3 in return of just a slight change in other metrics, showing even better performance.

Ensemble learning models all acquired better results than SVM, however MAE and MAPE are still as high as 5.18 and 1.91, respectively. Despite that, median values are significantly lower compared to the corresponding mean values, meaning there are some predictions with significantly skewed predictions. It is also worth noting that while the Gradient Boosting model used Decision Trees with a shallow depth of 3, Random Forest and XGBoost models both used deeper trees with a maximum depth of 12. In terms of feature importances, the most important features inferred from these models are similar, including area, property type, number of bedrooms and bathrooms.

For LR and SVM models, we also noticed that applying standardization to data slightly improved metrics' results, while doing so for tree-based methods like Random Forest or Gradient Boosting provides either no change or slight reduction in results. This may be because standardizing data converts them into a smaller range of value, making it harder for Decision Trees to discretize numerical values into smaller ranges during learning process. Longitudes and latitudes also saw high importances in several models, since they function as locational features and can be grouped to identify regions.

4.3.4. Price Prediction with LLM

According to Robert Vacareanu et al. [6], their research has demonstrated that large language models (LLMs) can perform regression tasks by providing a number of example contexts. The study was conducted on over 20 LLMs and compared them with more than 20 traditional regression models in performing regression tasks across a total of 16 datasets, including well-known regression test datasets such as Friedman, as well as randomly generated datasets. The models were trained and then ranked based on MAE. The results of the study show that although traditional regression models such as Linear Regression and Random Forest still yield good results, the performance of LLMs on certain datasets is comparable. Specifically, the strongest LLMs tested in the experiment were found to produce results that were on par with traditional models, and in some cases, even outperform them on certain datasets.

Motivated by these findings, in this project, we experiment with LLMs for regression tasks on our dataset. Due to resource constraints, budget limitations, and token restrictions, we only test the Llama Vision model, which is available for free on TogetherAI. First, the model is accessed from TogetherAI using an API key. Next, we prepare the prompt for the LLM using a prompt template from LangChain. The prompt displays a number of samples with their features and the corresponding ground truth, along with a single sample that has known features but no target value

(the target in this context is the price of a real estate property). The prompt instructs the LLM to predict the target price based on the provided information. Afterwards, we take the predicted target value and evaluate the performance of the regression task.

The model is trained with varying sample sizes and then used to make predictions on a single sample. This training process is repeated 10 times, and the results are averaged over these 10 iterations. The model's performance is evaluated using the same metrics as traditional models, including MAE, RMSE, MedAE, MAPE, and MedAPE.

K	MAE	RMSE	MedAE	MAPE (%)	MedAPE (%)	Time
401	4.21	5.89	2.35	54.46	53.13	2:53
301	4.30	6.30	2.17	160.55	88.21	1:50
201	5.03	10.44	2.08	211.31	58.91	1:06
101	9.77	12.60	9.52	191.94	73.26	1:00
51	10.34	14.16	7.60	124.38	77.63	1:03
21	14.07	17.80	12.78	212.48	93.40	0:58

4.3.5. Real Estates Type Classification

The 5-fold cross-validation results can be seen as below:

Table 2: Cross-validation results of property type classification models

	Accuracy (%)	Precision	Recall	F1 Score
Linear SVM	58.76	0.489	0.434	0.429
SVM with RBF kernel	49.62	0.4	0.35	0.351
Random Forest	80.93	0.804	0.724	0.751
Gradient Boosting	81.59	0.745	0.724	0.733
XGBoost	83.14	0.794	0.784	0.786

SVM models performed not too well with only around 58% accuracy for linear SVM and 49% for non-linear SVM. During training, using standardized data slightly improved the linear SVM model and significantly improved the non-linear one, and a breakdown of linear SVM's decision boundaries provided some interesting insights to the dataset.

Across all classes of property types, number of bedrooms and bathrooms, along with property price, are the most significant features with large coefficient values compared to other coefficients of the same hyperplane. With L1 regularization applied, the studio apartment class is revealed to be independent of area, latitude or longitude when placed next to other classes, and it

mostly depends on its price and number of bedrooms, meaning that a studio apartment tends to have possibly lower prices compared to other estate types.

