Predicting Housing Rental Prices using Machine Learning

Bouabdallah Khaled, Idir Abir, Al Rifai Asma, El Hadji Malick SY

Université Paris Cité

2023-2024

Abstract

Given the intricate nature of factors influencing housing prices, adopting machine learning techniques is essential. This project focuses on developing a machine learning prediction model tailored for accurately forecasting future housing prices and assisting both real estate agents and potential renters in making well-informed decisions. By leveraging various ML algorithms namely Random Forest Regressor, Gradient Boosting Regressor, SVR, Linear Regression, Ridge, Lasso, ElasticNet, KNeighbors Regressor, and Decision Tree Regressor, we seek to enhance the accuracy of prediction models. We used a dataset obtained through web scraping and the Google Maps API, comprising 7524 observations and multiple variables, with the target variable being the price. Evaluation metrics such as mean absolute error (MAE), root mean square error (RMSE), and adjusted R-squared value are employed to assess models performances. Our experiments reveal that the XGBRegressor outperforms other traditional models, exhibiting the lowest MAE and RMSE, and the highest adjusted R-squared value.

Key words: Machine learning, Housing price prediction, XGBRegressor

1 Introduction

The housing market stands as one of the most pivotal sectors influencing both the economy and individuals' daily lives. In recent years, the field of house renting prices holds particular significance, reflecting not only the fundamental need for shelter but also serving as a barometer of economic health and societal trends

Indeed, finding the accurate estimations of house renting prices has long been a challenge. From the perspective of both renters and landlords, the ability to set fair rental prices is essential for making informed decisions, whether it be for maximizing returns on property investments, or simply finding affordable and reasonable housing options.

What makes this project interesting is the potential of big data analytics to revolutionize the process of estimating house renting prices. The sheer volume and variety of data available in today's digital landscape offer unprecedented opportunities to uncover hidden patterns within it, refine predictive models, and hence enhance the accuracy of rental price estimations.

However, estimating house renting prices is no trivial task. Real estate markets are complex and always changing. There are so many things that can affect rental prices, like property characteristics and location-specific trends and even the state of the economy. All these factors complicate the predictive process. Thus, developing robust and reliable models capable of accurately forecasting rental prices represents a big challenge, demanding sophisticated methodologies and rigorous validation.

Overall, this project is about tackling the challenge of estimating house renting prices and contributing to a more informed and equitable real estate landscape through big data analysis. Indeed, predicting better rental prices can help people find homes they can afford and help landlords charge fair prices.

2 Related Work

The different articles we read tackle two major fields related to our work: Housing price forecasting, and Data streaming.

2.1 Prediction of house market

In examining the following papers on housing price prediction, we encounter a diverse array of methodologies and findings.

1 Machine Learning, Deep Learning, and Hedonic Methods for Real Estate Price Prediction

2 Housing Price Prediction via Improved Machine Learning Techniques

3 Real Estate Price Prediction Using Machine Learning

4 Estimating Warehouse Rental Price using Machine Learning Techniques

5 Beyond Spatial Auto-Regressive Models: Predicting Housing Prices with Satellite

6 Advanced Machine Learning Algorithms for House Price Prediction: Case Study in Kuala Lumpur

7 Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data

8 Housing price prediction incorporating spatio-temporal dependency into machine learning algorithms

Article 1[20] is focusing on property appraisal models in Boulder, Colorado, employs machine learning and deep learning techniques, showcasing the superiority of Random Forest in capturing housing market complexities. It delves into the non-linear association between house features and prices, revealing the limitations of traditional hedonic price regression models in capturing housing market complexities.

In contrast, Article 2[18] deals with the Housing Price in Beijing. It explores the impact of feature engineering techniques and regression models on housing price prediction, showcasing the superiority of Hybrid Regression and Stacked Generalization Regression over traditional methods like Random Forest. The comparison of regression techniques and advanced ML algorithms can inform similar studies in article [6].

The article 3[12]outlines a project aimed at predicting real estate prices for properties in Bengaluru using machine learning algorithms like linear regression, decision tree, random forest, and XGBoost. It emphasizes the importance of accurate price prediction in real estate transactions due to the complexity of factors influencing property values. The project proposes a comprehensive approach involving data preprocessing, model training, evaluation, deployment, and user interface development. It aims to provide a user-friendly platform for predicting property prices based on key features such as location, size, and amenities. Overall, the project highlights the feasibility of leveraging machine learning techniques to forecast real estate prices, with implications for the broader real estate market in Bengaluru.

The article 4[21] also uses datasets based in Beijing and explores warehouse rental pricing dynamics, highlighting the effectiveness of tree-based models and identifying factors such as distance from the city center as significant influencers. Indeed, it emphasizes that Random Forest outperform linear regression approaches, what correlates with the findings of Article [1], emphasizing the importance of location-based factors in pricing predictions.

