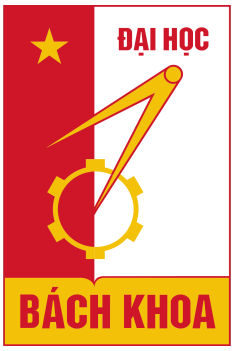
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
Symposium On Information and Communication Technology



Housing Marketing Prediction

Authors:

STT	Full name	Student ID
1	Nguyen Trung Dung	20204906
2	Bui Anh Duong	20225489
3	Nhu Minh Ha	20212784
4	Tran Quoc Khanh	20204881
5	Nguyen Thanh Minh	20225450
6	Bui Thi Thu Uyen	20225537

Supervising Lecturer: TS. Than Quang Khoat

Abstract

The creation of a reliable housing price forecast model is always necessary for the socioeconomic advancement and welfare of the populace. Using publicly available datasets, a variety of machine learning methods, including Gradient Boosting, XGBoost, KNN, DecisionTree, Random Forest, SVM, and ADA Boost are used in this study to forecast home prices. There were 14,464 records in the housing dataset, which was retrieved from the Batdongsan website (`https://batdongsan.com.vn/`). The real estate/economic database, maps, and other related information are included in the records, which are open to the public. Typically, the database is updated every day in accordance with Vietnamese legislation. Then, the classification model performances of the home price prediction models created using machine learning approaches are compared. Finally, a better methodology for predicting house prices is suggested to help the housing market. A home seller, buyer, or real estate broker in particular might gain knowledge to help them make better-informed decisions when taking the housing price projection into account. The empirical findings show that the XGBoost algorithm outperforms the other models in terms of prediction model performance for all metrics under study: the coefficient of determination, F1 score, Recall, Precision, and Accuracy.

Acknowledgement

We would like to thank our Machine Learning teacher: Mr. Than Quang Khoat, who gave us a golden opportunity to work on this project. The knowledge that he taught us is indispensable to completing this project. Preparing this project in collaboration with our teachers was a refreshing experience. We'd also like to express our gratitude to our school principal wholeheartedly.

Contents

1 Introduction

1.1 Background

Machine learning has been increasingly used for forecasting, yielding progressively better results that have significantly impacted the economic landscape. These prediction models are now crucial in almost every sector of economics, improving in accuracy as computing power grows, allowing for the processing of vast data sets. This study employs several machine learning techniques, including XGBoost, Random Forest, Support Vector Machine, K-nearest Neighbors, and GradientBoosting, to analyze the housing price issue. We propose a target binning technique based on machine learning to address this problem. Our findings show that the XGBoost algorithm with target binning outperforms other models in terms of model score and computation time.

1.2 Objective

In today's market, buying or selling real estate is both exciting and costly. Consumers often have to visit numerous locations and pay commissions to real estate brokers to obtain price information. Therefore, it is essential to forecast home values objectively to assist both buyers and sellers in making informed decisions. Accurate predictions of home prices, considering customers' priorities and financial constraints, are vital for real estate consumers. By analyzing housing data to forecast prices, customers can invest in real estate without relying on agents, thereby reducing transaction risks.

2 Methodology

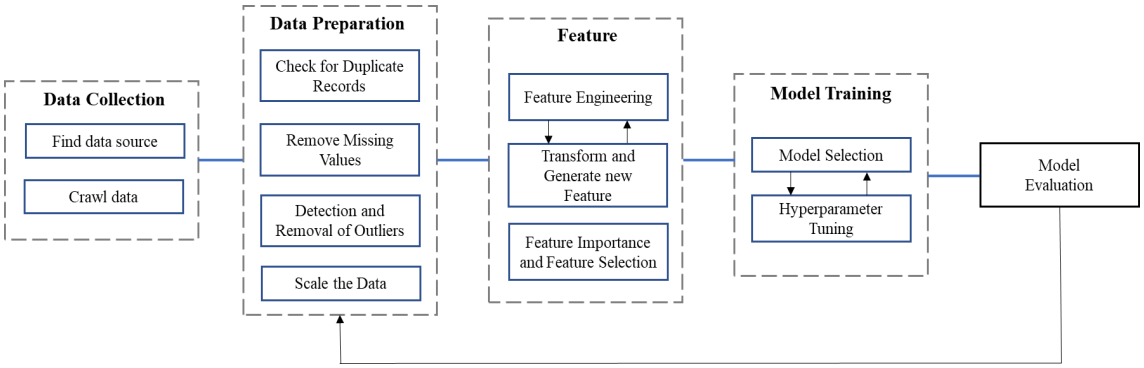


Figure 1: Fig 1 : Methodology

Before building any statistical and machine learning models, the initial task is data collection. This involves two steps: identifying the data source and extracting the data. Following this, data preparation is necessary, which includes data preprocessing and exploratory data analysis.

Once the data is prepared, we proceed to feature processing. This involves tasks such as feature engineering, transforming and generating new features, and

determining feature importance and selection. After this, we select the models to address the problem. We begin by analyzing the theoretical background of each model, selecting a baseline model, fitting the models to the data, and statistically evaluating the results for a deeper assessment. The collected data is divided without shuffling into two parts: an 60% training set a 20% test set and a 20% validation set. The detailed evaluation mentioned above is conducted using the test set. The best-performing model is then retrained and re-evaluated during the Model Evaluation phase.

Finally, we compare the performance of each machine learning model and draw conclusions based on the results.