Overall, the ensemble models performed well with above 80% accuracy. However, XGBoost stood out with precision and recall scores closest to each other, signaling that the model made well-generalized predictions. The most important features provided are like other models, including area, number of bedrooms and bathrooms, and price.

We also tried out feature selection with the XGBoost model, excluding some of the least significant features such as address, quarter in which the estate is posted for sale, and location importance; cross-validation results came out not too different with 82.04% accuracy, 0.77 for precision, and 0.76 for recall and F1 score. Although metrics' results slightly decreased, the difference is small enough to allow removal of some unimportant features, reducing data complexity and provide an easier-to-use model for real-time inference

5. Result

5.1. Testing results on Real Estates Price Regression

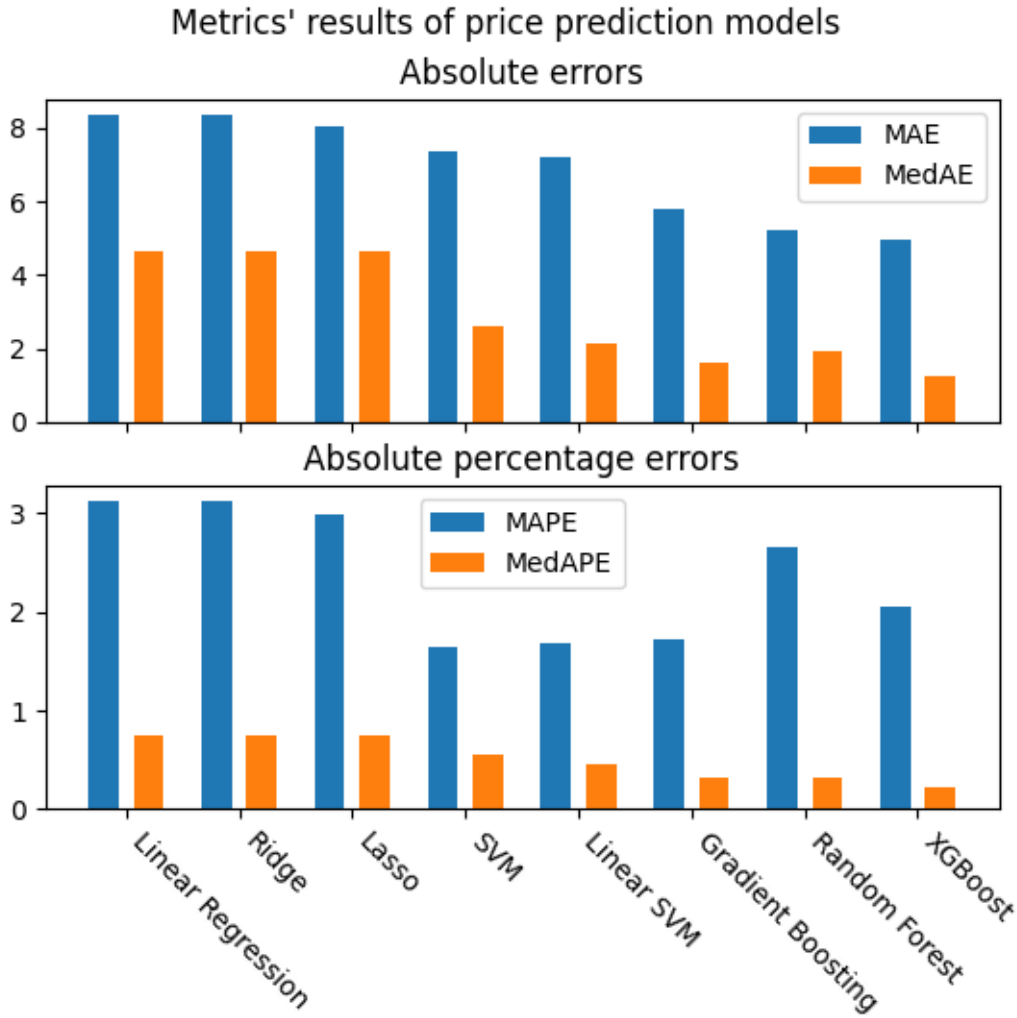
Table 3: Testing results of price regression models

