Predictive Real Estate for Florida

Empowering Insights Through Machine Learning Algorithms

Goutham Babu Mosya (G01387090), Ganga Raju Parla (G01394334), Venkata Muthyala Gowri Malla (G01387073)- GROUP - 5

May, 2024

Abstract

This study leverages advanced machine learning techniques to predict house prices in the USA, utilizing a comprehensive dataset that includes crucial geospatial data. After extensive data preprocessing, four predictive models—ridge regression, decision trees, random forest, and XGBoost—were evaluated for their effectiveness. The study emphasizes the integration of real-time geospatial data to enhance model accuracy. Results indicate that XGBoost, augmented by gradient descent and regularization, significantly outperforms other models, demonstrating the substantial impact of geospatial data on predictive accuracy in real estate markets. Specifically, the incorporation of geospatial data yielded noticeable improvements in model performance, underscoring its importance in enhancing predictive capabilities.

1. Introduction

In the dynamic real estate market of Florida, accurately predicting house prices is crucial for a multitude of stakeholders, including investors, developers, and policy makers. To tackle this challenge, our project began with a thorough investigation to source comprehensive datasets that include essential housing price insights, along with critical geospatial data that reflects the varied demographics and economic conditions across the state. This foundational step was vital to ensure that the data not only represents the broad spectrum of the USA housing market but also includes the spatial variables that influence property values.

Recognizing the complexity of the data and its potential inconsistencies, extensive preprocessing was undertaken to refine the datasets, making them suitable for advanced analytical techniques. This preprocessing involved cleaning missing values, encoding categorical variables, and normalizing data scales to enhance model performance and reliability.

With a clean and robust dataset in hand, we proceeded to evaluate and compare the efficacy of four different machine learning models: ridge regression, decision trees, random forest regression, and XGBoost. Each model was selected for its unique strengths and potential to handle the specific characteristics of real estate data. The models were rigorously tested through a series of experiments designed to assess their predictive accuracy and computational efficiency.

Furthermore, a key aspect of our project was to explore the impact of incorporating real-time geospatial data into these models. This approach aimed to determine how such data, which includes location-specific attributes like proximity to essential services and infrastructure, could enhance the predictive models' sensitivity to local market conditions. By integrating this geospatial data, we sought to uncover deeper insights into how environmental and regional factors contribute to housing prices in Florida.

Our analytical journey not only focused on identifying the best-performing model but also aimed to highlight the importance of real-time geospatial data in refining predictive outcomes. This comprehensive analysis is intended to provide a blueprint for leveraging machine learning in real estate market predictions, offering a sophisticated toolset for making informed decisions in one of the most fluctuating markets in the U.S.

2. Merits

Everybody contributed equally where Goutham focusing on XGBoost algorithm, Ganga focusing on Datasets search and Ridge Regression, and Gowri focusing on Decision Tree and Random Forest Regression.

3. DataSet

3.1 Data Source and Selection Criteria

The dataset used in this analysis was sourced from [6] . It provides comprehensive information about real estate properties across the USA. From this dataset, we specifically extracted and analyzed data

related to properties located in Florida. The decision to use this particular dataset was based on several factors, including the relevance of the data to the research question and the availability of key features that are crucial for predicting property prices in Florida.

The columns in the dataset that influenced our decision to choose this data include:

- **Brokered By:** This column indicates the brokerage firm or agent responsible for listing the property. We included this information to assess the influence of different brokers on property prices and market trends.
- **Status:** Denotes the current status of the property, such as "For Sale," "Sold," or "Under Contract." Understanding the status of properties allows us to analyze market dynamics and trends over time.
- **Price:** Provides the monetary value of the property. This is our target variable, as we aim to predict property prices based on other features.
- **Bedrooms (Bed) and Bathrooms (Bath):** Represent the number of bedrooms and bathrooms in the property, respectively. These features are crucial determinants of property value and are commonly used in real estate pricing models.
- **Lot Size (Acre Lot):** The acreage of the property's lot is an important factor influencing its price. Larger lots typically command higher prices, so we included this feature to capture variations in property size.
- **Location (Street, City, State, Zip Code):** The geographic location of the property plays a significant role in its value. We included street, city, state, and zip code information to analyze regional price variations and assess the impact of location on property prices.
- **House Size:** The size of the house, measured in square feet, is another crucial determinant of property value. Larger houses usually have higher prices, so we included this feature to account for variations in property size.
- **Previous Sold Date (Prev Sold Date):** Provides information about the date when the property was previously sold. Analyzing previous sale dates allows us to identify trends in property turnover and assess market activity.