On the other hand, the Article 5[4] introduced the fact that the integration of satellite imagery features into housing price estimation models offers a novel approach to capturing spatial information and improving prediction accuracy. By utilizing Deep Convolutional Neural Networks (DCNNs) to process satellite imagery combined with the points of interest in the neighborhood, the study showcases the potential of advanced image processing techniques in predictive modeling. The integration of satellite imagery features can be relevant for studies aiming to enhance housing price prediction, such as those in Articles [1] and [6].

Similarly, Article 6[19] compares traditional regression approaches with advanced ML algorithms in Kuala Lumpur's housing market context. By incorporating additional attributes like house size and proximity to amenities, the study assumes that XGBoost-based models being the most powerful algorithms for any regression or classification. Indeed, they emerge as superior predictors and outperform traditional regression methods, underscoring the importance of feature selection and model complexity.

Article 7[5] merges real estate, school ratings, and mortgage rate data to predict real estate trends in Fairfax County, offering a comprehensive analysis of predictive modeling techniques. Through the employment of machine learning classifiers like C4.5 and RIPPER, the study identifies the most effective models for predicting real estate trends in townhouses, highlighting the systematic performance evaluation methodology used.

Lastly, Article 8[2] investigates housing price variations in Adelaide, Australia, considering spatio-temporal dynamics. They used a 32 years long dataset that includes property attributes and neighborhood quality. Their research affirms that ML models like Random Forest and Gradient-Boosted Tree outperform traditional regression methods, especially when accounting for spatio-temporal dependencies. Key findings also indicate the importance of addressing non-linear effects and the efficiency of incorporating a spatio-temporal lag variable to improve predictive accuracy.

Paper	Utilized models
1	Hedonic Price Regression, AI Network Random Forest, KNN
2	Random Forest, XGBoost LightGBM, Hybrid Regression Stacked Generalization
3	linear regression, decision tree random forest, XGBoost
4	Linear Regression, Regression Tree Random Forest Gradient Boosting Regression Trees
5	DCNNs
6	Multiple Regression Analysis Ridge Regression LightGBM, XGBoost
7	C4.5, RIPPER, Naïve Bayesian AdaBoost
8	Multiple Linear Regression Gradient-Boosted Tree Decision Tree, Random Forest

Collectively, these articles deepen our understanding of housing market dynamics and predictive modeling techniques, aiding in selecting the most appropriate model based on available attributes and the features utilized in our study. Each one of them offers unique insights into factors influencing housing prices and inform future research and decision-making by evaluating predictive models' effectiveness.

2.2 Data Streaming

Data streams are instrumental in enabling real-time data analysis and empowering timely decision-making. Adapting machine learning to handle streaming data allows for swift forecasting, ensuring competitiveness in a dynamic digital environment. The following list comprises the articles we explored on this topic :

9 Learning from Data Streams: An Overview and Update

10 Time series big data: a survey on data stream frameworks, analysis and algorithms

11 An analysis of technological frameworks for data streams

12 Kafka-ML: Connecting the data stream with ML/AI frameworks

Article 9[11] discusses the need for reevaluation and updates in supervised data-stream learning methodologies, particularly focusing on concept drift and temporal dependence. It emphasizes the importance of addressing tasks beyond classification, such as regression, and suggests exploring reinforcement learning and neural transfer learning for dynamic environments. Recommendations include exploring reinforcement learning and neural transfer learning for dynamic environments.

The second article 10[3] emphasizes the significance of real-time processing of time series data for forecasting, monitoring, and anomaly detection. It discusses the importance of Stream Processing Engines (SPEs) and the need for forecasting and anomaly detection algorithms. Stream Processing Engines (SPEs) are highlighted for real-time big data processing.

We also have article 11[9] that analyzes open-source technological frameworks for data streams, focusing on factors such as the existence of data mining libraries, documentation availability, platform maturity, fault tolerance, and performance. The comparison of performance and latency between various frameworks such as Spark Streaming, Flink, etc., is discussed. The analysis provides insights into the characteristics of different open-source technological frameworks for data streams, aiding in decision-making for their selection.

Last but not least, article 12[7] introduces Kafka-ML, an open-source framework designed to manage ML/AI pipelines effectively in continuous data stream environments. It focuses on integrating ML/AI pipelines with data streams and supporting AutoML features. This framework facilitates their integration with data streams. Finally, It addresses challenges faced by ML/AI developers, offering a user-friendly interface and supporting AutoML features, aiming to democratize ML/AI.

In summary, while each article contributes unique insights and findings to the field of data stream processing and machine learning, there are common themes such as the importance of real-time processing, anomaly detection, and the development of frameworks for managing ML/AI pipelines in continuous data stream environments.