	RMSE	**MAE**	**MedAE**	**MAPE**	**MedAPE**
Linear Regression	34.382	8.36	4.659	3.111	0.740
Ridge	34.395	8.359	4.658	3.111	0.740
Lasso	22.339	8.036	4.657	2.992	0.744
Linear SVM	45.675	7.178	2.12	1.671	0.452
SVM with RBF kernel	23.55	7.368	2.607	**1.64**	0.548
Random Forest	**18.281**	5.198	1.902	2.657	0.325
Gradient Boosting	21.92	5.816	1.608	1.714	0.324
XGBoost	19.924	**4.94**	**1.242**	2.054	**0.227**

We selected some models stated in previous parts and measured their performance on the reserved testing dataset. For the price regression problem, testing results saw a similar pattern to the validation results: linear regression models performed worst; linear SVM has highest RMSE while SVM with RBF kernel has noticeably lower RMSE; and the three ensemble models performed best. However, Lasso regression performed significantly better than the OLS LR and Ridge regression, recording lower RMSE, MAE, and MAPE.

A visualization of metrics acquired from absolute errors and absolute percentage errors also provided some interesting insights. In both cases, median values are much lower than mean values; for absolute errors, the medians are approximately one-third to a half of the means, and

this ratio seems to be even smaller for absolute percentage errors. This indicates that either the models fail to make good predictions on more than 50% of the dataset, or there are some cases where models predict a much higher price compared to the actual price, leading to high percentage errors. By thorough checking of model outputs in comparison with actual results, we confirmed that the latter assumption is true: some cases resulted in predicted prices that are 500 to 800 times larger than actual prices. This means that models still fail to identify estates with higher prices up to hundreds of billion VND; this could also be due to the data not having features required to differentiate these estates from lower price ones'.



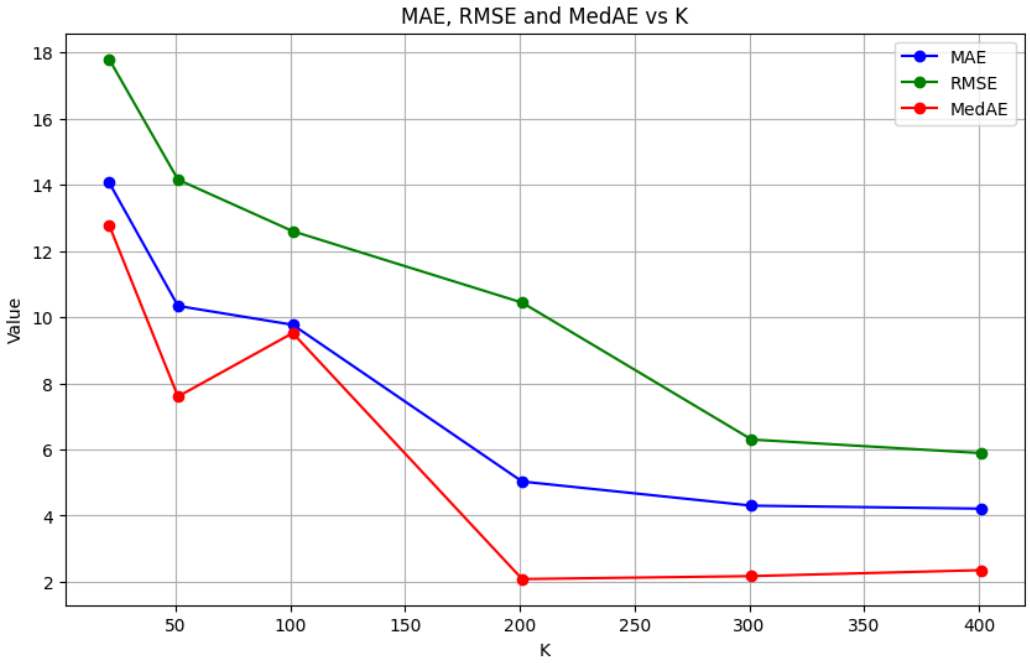
Plot of metrics' results of price regression models on testing set