By selecting these columns, we aimed to build a predictive model that accurately estimates property prices based on key features and attributes.

3.2 Geo-Spatial Data

In addition to the main real estate dataset, we also incorporated geo-spatial data to enhance our analysis. The geo-spatial data includes information about various points of interest and infrastructure within Florida. This geo-spatial data covers a diverse array of significant locations and infrastructure across Florida including:

- **Fire Stations**: Locations of fire stations across Florida.
- **Healthcare Centers**: Geospatial information about healthcare centers within the state.
- **Hospitals**: Data on the locations of hospitals throughout Florida.
- **Parks**: Information regarding the whereabouts of parks within the state.
- **Police Stations**: Locations of police stations across various regions in Florida.
- **Schools**: Data about the geographical distribution of schools in Florida.
- **Transportation Facilities**: Details on transportation infrastructure, including airports, bus stations, and train stations.

The incorporation of geo-spatial data is essential for enriching the analysis and improving the predictive performance of the model. By considering the proximity of properties to these points of interest and infrastructure, the model gains valuable insights into the desirability and convenience of a property's location, which are significant factors influencing property prices.
The geo-spatial data was obtained from reputable source [2] . These source provide reliable information on various points of interest and infrastructure across Florida, ensuring the accuracy and relevance of the data used in the analysis.

4. PreProcessing

4.1 Why PreProcessing?

PreProcessing is a crucial step in the data analysis pipeline. It involves cleaning and transforming raw data into a format that is suitable for analysis and modeling. Preprocessing helps to address issues such as missing values, outliers, and inconsistencies in the data, which can affect the performance of machine learning models.

Steps Involved:

4.2 Data Filtering

Filtering the Dataset for Properties in Florida: The dataset is filtered to only include records related to properties located in Florida. This filtering is based on the 'state' column, where only records with the value 'Florida' are retained.

4.3 Convert Categorical Columns to Numeric Codes

Columns Converted:

- status: Represents the status of the property.
- city: Represents the city where the property is located.
- state: Represents the state where the property is located.

Why It's Necessary: Machine learning algorithms typically require input features to be numeric. Categorical variables like status, city, and state need to be converted into numerical format for the algorithm to process them effectively. Label encoding is one way to achieve this, where each unique category is assigned a unique integer.
Dropped Column:

- prev_sold_date: This column is dropped as it may not be relevant for predicting property prices in the context of this project.

4.4 Remove Rows with NaN or Infinite Values in 'price'

Reasoning: The presence of missing or infinite values in the target variable (price) can adversely affect the training of machine learning models. Since the dataset is large enough, removing rows with missing or infinite values in the price column is a reasonable approach as it helps to maintain data integrity without significantly impacting the dataset size.

4.5 Filtering Out Properties with Extreme Prices

Properties with prices below $500 and above $95,000,000 are filtered out.
Reasoning for Lower Bound: Properties with prices below $500 are likely to be outliers or erroneous data points, as such low prices are uncommon in the real estate market.
Reasoning for Upper Bound: Properties with prices above $95,000,000 are considered extreme outliers and may skew the analysis. Excluding such properties helps to focus the analysis on the more typical range of property prices.