3 Problem Definition

Predicting house renting prices is a multifaceted challenge, requiring a nuanced understanding of the complex interplay between diverse housing attributes and market dynamics. The task involves developing a machine learning model capable of accurately estimating rental costs based on features such as property size, location, and neighborhood characteristics. Successfully navigating these nuances necessitates robust data preprocessing, feature engineering, and model selection techniques to build a predictive framework that can generalize well and provide reliable estimates in a dynamic housing landscape.

Therefore, in this project, we aim to address the key problematics related to leveraging big data for housing market analysis and prediction.

3.1 Data collection

How can we efficiently extract the needed data about houses from websites on the internet?

3.2 Data storage

Once data is extracted, what data management tool to use and how should it be stored to facilitate further analysis and model development?

3.3 Data preprocessing

What strategies should be employed to clean the extracted dataset and handle missing and abnormal values?

3.4 Feature selection

What contextual factors beyond traditional house attributes can we use to enhance our dataset to better understand housing market dynamics? Furthermore, what methods can be employed to acquire these additional features?

3.5 Model selection

Given the preprocessed dataset, how can we identify the most suitable machine learning model for Training and then forecasting housing prices?

3.6 Model evaluation

To what extent can the results of data analysis on house data reliably predict housing prices?

4 Solution

In this section, we will comprehensively explore the approaches we employed to address the underlying problem statement. Each approach will be dissected, highlighting its rationale, methodology, and relevance to effectively tackle the identified issues

4.1 Data collection

Our solution encompasses retrieving rental price data, specifically characteristics, from the following website: locservice.fr[*]. This website serves as a comprehensive platform offering rental options ranging from houses, apartments, and studios to rooms across the metropolitan French territory.

We employed web scraping techniques to extract relevant data about rental prices and the different characteristics from locservice.fr. This approach allowed us to gather comprehensive information efficiently from the website's listings.

We began our navigation by accessing the website's homepage, and then proceeded to explore the house listings by department through the location map. Upon selecting the desired city, we were directed to the web page showcasing all available rental properties.

- **Scrapy:** We used it as a scraping tool because it stands out as a top choice for web data extraction due to its comprehensive features and robust capabilities. As an open-source framework, it offers accessibility without any licensing costs. Its efficiency is notable, allowing for concurrent requests and asynchronous operations, enabling rapid scraping of large datasets. Scrapy's native support for HTML and XML extraction, coupled with seamless integration of CSS and XPath selectors, simplifies the process of pinpointing and retrieving specific data elements. Moreover, its extensibility enables customization and extension of functionality, while built-in tools streamline common scraping tasks.

- **Splash:** it is a powerful tool for web scraping, offering several advantages. Firstly, it supports JavaScript rendering, enabling it to handle dynamic content and interact with websites like a real browser. This capability ensures comprehensive data extraction from modern web pages. Additionally, it offers features for customizing requests, handling cookies, and managing sessions, enhancing scraping efficiency and reliability. Moreover, Splash's asynchronous processing capabilities enable parallel execution of multiple scraping tasks, improving performance and scalability.

4.2 Data storage

To manage the retrieved data effectively, we opted for Postgres as our database management system.

- **Postgres:** being free and open source, offers scalability for large datasets and complex queries, making it ideal for our needs. Its reliability and stability ensure data integrity, even in high-transaction environments. Additionally, its flexibility allows for customization to optimize data storage and retrieval processes.

4.3 Data preprocessing

- Initially, our dataset comprised 10029 entries and 17 variables that consists of: id, department name, city name, title, type, latitude, longitude, price, area, availability, furniture, energyDPE, energyGES, description, features, and url.
- Upon visualizing the dataset, we observed numerous missing values and incomplete attribute values. For example, some houses lacked classifications for "energyDPE" and "energyGES", denoted as "En cours" or "Vierge".
- We began by filling missing values in the house "type" variable, extracting that specific information from the title, "title" represents the house advertisement title. Subsequently, we dropped rows with missing values in latitude, longitude, features, and type variables. This step left us with 7524 entries.
- The "type" variable, consisting of four categories - "Chambre", "Studio", "Appartement", and "Maison" - was converted into ordinal type: 0, 1, 2, 3 respectively. Similar transformations were applied to the "energyDPE" and "energyGES" variables, assigning "En cours" and "Vierge" to the middle value "D".

[*]locservice.fr: https://www.locservice.fr/

- For categorical data such as "Features" and "Furniture", we performed one-hot encoding, creating new binary columns for each category. A value of 1 indicates the presence of the category, while 0 indicates its absence. This process resulted in 23 new columns, bringing the total to 40 variables.
- Lastly, variables deemed unnecessary for the prediction process, such as "department number" and "url", were removed, resulting in a fully prepared dataset of 7524 entries and 32 columns.