5.2. Testing results on Price Prediction with LLM

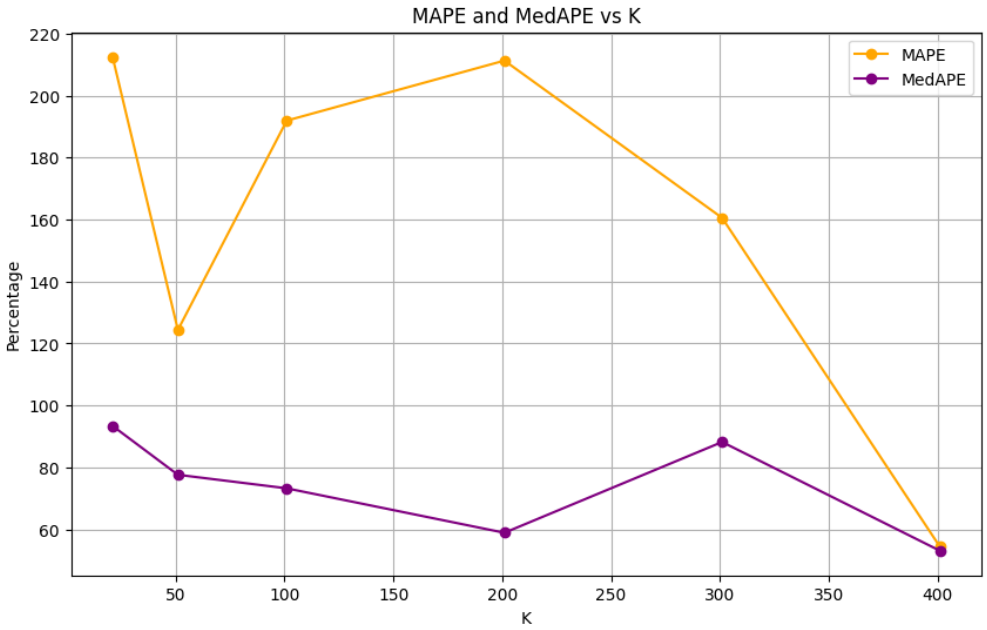
Table: Performance of LLM with Price Prediction task

K	MAE	RMSE	MedAE	MAPE (%)	MedAPE (%)	Time
401	4.21	5.89	2.35	54.46	53.13	2:53
301	4.30	6.30	2.17	160.55	88.21	1:50
201	5.03	10.44	2.08	211.31	58.91	1:06
101	9.77	12.60	9.52	191.94	73.26	1:00
51	10.34	14.16	7.60	124.38	77.63	1:03
21	14.07	17.80	12.78	212.48	93.40	0:58

The results are described in the Table above. The performance of the model varies significantly as the number of training samples (K) changes. As K decreases, there is a noticeable increase in the MAE, RMSE, MedAE, and MAPE metrics, which suggests that the model's accuracy and ability to generalize are adversely affected by smaller training datasets. For instance, with **K = 401**, the MAE is 4.21, the RMSE is 5.89, and the MedAE is 2.35, indicating relatively strong performance. However, when K is reduced to **K = 21**, the MAE rises to 14.07, and the RMSE increases to 17.80, showing that the model struggles to perform well with fewer training samples.



Interestingly, the MedAPE and MAPE metrics also demonstrate the same trend. With a larger training dataset (K = 401), the MAPE is 54.46%, and MedAPE is 53.13%. As K decreases, these values increase significantly, reaching 212.48% and 93.40%, respectively, when K is reduced to 21. This indicates that the model's ability to predict target values accurately is highly sensitive to the size of the training dataset.