3 Data Preparation

3.1 Data collection

For this research, real estate datasets were gathered from the publicly available website `https://batdongsan.com.vn/`, which is state-licensed. The datasets include an up-to-date property sales database, maps, and other relevant information, and are updated daily based on projects posted by investors and moderated for accuracy. Specifically, property data and sales records, including listing and closing prices, were collected. The housing data for this study was collected in December 2023 using Selenium and BeautifulSoup libraries to extract data from the website. Initially, a total of 14,464 records with 10 variables (area, price, house direction, balcony direction, number of rooms, number of toilets, furniture, legal status, district, number of hospital) were extracted from the database.

3.2 Data preprocessing

For the data preprocessing process, we remove duplicates which have the same information in all columns. After that, we remove some data that do not have meaning in future work. We reformatted the data to ensure it was in the correct format.

For the area column, which had data in the format “x m2”, we removed “m2” and returned the float value “x”. For the price column, which included prices in different units (ty/m2, ty, trieu, etc.), we converted all data to the same unit, “trieu/m2”, to facilitate future work. The columns for rooms and toilets were processed similarly to the area column.

For the address column, which contained latitude and longitude attributes, we created two separate columns: ’x’ for latitude and ’y’ for longitude. The columns for legal status and furniture presented more challenges, as the data were strings described by the sellers without a consistent format. Our task was to standardize this data for analysis.

Upon analysis, we identified three types of legal status: “so do”, “so hong”, and “chua so”, along with various other descriptions given by sellers, which we categorized as “nan”. Similarly, for furniture, we identified four types: “cao cap” (high-end), “day du” (fully furnished), “co ban” (basic), and “nguyen ban” (unfurnished).

Room	Number of bedrooms
Area	Area of the house
Toilet	Number of toilets
x	Latitude of the house
y	Longitude of the house
Khoang_cach	Distance from the house to the District People's Committee
Quan	The district that the house located in
Polistic	The legal status of the house
Furniture	Type of furniture of the house
direct	The orientation of the house
N_hospital	Total number of hospitals within 2km radius

Figure 2: Fig 2 : Feature

We detect and remove the next part in the pipeline.

The dataset preparation involved several steps to handle outliers and missing data. First, for the 'area' column, which had multiple outliers above 200 m^2, an upper threshold of 400 m^2 was set for removal. For the 'room' and 'distance' columns, each had only one outlier, and the upper thresholds were set to 4 and 15, respectively.

Handling missing data was crucial due to its extent in six columns: house direction, furniture, polistic, toilet, and room. The 'toilet' and 'room' columns had less than 10% missing values, so the missing values in these columns were dropped. For the other four columns, which had over 20% missing values, a different approach was used. A "Extra Trees Classifier" was employed to fill in the missing values. This model was trained on the dataset excluding columns with missing values (except the price column).

After the data preparation, the dataset consisted of 11,004 records. The correlation matrix of the dataset was generated, showing the relationships between features. Table 1 provided descriptions of the features and target variables, highlighting that most features were real data types, which are essential for building the classification model.

3.3 Data analysis

In this project, our objective is to transform the task of predicting the value of a house into a classification problem. The prices of houses vary within a range of 10 million to 140 million VND per square meter. To address this issue, we have devised a plan to divide the selling prices into 5 separate groups, which we have defined as the *price_class* feature.

The following is a list of the *price_class*:

-Less than 20 million VND/m2

-20-30 million VND/m2

-30-45 million VND/m2

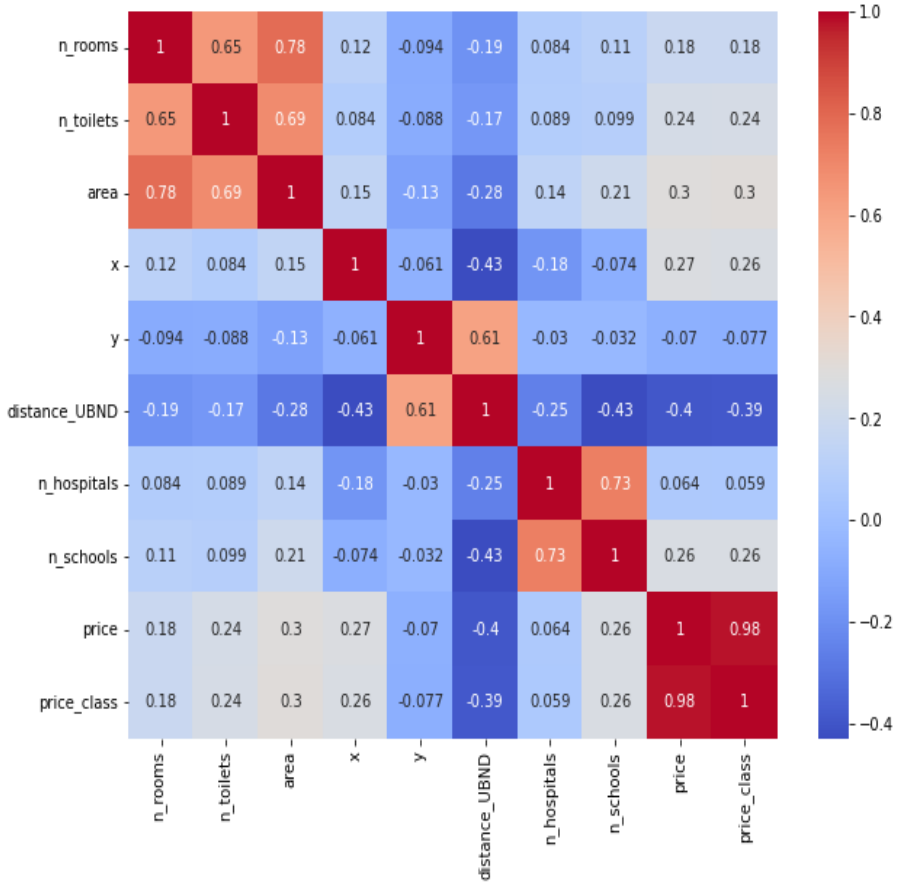


Figure 3: Fig 3: Correlation matrix for the given dataset generated

-45-60 million VND/m2

-More than 60 million VND/m2

After dividing the house prices into these classes, we generated a distribution of the *price_class* feature