4.4 Feature engineering

To enrich our dataset and capture relevant contextual information, we integrated the Google Maps API. This API facilitated the retrieval of points of interest (POIs) data, which encompass locations on the map with economic, social, or cultural significance. These POIs include establishments such as schools, restaurants, shopping centers, train stations, subway stations. Each POI is characterized by its latitude and longitude coordinates, a name, and a place type, which collectively provide valuable insights into the neighborhood surrounding each house.

Instead of just counting the number of POI's. We also applied weighted sum using the distance between our query point (home) and the POI. We experiment with multiple weighting techiques to show the effect of feature engineering on the final prediction, here are the methods that we used:

Inverse Distance Weighting:

$$w_i = \frac{1}{d_i^k}$$

This method assigns higher weights to points closer to the reference point. The rate at which the weight decreases with distance is controlled by k. It is commonly used when closer points are expected to have a stronger influence.

Gaussian Distance Weighting:

$$w_i = \exp\left(-\frac{d_i^2}{2\sigma^2}\right)$$

This method models a Gaussian distribution to assign weights. The weights fall off exponentially with distance, where the spread is controlled by σ. This is useful when you want a smooth, bell-curve-like decay in weights with distance.

Logarithmic Distance Weighting:

$$w_i = \frac{1}{\log_{\text{base}}(d_i + 1)}$$

This method assigns weights that decrease logarithmically with distance, providing a slower decline in weight compared to other methods. It can be used when you want a gentle drop-off, emphasizing a broader range of distances.

Linear Distance Weighting:

$$w_i = \max(0, \text{max\_distance} - d_i)$$

This method decreases weights linearly with increasing distance. Beyond a certain point, defined by max_distance, the weight becomes zero. It's useful when you need weights to fall off at a constant rate and then drop to zero at a specified distance.

Total Score:

$$\text{total\_score} = \sum_{i=1}^{n} w_i$$

The total score is the sum of weights across all points, reflecting the cumulative effect of different weighting methods. This score can be used to evaluate or compare the influence of different distance-based factors.

4.4.1 API Call

Our methodology involved calling the Google Maps API and iterating over our dataset to extract POI information for each house entry (row). Our focus was primarily on identifying key transportation hubs, namely "train stations," "subway stations," "bus stations," and "bus stops," we constrained our search to the 20 most popular POI locations within a 2km radius to each house. This approach ensured that we captured relevant transportation facilities crucial for assessing accessibility and convenience for potential residents. Figure 1 shows the usage of the API.

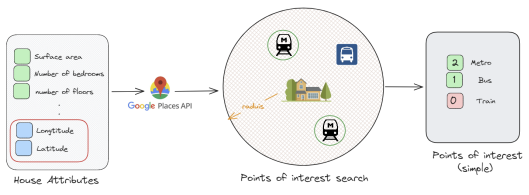

Figure 1: Google Places API call for collecting nearby transportation stations

4.4.2 Feature extraction and DataBase storage

Upon retrieving the POI data, we efficiently saved the extracted features, including the distance from each house to the specific POI location in a database comprising a total of 135,623 entries.

4.4.3 Data preprocessing

We conducted one-hot encoding on the "Type" variable to get specific POI types namely: transit stations, bus stops, train stations, bus stations, and subway stations. This encoding facilitated the representation of categorical data, ensuring efficient processing and analysis. Subsequently, we merged the datasets of houses and POIs based on their corresponding IDs and assigned both basic and weighted scores to each house.

- **Basic score:** reflects the number of each POI type found nearby. For instance, if a house had 5 bus stations nearby, the variable representing bus stations would be assigned the value of 5.

- **Weighted score:** in contrast to the basic score, here we take into account the distance from the house to the POI.
  To calculate the weighted score, we explored several functions, including inverse weight, Gaussian weight, and linear weight.

By following this structured approach, we effectively leveraged the Google Maps API to augment our dataset with pertinent POI information, particularly focusing on transportation-related facilities. This methodological framework ensured the inclusion of crucial contextual factors essential for understanding housing market dynamics and informing real estate decisions.

4.5 Model selection

We employed a diverse set of machine learning algorithms specifically crafted for forecasting purposes. In this section, we will delve into each of these models:

4.5.1 Random Forest Regressor

Random Forest Regressor is a versatile and powerful machine learning algorithm used for regression tasks. It belongs to the ensemble learning family and is based on the concept of decision trees. The Random Forest algorithm creates a multitude of decision trees during training and merges their predictions to obtain more accurate and stable results. Here's a brief overview of how Random Forest Regressor works:

- **Ensemble of Decision Trees:** Random Forest Regressor consists of a collection of decision trees, each trained on a random subset of the training data and a random subset of features. This randomness helps prevent overfitting and makes the algorithm robust to noise.
- **Decision Making:** During prediction, each decision tree in the ensemble independently predicts the target variable based on the input features. The final prediction is then obtained by averaging (for regression) or taking a vote (for classification) across all the trees in the forest.
- **Feature Importance:** Random Forest Regressor can also provide insights into feature importance, helping identify which features have the most significant impact on the target variable.