In terms of training time, we observe that as K decreases, the time taken for training also decreases, which is expected due to the smaller dataset size. The training time for **K = 401** is 2:53, while for **K = 21**, it only takes 0:58, demonstrating the trade-off between the amount of training data and the model's performance.

Overall, the results highlight the importance of having an adequately sized dataset for regression tasks using LLMs. While large datasets lead to better performance in terms of error metrics, smaller datasets result in a notable deterioration of prediction quality, particularly for metrics like MAPE and MedAPE. This emphasizes the need for a careful balance between training time and dataset size when using LLMs for regression tasks.

5.3. Testing results on Real Estates Type Classification

The property type classification problem saw similar results to the regression problem, that is patterns of metrics on testing set are not too different from that of the training set. Aside from the SVM model, remaining models saw consistent precision and recall scores, with Gradient Boosting having the largest difference between the two metrics of just 0.016. The XGBoost model once again stands out with 83.6% accuracy compared to just below 82% of RF and GB models.

Table 4: Cross-validation results of property type classification models

	Accuracy (%)	**Precision**	**Recall**	**F1 Score**
Linear SVM	57.86	0.483	0.426	0.421
Random Forest	81.7	0.793	0.787	0.79
Gradient Boosting	81.99	0.753	0.737	0.744
XGBoost	83.64	0.78	0.778	0.779

We then explored confusion matrices of the ensemble learning models to find out which classes are most mistaken. Surprisingly, townhouses seem to be the most confusing estates across all three models. Townhouses are houses open to main roads or streets and usually used for business purposes like stores or cafeterias. Therefore, their prices tend to be much higher than ordinary private houses, while still having similar characteristics. On the other hand, some houses used for private purposes can have high prices according to the real estate markets, which makes it confusing to identify between private houses and townhouses with just the provided features. Apartment seems to be the least confusing class, which can be understandable since apartments have characteristics easy to be differentiated from houses or villas, such as lower prices, less rooms, etc.