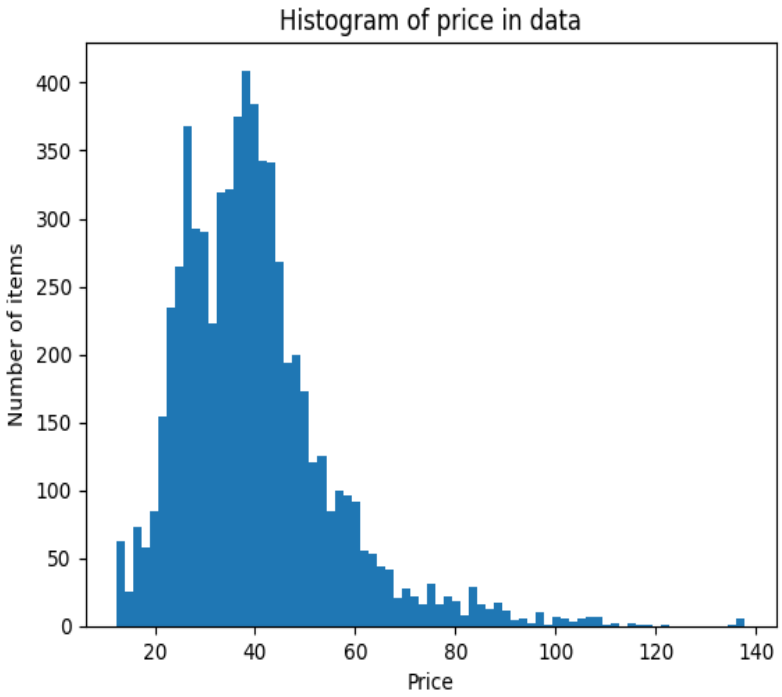


Figure 4: Price distribution

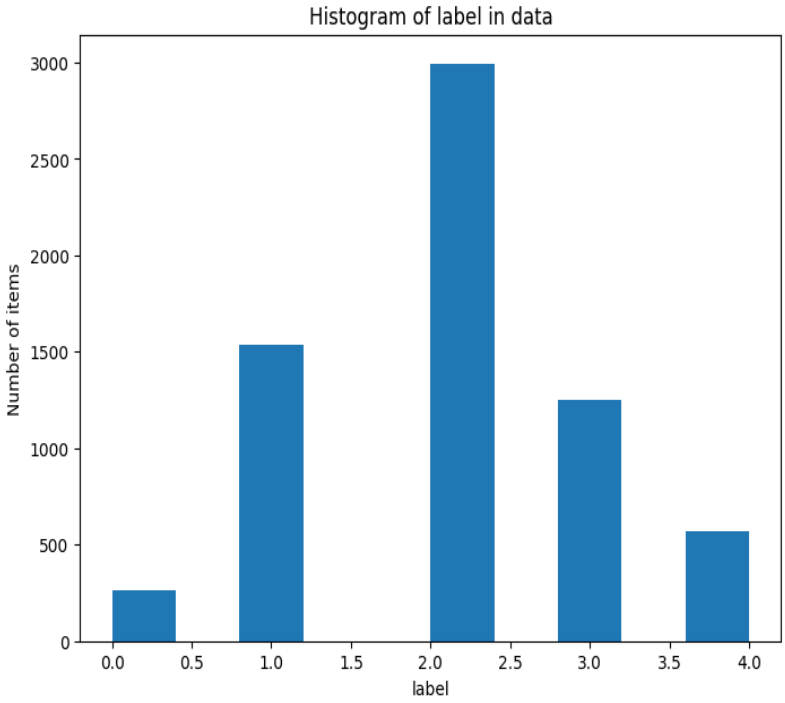


Figure 5: Price distribution

The data distribution is not balanced between the different *price_class*. This imbalance can lead to inefficiencies and inaccuracies in machine learning models. To overcome this challenge, we experimented with the oversampling technique.

Next, we analyze the impact of several features on the *price_class*.

The majority of sellers in the dataset offer houses with two or three bedrooms. This indicates a significant demand for houses with a limited number of bedrooms among buyers. Consequently, it can be inferred that most of the residents in the purchased houses are nuclear families, which typically consist of a small number of people. This preference for houses with fewer bedrooms underscores the housing needs and living arrangements of the majority of buyers in the dataset.