4.5.2 Gradient Boosting Regressor

GradientBoostingRegressor is an ensemble learning algorithm that iteratively combines weak learners, typically decision trees, to create a powerful predictive model. It sequentially corrects errors made by previous models, resulting in a robust and accurate predictor. Here are few advantages of using it :

- **High Predictive Accuracy:** Excels in capturing complex relationships and delivering accurate predictions.
- **Robustness:** Handles outliers and noisy data effectively due to its ensemble nature.
- **Feature Importance:** Provides insights into feature importance, aiding in understanding the driving factors behind predictions.

Use cases for the GradientBoostingRegressor can be:

- **Regression Tasks:** Widely used for predicting continuous target variables like house prices or stock prices.
- **Complex Data:** Suited for datasets with intricate relationships between features and target variables.

4.5.3 SVR

Support Vector Regression (SVR) is a machine learning algorithm used for regression tasks. It aims to find the best-fitting line or hyperplane that separates data points by maximizing the margin while minimizing prediction error. SVR handles non-linear relationships using kernel functions and is adept at capturing complex patterns in data. Tuning hyperparameters like kernel function and regularization parameter is essential for optimal performance. Overall, SVR is a versatile algorithm applicable to various regression problems.

4.5.4 Linear Regression

Linear Regression is a basic yet powerful machine learning algorithm used for regression tasks. It models the relationship between input variables and a target variable by fitting a straight line to the data. Despite its simplicity, linear regression provides interpretable insights and is computationally efficient. However, it may not perform well with non-linear relationships.

4.5.5 Ridge

Ridge regression is a form of linear regression that addresses multicollinearity by adding a penalty term to the regression equation. This penalty term, controlled by a tuning parameter, helps prevent overfitting by shrinking the coefficients. Ridge regression is effective for high-dimensional datasets and improves model stability and interpretability.

4.5.6 Lasso

Lasso, or Least Absolute Shrinkage and Selection Operator, is a regression technique that uses L1 regularization to shrink the coefficients and perform feature selection. Unlike ridge regression, Lasso tends to produce sparse models with fewer non-zero coefficients, making it useful for high-dimensional datasets. While it helps prevent overfitting and improves model interpretability, Lasso may exhibit instability when features are highly correlated. Nonetheless, it remains a valuable tool for regression tasks, particularly in scenarios requiring feature selection.

4.5.7 ElasticNet

ElasticNet combines Lasso and ridge regression techniques, offering flexibility in balancing feature selection and coefficient shrinkage. By tuning the mixing parameter, it allows for control over regularization, making it versatile for high-dimensional datasets with multicollinearity. ElasticNet is a valuable regression method for achieving both sparsity and model accuracy.

4.5.8 KNeighbors Regressor

The KNeighbors Regressor is a simple yet effective algorithm for regression tasks. It predicts the target value for a new data point by averaging the target values of its k nearest neighbors in the training set. While it's easy to implement and can capture complex relationships, it may face challenges with large datasets and high-dimensional spaces due to its memory-intensive nature. Nonetheless, it remains a valuable tool for capturing local patterns and non-linear relationships in the data.

4.5.9 Decision Tree Regressor

The Decision Tree Regressor partitions the feature space into smaller regions to predict target values. It selects the best feature to split the data at each step, aiming to minimize variance within each region. While it's interpretable and can capture non-linear relationships, it's prone to overfitting, especially with deep trees or noisy data. Nonetheless, it's a versatile algorithm for regression tasks, offering a balance between interpretability and predictive performance.

4.6 Model evaluation

The accuracy of models is evaluated using the root mean square error (RMSE), R-squared score, and mean absolute error (MAE) on the split train and test dataset.

5 Experiments

5.1 Hardware

For this data science project, we used vast.ai [†], a cloud-based platform that allows users to rent virtual machines with GPU capabilities. The specifications for the cloud instances used during our experiments can be seen in table 1.

Table 1: Cloud Instance Specifications

Component	**Specification**
Cloud Platform	vast.ai
GPU	Nivida RTX 2080 Ti
MAX CUDA	12.1
CPU	Intel Xeon® E5-2680 v3
RAM	16 GB DDR4
Storage	128 GB SSD
Operating System	Ubuntu 20.04 LTS

5.2 Software Tools

Multiple libraries were used in our experiments, this is the list of the most important ones:

- pandas [14] : Powerful data manipulation and analysis library for Python.
- NumPy [8] : Foundation for numerical computing in Python.
- PyTorch [15] : Popular deep learning framework.
- XGBoost [6] : Efficient gradient boosting implementation.
- Plotly [17] : Interactive visualization library for creating plots and charts.
- Matplotlib [10] : Widely used library for creating static visualizations.
- scikit-learn (sklearn) [16] : General-purpose machine learning library (potentially for preprocessing or other models).
- Optuna [1] : Library for hyperparameter optimization to fine-tune models.
- cuPy [13] : Provides GPU acceleration for NumPy-like operations.