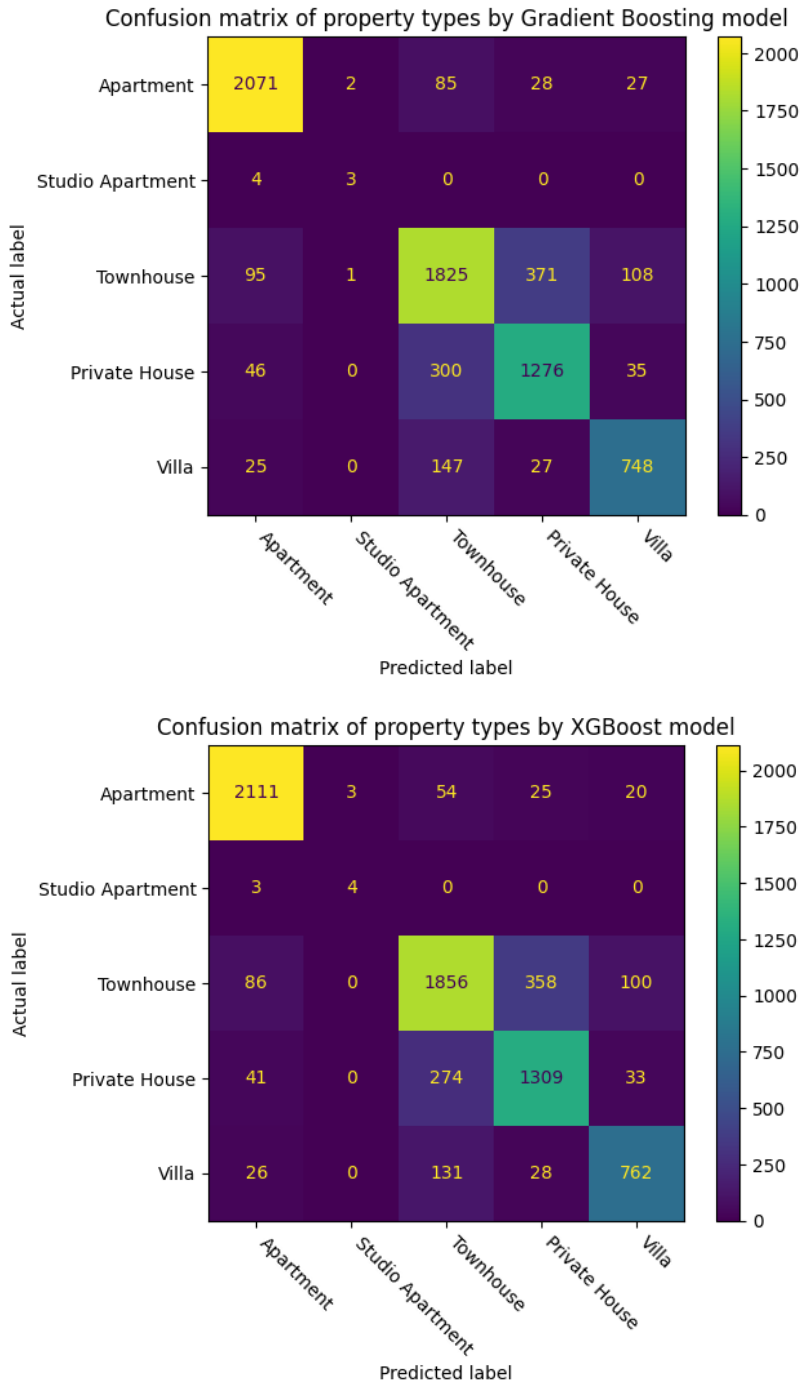


Figure 2: Confusion matrices of Random Forest, Gradient Boosting, and XGBoost models

6. Challenges

The inconsistency and incompleteness of the dataset is one of the primary challenges in this project. While collecting data from various sources, some records had missing values for critical features such as square footage, number of rooms, or geographic location. Handling missing data through imputation or exclusion can introduce bias or lead to the loss of valuable

information. Additionally, the accuracy of the data was sometimes questionable due to variations in property listings, outdated information, or inconsistencies in data formats across different sources.

The dataset contained a wide variety of attributes, but not all of them were relevant for the price prediction task. The mission to decide which features to include and how to preprocess them (e.g., normalizing numerical values, encoding categorical features) required careful analysis and domain knowledge.

Real estate price predictions is considered sensitive to outliers, for example, properties that are extremely overpriced or underpriced compared to the rest of the dataset. Identifying and handling these outliers, whether through data transformations or the use of robust models, was crucial for improving model performance and preventing skewed predictions.

The size of the dataset was another limitation. The dataset we used in this project only have about 20,000 records, focus only on Hanoi and Ho Chi Minh City, which could reduce the robustness of the model. It also limits the ability to generalize across the broader real estate market.

7. Future Planning

The next steps in improving the prediction model for house prices, based on our self-collected dataset, are outlined as follows:

1. **Data Expansion**:
    - One of the most important steps will be to expand the dataset. This will involve sourcing more data from diverse geographical areas, different types of properties (e.g., single-family homes, apartments, commercial properties), and including additional features such as property age, the number of nearby schools or parks, and proximity to public transport. More data will help improve the model's generalizability and accuracy.
    - Additionally, we will explore data augmentation techniques to synthetically expand the dataset while preserving the diversity of real estate price variations.
2. **Enhanced Feature Engineering**:
    - Future efforts will be directed towards refining feature selection and engineering. This includes identifying new, potentially influential features that could improve model performance. We will also explore advanced feature interactions and combinations using techniques such as polynomial feature expansion and deep learning-based embeddings.
    - Another aspect will involve incorporating temporal features, such as seasonality or economic factors, that could affect real estate prices over time. This could help the model account for fluctuations in the market.
3. **Incorporation of Additional Models**:

- While traditional machine learning models such as Random Forest and Linear Regression were explored, future work will investigate the use of more advanced techniques, such as Gradient Boosting, XGBoost, and deep learning models (e.g., Neural Networks). These models may offer improved predictive performance and the ability to capture complex, non-linear relationships between features.
- We will also explore the use of ensemble methods, where predictions from multiple models are combined to improve accuracy and robustness.