3.4 Feature engineering

3.4.1 Data importance and feature selection

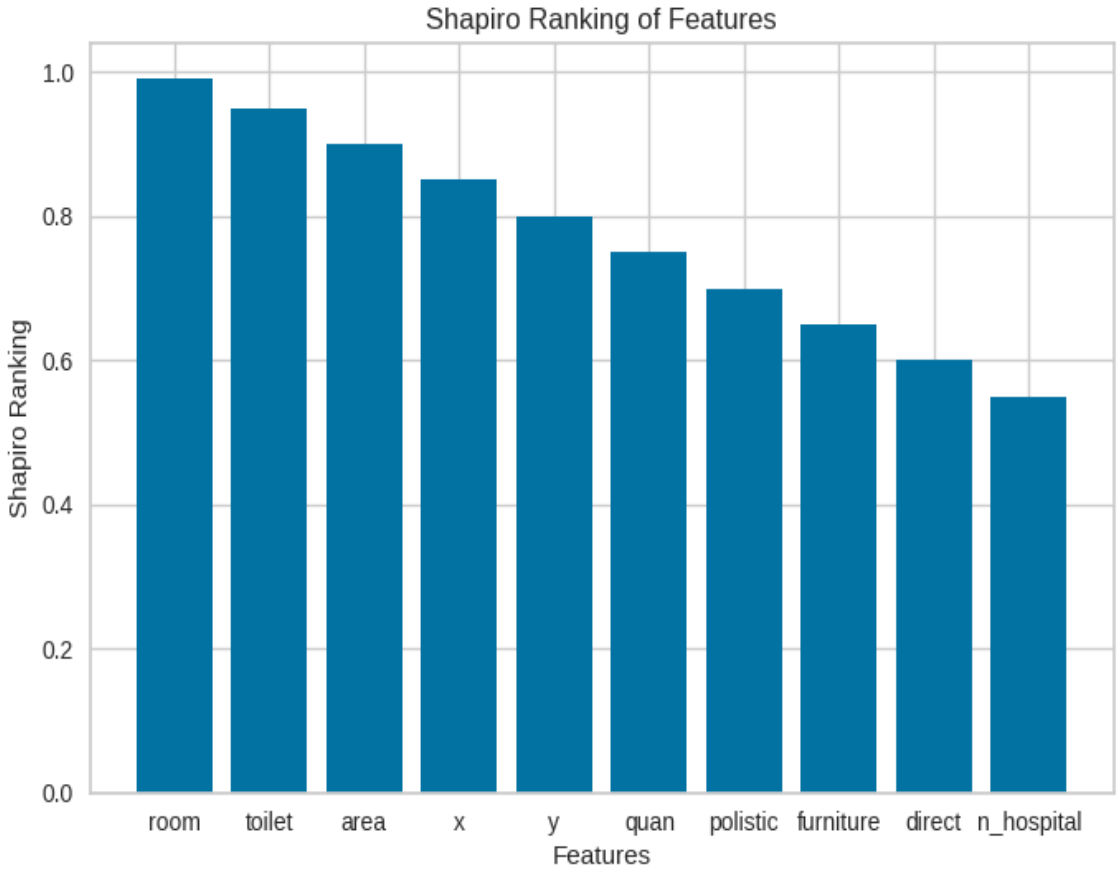


Figure 6: Ranking feature

4 Models

4.1 Theoretical background

4.1.1 Random Forest

The random forest algorithm has proven to be a successful general-purpose classification method . It is a bagging method in Ensemble Learning, which combines several randomized decision trees and aggregates their predictions by averaging. The algorithm selects random features to train each decision tree. In practice, random forests have demonstrated excellent performance in classification tasks.

Breiman's random forest algorithm predicts values at x by combining multiple decision trees. This method offers several benefits: it can handle a large number of features and is less prone to overfitting compared to individual decision trees. Random forests are highly accurate and robust to outliers. Additionally, the feature importance calculation in random forests is a valuable tool for feature selection, aiding in feature engineering and reducing the model's complexity.

Algorithm 1: Breiman's random forest predicted value at $\mathbf{x}$.

Input: Training set $\mathcal{D}_n$, number of trees $M > 0$, $a_n \in \{1, \ldots, n\}$, `mtry` $\in \{1, \ldots, p\}$, `nodesize` $\in \{1, \ldots, a_n\}$, and $\mathbf{x} \in \mathcal{X}$.

Output: Prediction of the random forest at $\mathbf{x}$.

for $j = 1, \ldots, M$ **do**
  Select a_n points, with (or without) replacement, uniformly in $\mathcal{D}_n$. In the following steps, only these a_n observations are used.
  Set $\mathcal{P} = (\mathcal{X})$ the list containing the cell associated with the root of the tree.
  Set $\mathcal{P}_{\text{final}} = \emptyset$ an empty list.
  while $\mathcal{P} \neq \emptyset$ **do**
    Let A be the first element of $\mathcal{P}$.
    if *A contains less than* `nodesize` *points or if all* $\mathbf{X}_i \in A$ *are equal* **then**
      Remove the cell A from the list $\mathcal{P}$.
      $\mathcal{P}_{\text{final}} \leftarrow \mathit{Concatenate}(\mathcal{P}_{\text{final}}, A)$.
    else
      Select uniformly, without replacement, a subset $\mathcal{M}_{\text{try}} \subset \{1, \ldots, p\}$ of cardinality `mtry`.
      Select the best split in A by optimizing the CART-split criterion along the coordinates in $\mathcal{M}_{\text{try}}$ *(see text for details)*.
      Cut the cell A according to the best split. Call A_L and A_R the two resulting cells.
      Remove the cell A from the list $\mathcal{P}$.
      $\mathcal{P} \leftarrow \mathit{Concatenate}(\mathcal{P}, A_L, A_R)$.
    end
  end
  Compute the predicted value $m_n(\mathbf{x}; \Theta_j, \mathcal{D}_n)$ at $\mathbf{x}$ equal to the average of the Y_i falling in the cell of $\mathbf{x}$ in partition $\mathcal{P}_{\text{final}}$.
end
Compute the random forest estimate $m_{M,n}(\mathbf{x}; \Theta_1, \ldots, \Theta_M, \mathcal{D}_n)$ at the query point $\mathbf{x}$ according to (1).