5.3 Dataset

This study utilizes the dataset compiled through web scraping, comprising renting listings in France as of April 2024. The initial Scrapping results results in **10029** rent advertisement. Following prepossessing steps, the finalized dataset contains **7524**. We created **7** variations of the dataset using the extra data that was collected from *Google Places API*. The datasets were created by combining original features of the listings and costume features that we discussed in section

[†]Vast.ai: https://vast.ai/

4.4, all datasets have a singular target variable *price*. Below you can find table 2 that contains the datasets descriptions.

5.3.1 Normalization

We opted for Max absolute scaling because it effectively handles scenarios where there's a mix of sparse and dense columns and we want to apply a single transformation to all columns, it is crucial to choose a transformation that doesn't inadvertently increase data density or change the nature of sparse data. The goal is to maintain sparsity while achieving the desired scaling or normalization effect. We obtain the scaling using the following formula:

$$X_{\text{scaled}} = \frac{X}{\max(|X|)}$$

5.4 Fine-tuning

We employed Bayesian search, a technique that iteratively explores the hyperparameter space while favoring regions with promising performance based on past evaluations which are ranked by Mean Absolute Error on the validation set, to identify the optimal hyperparameters for each model. This approach offered advantages over methods like grid search due to its efficient exploration capabilities.

For most models, we utilized K-fold cross-validation (KFold) with $K = 5$ to ensure robust hyperparameter selection. In KFold, the training data is split into K folds. The model is trained on $K - 1$ folds and validated on the remaining fold. This process is repeated K times, providing a more comprehensive evaluation compared to a single split.

However, for the Multi-Layer Perceptron (MLP), we opted for a simpler validation approach due to potential drawbacks of KFold in certain MLP configurations. Here, we used a validation set comprising 10% of the training data for evaluation.

The Bayesian search was executed for 25 iterations, striking a balance between exploring the hyperparameter space and achieving convergence within reasonable computational constraints. We repeat the whole process for each dataset of the 7 datasets that we have.

To analyse the performance of the Bayesian search and the effect of hyperparameters on the models, we created multiple visualization, from optimization history, contour plots to hyperparameters importance plots. Figures 2, 4 and 3 are shown here which correspond to the Bayesian search of the best XGBoost model (best model overall) on the test set.

5.5 Evaluation

We used three separate metrics to evaluate our models that were yield from the Bayesian search. each metric

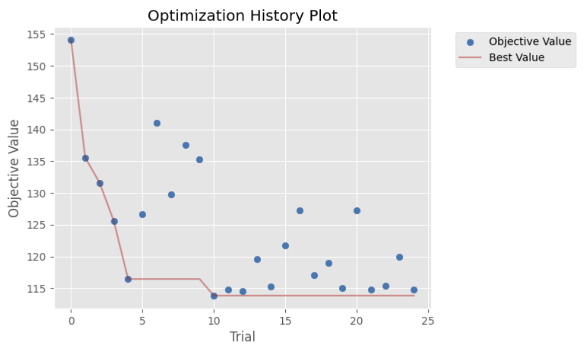

Figure 2: Validation MAE Bayesian search evolution

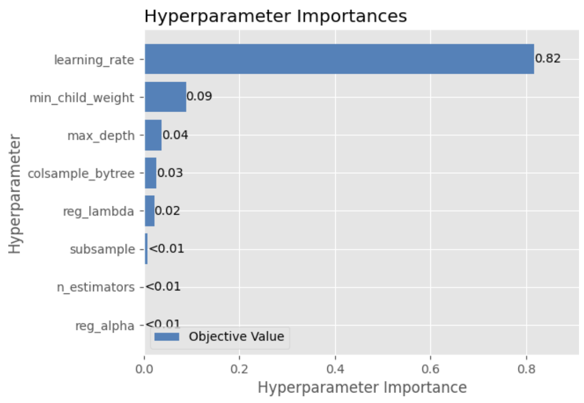

Figure 3: Hyperparamater Importance

is applied on the test set. which represents 20% of the original data set.

R-squared (R²): This metric represents the proportion of variance in the dependent variable explained by the independent variables in the regression model. It is calculated using the following formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Here, n represents the number of data points, y_i is the actual value for the i-th data point, $\hat{y}_i$ is the predicted value for the i-th data point, and $\bar{y}$ is the mean of the actual values. $\mathbf{R^2}$ ranges from 0 to 1, with higher values indicating a better fit.