4. **Model Optimization and Hyperparameter Tuning**:
    - Ongoing efforts will focus on hyperparameter optimization to improve the model's performance. Techniques like grid search, random search, and Bayesian optimization will be employed to identify the best hyperparameters for the models.
    - In addition to this, we plan to evaluate the performance of the models with cross-validation to ensure that they generalize well to unseen data.
5. **Real-Time Data Integration**:
    - In the future, we plan to integrate real-time property data (e.g., recent sales, auction prices, or new listings) into the model, allowing the prediction system to adapt to changing market conditions. By continuously updating the dataset with fresh data, the model can provide more accurate and up-to-date predictions.
6. **Evaluation and Deployment**:
    - Once the model reaches satisfactory performance, it will be deployed as a web service or integrated into a real estate platform for practical use. The model's accuracy and efficiency will be continuously monitored in real-world conditions, and user feedback will be used to make iterative improvements.
    - We will also consider making the model available for public use, where potential homebuyers and real estate professionals can leverage it for price predictions and market analysis.

8. Libraries and Tools

Data Handling and Preprocessing

- **`pandas`**: The core library for data manipulation, used for cleaning, resampling time series data, and managing structured datasets. It enabled efficient handling of large datasets, including operations such as time-based indexing, grouping, and interpolation of missing values.
- **`NumPy`**: Essential for numerical computations, particularly in feature engineering tasks like creating lagged features and calculating rolling statistics. It also facilitated efficient array-based operations for scalability.

- **`Selenium`**: Employed for web scraping tasks to collect real-time data from dynamic websites. Selenium's ability to interact with web elements programmatically was crucial for augmenting the dataset with additional real estate market data.

Visualization and GUI Development

- **`Matplotlib`**: Used to create detailed plots for time series trends, forecasted values, and confidence intervals. These visualizations were essential for presenting the results in an interpretable and user-friendly manner.
- **`seaborn` (sns)**: Complemented Matplotlib by providing advanced, aesthetically pleasing statistical graphics. Seaborn was particularly useful for generating heatmaps, pair plots, and distribution plots to uncover relationships between variables and enhance exploratory data analysis (EDA).
- **`logging`**: Implemented to log events and track workflow progress during data processing, feature engineering, and model training. This ensured transparency and traceability throughout the analysis.
- **`tkinter`**: Integrated to build a graphical user interface (GUI) for user-friendly interaction with the project. The GUI allowed users to upload datasets, select forecasting models, view predictions, and interact with visual outputs without requiring direct access to the underlying code.

Statistical Modeling

- **`statsmodels`**: Leveraged for conducting stationarity tests (Augmented Dickey-Fuller test) and performing Seasonal-Trend decomposition (STL). These capabilities were critical for identifying trends, seasonality, and residual patterns in the data and ensuring compatibility with statistical models.
- **`pmdarima`**: Simplified the implementation of ARIMA and SARIMA models by automating the parameter tuning process through the `auto_arima` function. It was instrumental in identifying the best model configurations for capturing linear patterns and seasonality.

Machine Learning and Advanced Forecasting

- **`Prophet`**: A robust forecasting tool developed by Facebook, used for modeling long-term seasonality and growth trends in real estate prices. Its ability to handle missing data and incorporate yearly, weekly, and daily seasonality made it suitable for capturing recurring patterns.
- **`XGBoost`**: An advanced gradient boosting framework used for non-linear modeling of time series data. XGBoost integrated engineered features such as lagged values, rolling statistics, and STL decomposition components to enhance prediction accuracy. Hyperparameter tuning was performed using scikit-learn's `GridSearchCV` with `TimeSeriesSplit` for time-sensitive cross-validation.