$\mathbf{X}_i = (\mathbf{X}_i^{(1)}, \ldots, \mathbf{X}_i^{(p)})$, for any $(j, z) \in \mathcal{C}_A$, the CART-split criterion takes the

Figure 7: Random Forest Algorithm

4.1.2 K-nearest neighbors

K-Nearest Neighbor (KNN) is a supervised machine learning algorithm utilized for classification and regression tasks. It operates by classifying an object based on the majority vote of its neighbors, assigning the object to the most common class among its k nearest neighbors (where k is typically a small positive integer). When k equals 1, the object is assigned to the class of its nearest neighbor.

In classification, KNN determines the class membership of the object, while in regression, it predicts the property value for the given data point. The algorithm stores all available cases and classifies new cases using a similarity measure, such as distance functions.

KNN does have drawbacks, including the necessity to choose the appropriate value for k and its sensitivity to irrelevant features and data scaling. Nevertheless, it remains a simple and effective algorithm suitable for various problems.

In summary, KNN is an intuitive and straightforward algorithm that is easy to implement and performs well on small datasets. It excels when dealing with

complex and nonlinear decision boundaries between classes. Moreover, KNN can handle both continuous and categorical features with minimal data preparation, making it a suitable choice for this project.

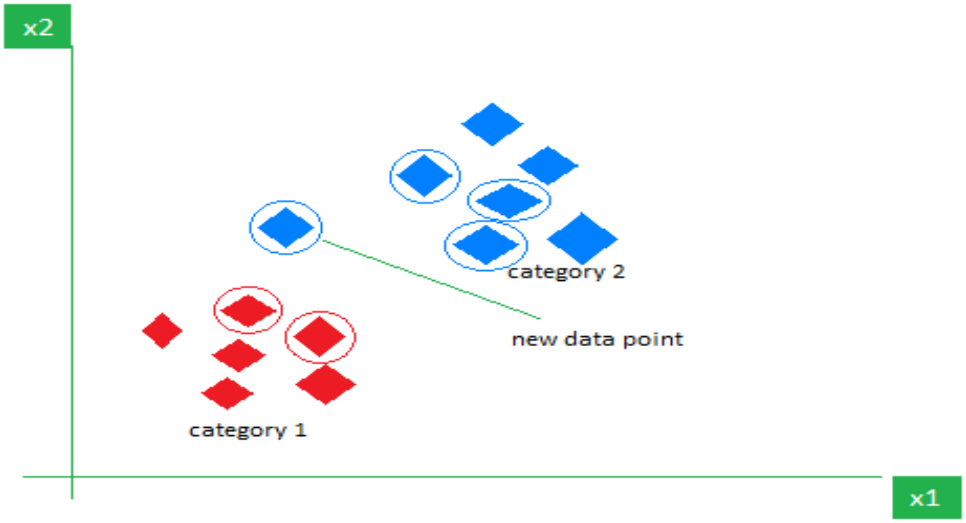


Figure 8: KNN demonstration image

4.1.3 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm utilized for both classification and regression tasks. In SVM, a hyperplane is constructed to separate the data into distinct classes. This hyperplane is selected to maximize the margin between the classes, defined as the distance between the hyperplane and the closest data points, termed as support vectors.

For binary classification problems, SVM aims to construct a hyperplane with the widest possible margin, known as the maximum margin classifier. In scenarios involving more than two classes, SVM adopts either a one-vs-one or one-vs-all strategy, constructing multiple hyperplanes to distinguish each class from the rest.

SVM is also applicable to regression problems, aiming to fit a continuous function that predicts the output value given an input.

Among its advantages, SVM excels in handling non-linearly separable data by utilizing kernel functions to transform the data into a higher-dimensional space, where a linear separator can be identified. Additionally, SVM performs well in high-dimensional spaces and exhibits relatively low generalization error.

However, SVM is sensitive to the choice of kernel function and hyperparameters, necessitating efficient algorithms to solve the optimization problems during model training. Despite these drawbacks, SVM remains a popular and effective method across various problem domains, particularly in high-dimensional spaces.

The selection of Support Vector Machine is due to its effectiveness in high-dimensional spaces, its capability to establish non-linear decision boundaries, its resilience against overfitting, and its versatility.