4.6 Log-Transforming the Target Variable

Why Log-Transformation: Log-transforming the target variable (price) is a common technique used in regression tasks, especially when the target variable is skewed or does not follow a normal distribution. Log-transformation helps to stabilize variance and make the distribution of the target variable more symmetric, which can improve the performance of regression models. In this case, log-transforming the price variable and scaling it by a factor helps to normalize the distribution of property prices, making it more suitable for modeling.

4.7 GeoSpatial data PreProcessing

Incorporated additional information from various geospatial datasets by appending an extra column to each dataset, specifying the type of geospatial dataset it represents. This additional column enabled us to differentiate between different types of geospatial data. Identified the unique type of geospatial data represented by the file and grouped the data by 'zipcode', counting the occurrences of each unique zipcode. These counts were then integrated into a DataFrame, where each column represented a different type of geospatial dataset, and each row represented a zipcode with its corresponding count of occurrences across the datasets. Any missing values in the DataFrame were filled with zeros to ensure completeness of the dataset. This comprehensive preprocessing step augmented our dataset with valuable geospatial information, enhancing its richness and utility for subsequent analysis and modeling tasks.

These preprocessing steps ensure that the data is in a suitable format for analysis and modeling, improving the accuracy and reliability of the machine learning models we built on it.

5. Experiment Models

Explain what models we choose and why we choose these and what we were trying to achive with these models.

5.1 Linear Regression with L2 Regularization

Linear regression is a fundamental technique used in machine learning for modeling the relationship between a dependent variable and one or more independent variables. The basic premise of linear regression is to fit a linear equation to the observed data by minimizing the sum of squared differences between the observed and predicted values. Mathematically, this can be represented as:

$$\min_{\beta_0,\beta_1,...,\beta_p} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2$$

Here, y_i represents the observed value of the dependent variable, x_{ij} represents the jth feature of the ith observation, and $\beta_0, \beta_1, ..., \beta_p$ are the coefficients of the linear equation.

However, basic linear regression is susceptible to overfitting when dealing with high-dimensional data or when the number of features is close to or exceeds the number of observations. L2 regularization, also known as Ridge regression [4], addresses this issue by adding a penalty term to the loss function, which penalizes large values of the coefficients. The modified objective function for Ridge regression is given by:

$$\min_{\beta_0,\beta_1,...,\beta_p} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2 + \alpha \sum_{j=1}^{p} \beta_j^2$$

Here, α is a hyperparameter that controls the strength of the regularization. A higher value of α results in stronger regularization, leading to smaller coefficient values and a simpler model.

L2 regularization helps in preventing overfitting by shrinking the coefficients towards zero, effectively reducing the model's complexity and variance. This regularization technique is particularly useful when dealing with multicollinearity, where two or more independent variables are highly correlated. By penalizing large coefficient values, L2 regularization encourages the model to select all relevant features and assign them appropriate weights, leading to improved generalization performance.

5.1.1 Parameter Choosing

The choice of the regularization parameter α is crucial in Ridge regression, as it determines the trade-off between fitting the training data and keeping the model simple. The optimal value of α can be selected through techniques such as cross-validation, where the dataset is divided into training and validation sets. Various values of α are tested on the training set, and the one that results in the best performance on the validation set is chosen. Mathematically, the objective function used in cross-validation can be represented as:

$$\min_{\alpha} \sum_{k=1}^{K} \text{Loss}(k)$$

Here, K represents the number of folds in cross-validation, and $\text{Loss}(k)$ represents the loss function (e.g., mean squared error) for the kth fold.

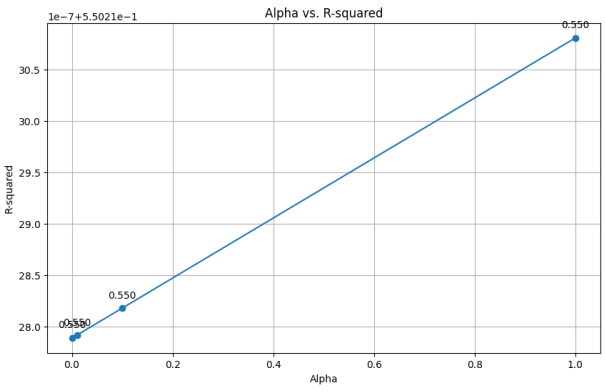
5.1.2 Important Info/Graphs

In this section, we present important information and graphs related to the linear regression with L2 regularization analysis:

In the context of linear regression with L2 regularization, it's essential to choose appropriate hyperparameters such as the regularization parameter (α) and the number of folds in cross-validation. Visualizing the relationship between these hyperparameters and the model performance metrics, such as R^2 (coefficient of determination), can aid in selecting the optimal values.