Evaluation and Model Comparison

- **`scikit-learn`**: A comprehensive machine learning library used for:
    - **Feature Scaling**: Standardizing input features to ensure consistency during model training.
    - **Metrics**: Evaluating models using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and R^2. These metrics provided a rigorous framework for assessing and comparing model performance.
    - **Hyperparameter Optimization**: Facilitated through `GridSearchCV`, enabling systematic parameter tuning for XGBoost and other models.

Deployment and Reusability

- **`joblib`**: Utilized for model persistence and serialization. This allowed trained models to be saved and reloaded for subsequent forecasting tasks without retraining, ensuring computational efficiency and reproducibility.
- **`os`**: Integrated for directory management, ensuring an organized structure for saving models and outputs.

9. References

[1] Pai, Ping-Feng, and Wen-Chang Wang. 2020. "Using Machine Learning Models and Actual Transaction Data for Predicting Real Estate Prices" *Applied Sciences* 10, no. 17: 5832. https://doi.org/10.3390/app10175832

[2] Van Buuren, Stef. *Flexible Imputation of Missing Data*. Accessed December 11, 2024. https://stefvanbuuren.name/fimd/.

[3] Little, Roderick J. A., and Donald B. Rubin. 2011. "Complete-Case and Available-Case Analysis." *Statistical Analysis with Missing Data*, 3rd edition. Accessed December 11, 2024. https://pmc.ncbi.nlm.nih.gov/articles/PMC3074241/.

[4] Kearnz. 2024. "Autoimpute Tutorials: A Guide to Imputing Missing Data with Python." Accessed December 11, 2024. https://kearnz.github.io/autoimpute-tutorials/.

[5] Statistical Horizons. 2024. "Predictive Mean Matching: A Method for Multiple Imputation." Accessed December 11, 2024. https://statisticalhorizons.com/predictive-mean-matching/. [6] R. Vacareanu, V.-A. Negru, V. Suciu, and M. Surdeanu, “From Words to Numbers: Your Large Language Model Is Secretly A Capable Regressor When Given In-Context Examples,” *arXiv (Cornell University)*, Apr. 2024, doi: 10.48550/arxiv.2404.07544.

10. Contribution

- **Le Nhat Quang, Nguyen Tri Thanh - Data Preparation**

  Le Nhat Quang and Nguyen Tri Thanh were responsible for curating and preparing the dataset for analysis. Their contributions included data cleaning, resolving inconsistencies,

handling missing values, and ensuring data integrity. This phase also involved preprocessing tasks such as formatting date columns, converting categorical variables into machine-readable formats, and ensuring all numerical variables were standardized and free of outliers. Additionally, they established a robust pipeline to streamline data processing for subsequent stages of the project.

- **Pham Tuan Kiet - Exploratory Data Analysis (EDA)**

  Pham Tuan Kiet conducted a detailed exploratory data analysis to uncover initial insights and patterns within the data. This included visualizing trends, distributions, and correlations to identify key factors influencing real estate prices. The EDA phase provided critical insights into potential seasonality, non-linearity, and anomalies within the data, which informed both feature engineering and model selection. Furthermore, summary statistics and data visualizations prepared during this stage laid the foundation for understanding the dataset's structure and characteristics.

- **Phan Tran Viet Bach, Nguyen Xuan Thanh - Modeling and Forecasting**

  Phan Tran Viet Bach and Nguyen Xuan Thanh led the development and evaluation of predictive models. Their work included implementing and fine-tuning multiple forecasting algorithms, such as ARIMA, SARIMA, Prophet, and XGBoost and other machine learning models to capture both linear and non-linear relationships within the data. They performed rigorous hyperparameter tuning, cross-validation, and comparative analysis to identify the most effective model for real estate price prediction. Their contributions also encompassed advanced feature engineering, such as creating lagged features and incorporating seasonal components, to enhance model accuracy and reliability. Additionally, they were responsible for evaluating model performance using metrics like MSE, RMSE, MAE, MAPE, and R^2, ensuring a data-driven selection process for the final model.