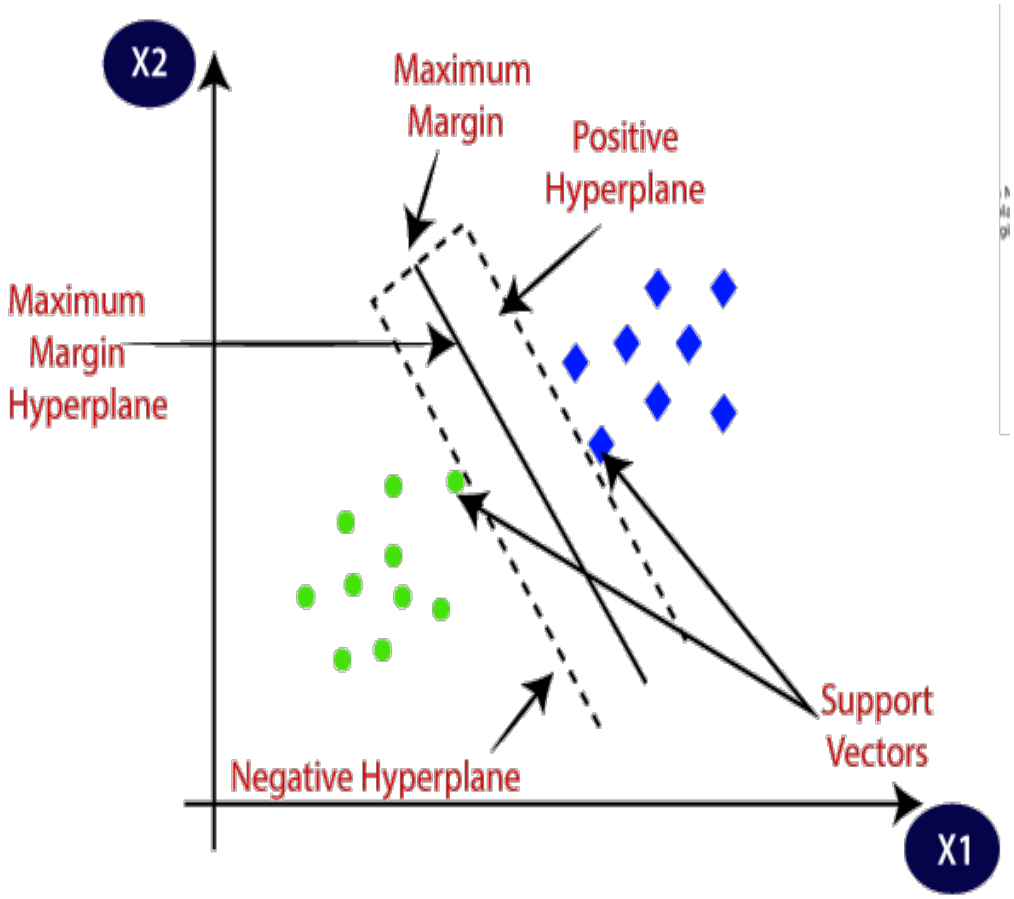


Figure 9: SVM demonstration image

4.1.4 eXtreme Gradient Boosting (XGBoost)

XGBoost, an open-source machine learning algorithm, is extensively employed in resolving regression and classification problems. It operates based on gradient boosting, an ensemble learning technique amalgamating multiple weak models to create a stronger one. The algorithm trains these weak models sequentially, with each subsequent model attempting to rectify the errors of its predecessor. Consequently, the final model is more robust and accurate compared to any individual weak model.

XGBoost boasts several noteworthy features making it apt for tackling house price prediction challenges. Firstly, it employs decision trees as its base model. Decision trees are straightforward to comprehend and interpret, adept at capturing intricate relationships between features and the target variable. Secondly, XGBoost is proficient at handling common real-world data issues like missing values and outliers. Lastly, it's engineered for swiftness and scalability, rendering it well-suited for large datasets.

Nonetheless, XGBoost has its limitations, including sensitivity to noisy data and a proclivity to overfitting. Addressing these constraints necessitates meticulous data preprocessing. Calibration of parameters is also essential to optimize XGBoost's performance, a task accomplished through extensive training iterations.

XGBoost emerges as a fitting choice for house price prediction due to its high accuracy, rapid training pace, and adeptness at handling sizable datasets. Grounded in boosting principles and utilizing decision trees as its foundation, XGBoost will be utilized in subsequent project phases to construct a predictive model for house prices. By accurately forecasting house prices, we can facilitate informed decision-making for potential buyers and aid sellers in determining optimal property prices.