Mean Absolute Error (MAE): This metric measures the average absolute difference between the predicted and actual values. It is calculated using the following formula:

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$$

Here, the absolute value function ensures that all differences are considered positive. Lower MAE values indicate better model performance.

Table 2: Datasets Variants

Dataset	**Features**
Classic	original features
Stations basic	original + unweighted stations counts
Stations weights inverse	original + Reverse distance weighted score of stations counts
Stations weights Gaussian	original + Gaussian distance weighted score of stations counts
Stations weights logarithmic	original + logarithmic distance weighted score of stations counts
Stations weights linear	original + linear distance weighted score of Stations counts
Stations weighted all	original + all distance weighted scores

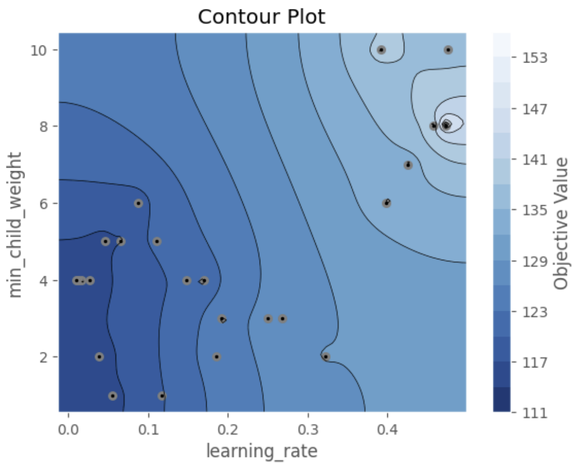

Figure 4: Learning rate vs min child weight validation MAE contour

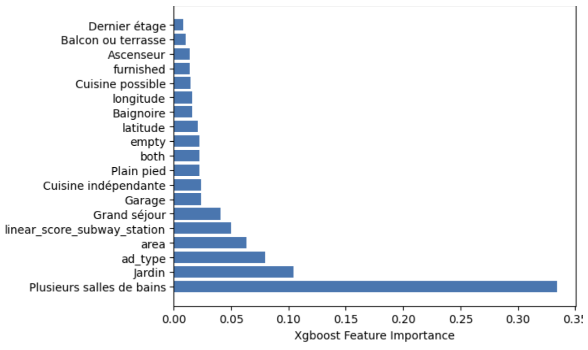

Figure 5: Hyperparamater Importance

Mean Squared Error (MSE): This metric measures the average squared difference between the predicted and actual values. It is calculated using the following formula:

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

MSE penalizes larger errors more heavily compared to MAE. Lower MSE values indicate better model performance.

By evaluating these metrics for each model after fine-tuning, we can gain valuable insights into their ability to fit the data and make accurate predictions.

5.6 Results

Fine-tuning the models with Bayesian search yielded significant improvements. As shown in Table 3, XGBoost demonstrably outperformed every other model on every dataset variation we evaluated. This dominance across the board highlights the effectiveness of XGBoost for regression tasks, particularly after careful hyperparameter tuning.

Strength of Tree Ensembling Methods: While XGBoost reigned supreme, it's noteworthy that Random Forest consistently secured second place across all datasets. This strong performance by both models reinforces the effectiveness of tree-based ensemble methods in capturing complex relationships within the data for regression problems.

Impact of Feature Engineering: The table also reveals an interesting trend regarding the "Stations weights" datasets. Here, variations that incorporated weighted distance scores (inverse, linear) as features (Stations weights inverse, Stations weights linear) resulted in superior performance for both XGBoost and Random Forest compared to the baseline "Classic" dataset and the "Stations basic" variant. This suggests that the inclusion of these weighted distance scores as features played a crucial role in enhancing model performance. This finding underscores the importance of thoughtful feature engineering in improving the effectiveness of machine learning models.

As discussed earlier, the success of machine learning models hinges not only on the chosen algorithms but also on the quality of the features used. Feature engineering, the process of creating and selecting informative features from raw data, plays a critical role in improving model performance. Figure 5 shows the feature importance for the best model overall. the impact of the "linear$_{s}core_{s}ubway_{s}tation$" $featureexemplifythisprinciple.Thisa$

Table 3: Test results for every dataset variation, showing top two models

Dataset	Model	R^2	MSE	MAE
Classic	Xgboost	0.78391	31158	102.29
	RandomForest	0.77028	33123	109.06
Stations basic	XGboost	0.78135	31526	102.80
	RandomForest	0.76862	33361	108.91
Stations weights inverse	XGboost	**0.79097**	**30141**	101.48
	RandomForest	0.76624	33706	111.12
Stations weights Gaussian	XGboost	0.78901	30423	101.93
	RandomForest	0.76829	33411	111.49
Stations weights logarithmi	XGboost	0.78093	31588	102.28
	RandomForest	0.76255	34239	112.31
Stations weights linear	Xgboost	0.78889	30439	**101.32**
	RandomForest	0.762149	34295	111.92
Stations weighted all	Xgboost	0.78851	30496	102.6
	RandomForest	0.76264	34226	112.66