The graph is instrumental in fine-tuning the hyperparameter Alpha of the L2 regularized regression model, ensuring that it achieves the best balance between bias and variance and generalizes well to unseen data. By visually inspecting these relationships, we can make informed decisions regarding hyperparameter selection, leading to more reliable and robust predictive models.

1. **Alpha vs.** R^2: Plotting alpha against the coefficient of determination (R^2) helps in selecting the optimal regularization strength. This plot typically shows how the model's performance varies with different values of α. A U-shaped curve is often observed, indicating that there is an optimal value of α that maximizes model performance. The goal is to choose the α that maximizes R^2 without overfitting or underfitting the data.



Based on the analysis of the graph depicting the relationship between alpha and R-squared values, it was observed that the R-squared metric exhibited minimal variation across different alpha values. Consequently, for the Ridge regression model, an alpha value of 1.0 was selected as it yielded comparable R-squared values while maintaining model simplicity and interpretability.

2. **Feature Importance Plot**: A horizontal bar plot is used to visualize the importance of each feature in the Ridge regression model. The importance of each feature is determined by the absolute value of its coefficient in the regression equation. Features with higher importance contribute more to predicting the target variable, while features with lower importance have less influence on the predictions. The most important features are positioned at the top of the plot, while less important features are located towards the bottom.

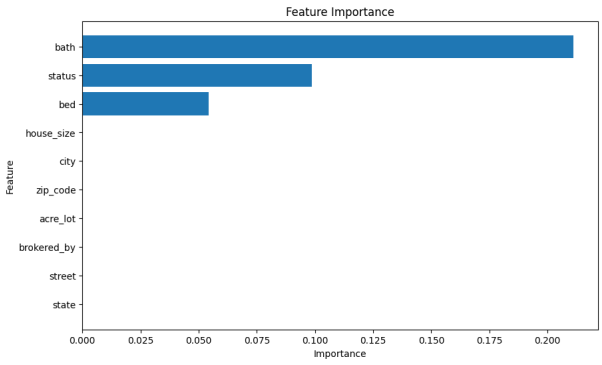


Figure 1: Feature Importance Plot WITHOUT Geo Spatial data

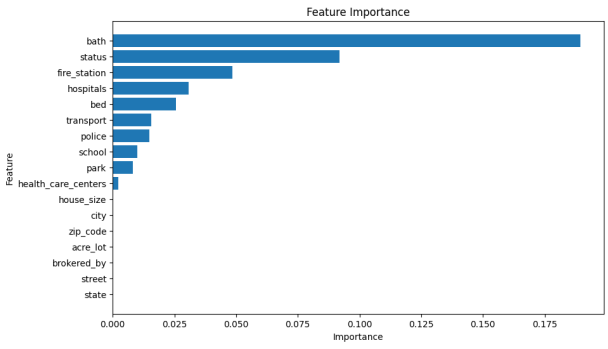


Figure 2: Feature Importance Plot WITH Geo Spatial data

The dataset used for linear regression with L2 regularization typically consists of features (independent variables) and a target variable (price). Visualizing the relationship between the features and the target variable can provide insights into the data's underlying patterns and help in feature selection and model interpretation.