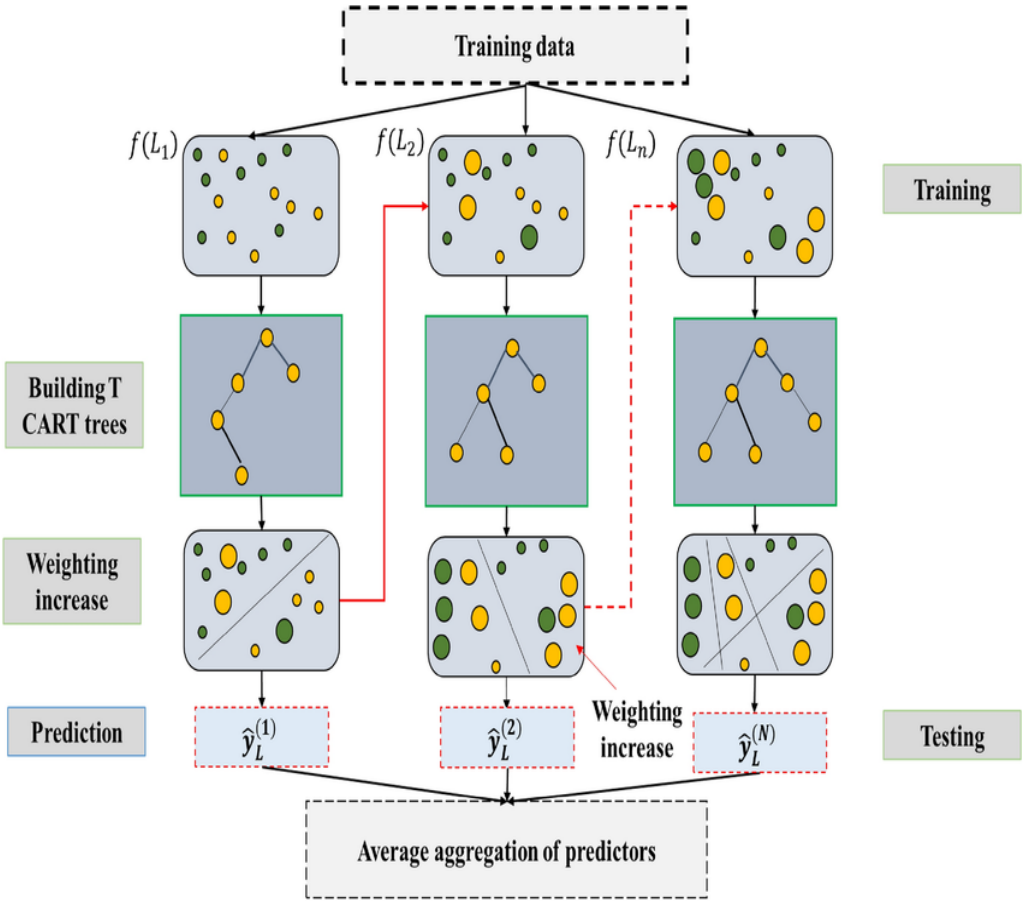


Figure 10: XGBoost demonstration image

4.1.5 Gradient Boosting

Gradient Boosting is a powerful machine learning algorithm used for both classification and regression tasks. It works by building a synthetic model from multiple weak decision tree models, each trained to correct the errors of the previous model. Gradient Boosting is capable of learning complex relationships in data and achieving high accuracy across a variety of data sets.

Algorithm:

1. Initialization: Start with a simple model, such as a small decision tree.

2. Repeat:

2.1. Training: Train a new decision tree model to correct the errors of the current model. Use the gradient of prediction error to determine the direction of improvement.

2.2. Update: Updates the weights of all models in the ensemble by increasing the weights of models with good performance and decreasing the weights of models with poor performance.

Inference: New prediction by combining predictions from all models in the ensemble, with their respective weights.

4.2 Practical Experiment and Results

4.2.1 Experiment (Hyper-parameter Tuning)

4.2.2 Random forest

We use K-fold (k =10) to evaluate the model. Finally, the best parameter for Random classifier: N_estimators = 150, criterion = entropy.

The model's accuracy is higher with a small range of error when utilizing the criterion. The optimal value for n_estimators appears to be 150, as it results in the smallest range error. Although increasing the number of estimators leads to a slight improvement in accuracy, it also entails a moderate increase in running time. This poses a significant challenge for our computer, which has limited computing power.

4.2.3 K-nearest neighbors

We employed Grid Search to determine the optimal parameter for K-nearest neighbors. This method is a brute-force approach involving the definition of a range of values for each parameter, followed by the evaluation of the model's performance for every possible combination of these parameters. Our analysis revealed that the best parameters for KNN are n_neighbors = 3 and metric = 'manhattan', as illustrated in the line chart

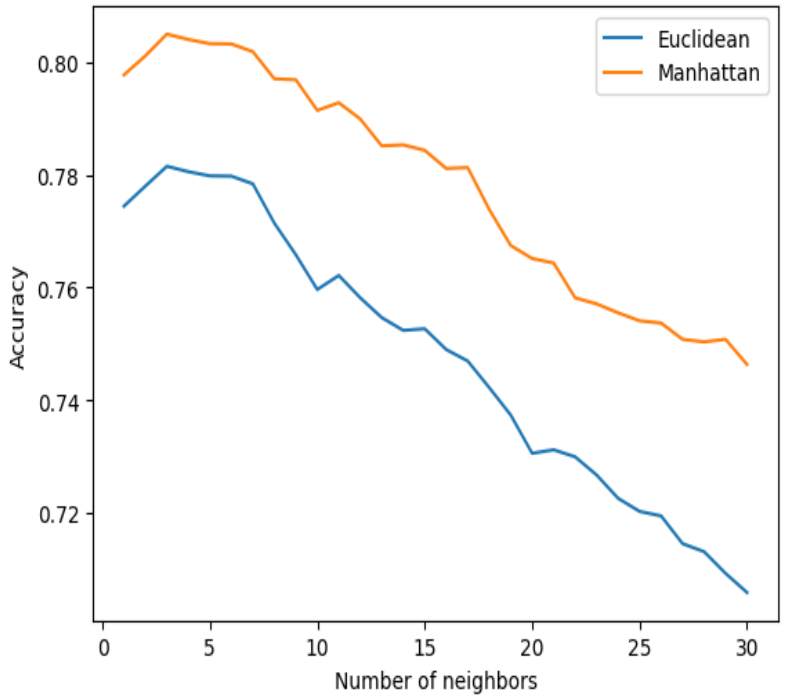


Figure 11: KNN Hyper-tuning demonstration image

4.2.4 Support Vector Machine

We utilized Grid Search to optimize the parameters of the Support Vector Machine (SVM). To conserve computational resources, it's standard practice to employ only a subset of the dataset, typically around 10%. The outcomes of the Grid Search revealed that the best parameters for this SVM are C=100 and gamma=10

4.2.5 eXtreme Gradient Boosting (XGBoost)

We applied the Grid Search method to identify hyperparameters for XGBoost. This method is robust and effective for evaluating and obtaining optimal results for the model. Our analysis yielded promising outcomes with n_estimators = 1000.

4.2.6 Gradient Boosting

We applied the Grid Search method to identify hyperparameters for Gradient Booosting. This method is robust and effective for evaluating and obtaining optimal results for the model. Our analysis yielded promising outcomes with max_depth = 8 and n_estimators = 150, learning_rate = 0.1, subsample = 1.