6 Conclusion

By leveraging the different machine learning models on our dataset, our experiments illustrate the supremacy of the XGBRegressor in predicting house prices. The comparative analysis reveals the XGBRegressor as the standout performer, exhibiting significantly improved Mean Squared Error results as well as Mean absolute error compared to other models. Remarkably, this model demonstrates exceptional accuracy in house price prediction, boasting the highest coefficient of determination R-squared to 0.78. Moreover, the incorporation of Points of Interest (POIs) into our model enhances prediction accuracy by capturing nuanced spatial features of the housing market. This integration allowed for more refined predictions, emphasizing the critical role of spatial features in housing price prediction.

Future research might consider enhancing the size of the dataset to train predictive models more effectively. We can particularly incorporate a larger dataset that includes not only rental but also selling prices. Additionally, exploring the utilization of continuous data streams, such as those facilitated by Apache Kafka technology, offers the potential for real-time insights into market trends and fluctuations, thus improving the responsiveness and accuracy of predictive models. The findings of this study carry significant implications for the prediction of future house renting prices and the formulation of real estate policies.

References

[1] T. Akiba, S. Sano, T. Yanase, Y. Oono, and N. Ohta. Optuna: A distributed hyperparameter optimization framework, 2019.

[2] Fatemeh Aghaei Christopher James Pettit Ali Soltani, Mohammad Heydari. Housing price prediction incorporating spatio-temporal dependency into machine learning algorithms. *Cities*, 131:15, 2022.

[3] Susana Sargento Filipe Cabral Pinto Ana Almeida, Susana Brás. Time series big data: a survey on data stream frameworks, analysis and algorithms. *Journal of Big Data*, 10(83):32, 2023.

[4] Raghu K. Ganti Mudhakar Srivatsa Archith J. Bency, Swati Rallapalli and B. S. Manjunath. Beyond spatial auto-regressive models: Predicting housing prices with satellite imagery. *International Journal of Advanced Computer Science and Applications*, page 10, 2017.

[5] Jae Kwon Bae Byeonghwa Park. Using machine learning algorithms for housing price prediction: The case of fairfax county, virginia housing data. *Expert Systems with Applications*, 42(6):2928–2934, 2015.

[6] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. 2016.

[7] Pouya Soltani Zarrin Manuel Díaz Bartolomé Rubio Cristian Martín, Peter Langendoerfer. Kafka-ml: Connecting the data stream with ml/ai frameworks. *Future Generation Computer Systems*, 126:15–33, 2022.

[8] NumPy Developers. Numpy, 2010.

[9] Pedro González María José Del Jesus Fernando Puentes, María Dolores Pérez-Godoy. An analysis of technological frameworks for data streams. *Progress in Artificial Intelligence*, 9:239—-261, 2020.

[10] J. D. Hunter. Matplotlib: A comprehensive system for creating visualizations in python, 2007.

[11] Indrė Žliobaitė Jesse Read. Learning from data streams: An overview and update. page 48, 2022.

[12] Laskhya Arsh Nishant, Mukul. Real estate price prediction using machine learning. *International Journal for Research in Applied Science and Engineering Technology*, 11(5):7, 2023.

[13] T. Okabe and A. Murray. cupy: A gpu interface for numpy, 20.

[14] pandas development team. pandas: Powerful python data analysis toolkit, 2010.

[15] A. Paszke, S. Zhang, and S. et al. Chintala. Pytorch: An open-source deep learning platform, 2019.

[16] F. Pedregosa, G. Varoquaux, and A. et al. Gramfort. Scikit-learn: Machine learning in python, 2011.

[17] Inc. Plotly. Plotly: Open-source graph-sharing for r and python, 2015.

[18] Hy Dang Bo Mei Quang Truong, Minh Nguyen. Housing price prediction via improved machine learning techniques. *Procedia Computer Science*, 174:433–442, 2020.

[19] Nor Hamizah Zulkifley Ismail Ibrahim Shuzlina Abdul-Rahman1, Sofianita Mutalib. Advanced machine learning algorithms for house price prediction: Case study in kuala lumpur. *International Journal of Advanced Computer Science and Applications*, 12(12):736–745, 2021.

[20] Mahdieh Yazdani. Machine learning, deep learning, and hedonic methods for real estate price prediction. page 33, 2021.

[21] Alexander Ihler Baoxiang Pan Yixuan Ma, Zhenjiang Zhang. Estimating warehouse rental price using machine learning techniques. *International Journal of Computers, Communications Control*, 13(2):235–250, 2018.