4.2.7 Results

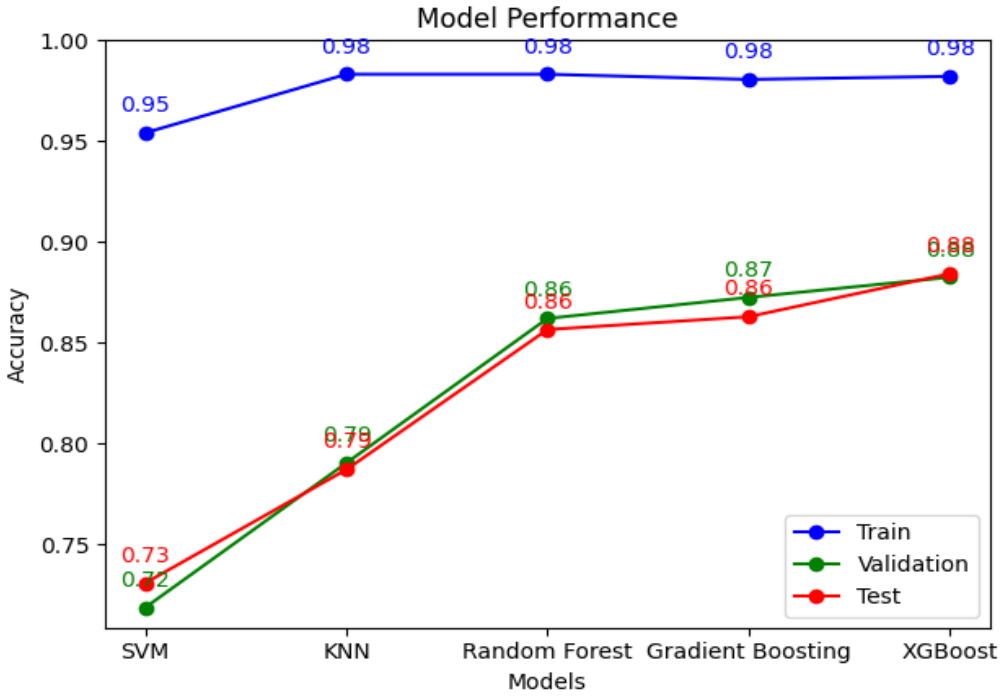


Figure 12: Comparing models

Based on the precision scores provided for different models, we can draw the following conclusion:

The data we cleaned exhibited an imbalance in the target column, as noted in the Data Analysis section. To address this, we applied an upsample method for each class using Sklearn's built-in function. This adjustment significantly improved the accuracy and precision scores of our models.

Among the machine learning methods tested, XGBoost demonstrated the best performance with a precision score of 0.8842, outperforming other models such as SVM (0.7308), KNN (0.7871), Gradient Boosting (0.8629), and Random Forest(0.8566). This superior performance of XGBoost is notable, particularly considering its fast training times and high precision.

While models like KNN, SVM, and Gradient Boosting also performed well, they did not match the precision achieved by XGBoost. In terms of deep learning models, although they reached a good performance threshold, we believe their potential could be further realized with more complex neural network architectures. However, due to resource limitations, we were unable to tune the parameters or increase the complexity of the neural networks to potentially surpass the boosting models' performance.

In summary, XGBoost stands out as the most effective model among those tested, combining high precision with efficient training times. Given its strong performance and practical advantages, we have chosen XGBoost as the representative model for further analysis in the subsequent sections.

5 Conclusion

This study employs machine learning techniques to develop a price prediction model for the housing market, using a large, publicly available dataset of real estate transactions. Various classification models are compared against each other and a benchmark model. The findings show that the XGBoost algorithm with target binning surpasses all other models in every evaluated metric, including the coefficient of determination, F1 score, precision, recall, and accuracy. Due to the complexity and multitude of influencing factors, achieving highly accurate results with common analytical data types remains challenging. This area continues to offer significant research potential for future exploration.

6 Contribution

Nguyen Thanh Minh:
- Code: K-nearest neighbors ,Support Vector Machine
- Report: 20%
- Other : Crawl data : 20%

Nguyen Trung Dung:
- Code: K-nearest neighbors ,Support Vector Machine
- Report: 20%
- Other : Crawl data : 40%, Feature engineering, Data Analysis

Tran Quoc Khanh:
- Code: XGBoost, Gradient Boosting, Random Forest
- Report: 20%
- Other : Crawl data : 40%, GUI

Bui Anh Duong:
- Code: K-nearest neighbors
- Slide: 100%

Nhu Minh Ha :
- Code: Gradient Boosting, Random Forest
- Report: 20%, Data Preprocessing

Bui Thi Thu Uyen:
- Code: XGBoost
- Report: 20%

References

[1] Biau, Gérard, and Erwan Scornet. "A random forest guided tour." *Test* 25 (2016): 197-227.

[2] Kiel, Katherine A., and Jeffrey E. Zabel. "Location, location, location: The 3L Approach to house price determination." *Journal of Housing Economics* 17.2 (2008): 175-190.

[3] Wen, Haizhen, Yan Zhang, and Ling Zhang. "Do educational facilities affect housing price? An empirical study in Hangzhou, China." *Habitat International* 42 (2014): 155-163.

[4] Hao, Jinlian, and Haitao Ma. "Spatial heterogeneity of public service facilities in the living circle and its influence on housing prices: a case study of central urban Dalian, China." *Land* 11.7 (2022): 1095.

[5] Park, Byeonghwa, and Jae Kwon Bae. "Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data." Keimyung University, April 2015.

[6] Christopher, Antony. "K-Nearest Neighbor. A complete explanation of K-NN." Medium, 2 February 2021. Available at: `https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4`.

[7] Shah, Rajvi. "Support Vector Machine — Introduction to Machine Learning Algorithms Supervised ML Algorithm: Support Vector Machines (SVM)." Medium, 14 January 2021. Available at: `https://medium.com/analytics-vidhya/supervised-ml-algorithm-support-vector-machines-svm-fb674430ab74`.

[8] Chen, Tianqi, and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016): 785-794.