5.2 Decision Tree

A decision tree regressor is a machine learning model used for regression tasks. It makes predictions by partitioning the data into subsets based on certain conditions, then makes a prediction based on the average or mean target value in the final subset (or leaf node). In a decision tree model for regression tasks, such as predicting house prices, splitting the data into subsets is often based on measures of impurity or variance reduction. The goal is to minimize the variance within the groups resulting from the split. The chosen split is the one that maximizes the reduction in variance.

Variance is a measure of the spread of the data around the mean:

$$\text{Variance}(D) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

Where,
- D is the dataset (e.g., all samples in a node).
- y_i is the target value of the i-th sample.
- $\bar{y}$ is the mean of the target values in the dataset.
- n is the number of samples.

The decision tree algorithm evaluates potential splits by calculating the reduction in variance. The formula for variance reduction (ΔVariance) is:

$$\Delta\text{Variance} = \text{Variance}(D) - \left(\frac{n_{\text{left}}}{n}\text{Variance}(D_{\text{left}}) + \frac{n_{\text{right}}}{n}\text{Variance}(D_{\text{right}})\right)$$

Where,
- D is the original dataset at the node.
- D_{left} and D_{right} are the datasets resulting from the split (left and right child nodes).
- n_{left} and n_{right} are the number of samples in the left and right child nodes, respectively.
- n is the total number of samples in the original dataset at the node.

Building the Decision Tree

- Starting at the Root Node:

  The algorithm starts at the root node with all the samples.

- Evaluating Potential Splits:

  For each feature and each possible value (threshold) to split the data, calculate the reduction in variance.

  Choose the split that maximizes the variance reduction.

- Recursive Splitting:

  After choosing the best split, the data is divided into two subsets (left and right).

  The algorithm recursively applies the same process to the child nodes.

- Stopping Criteria:

The algorithm stops splitting when a stopping criterion is met, such as reaching a maximum depth, a minimum number of samples at a leaf, or minimal variance reduction.

Implemented this Decision tree model for predicting the house prices [8] and enhanced model in various ways.

5.2.1 Parameter choosing

'max_depth' This parameter specifies the maximum depth of the decision tree. A higher value allows the tree to grow deeper and potentially capture more intricate patterns in the data. Setting max_depth too high can lead to overfitting, where the model performs well on training data but poorly on new data. Lower values of max_depth may result in underfitting, where the model is too simplistic to capture the complexity of the data.

'min_samples_split' specifies the minimum number of samples required to split an internal node in the decision tree. A higher value of 'min_samples_split' reduces the likelihood of creating smaller branches, which can help prevent overfitting. Lower values of 'min_samples_split' can lead to more granular splits, increasing model complexity.

'min_samples_leaf' specifies the minimum number of samples required to be at a leaf node in the decision tree. Setting a higher value of 'min_samples_leaf' can help smooth the model and prevent overfitting. Lower values of 'min_samples_leaf' allow the tree to create smaller leaf nodes, which can potentially increase model complexity.

Pruning:

Cost complexity pruning is a technique used to control the complexity of the decision tree and avoid overfitting. The pruning process uses a complexity measure based on a cost function and the number of leaves in the tree.

The formula for cost complexity pruning (CCP) is:

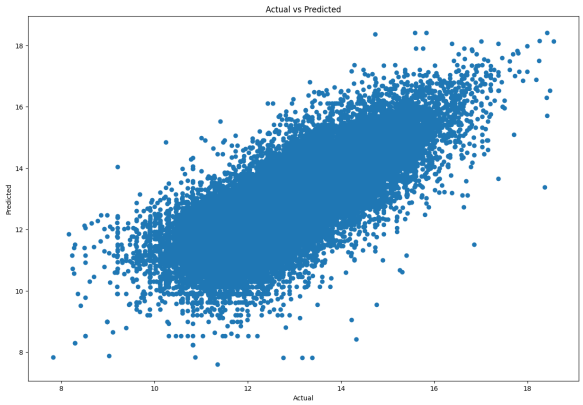
$$\text{Cost Complexity} = \text{SSE}(T) + \alpha \cdot |T|$$

Where: - $\text{SSE}(T)$ is the sum of squared errors in the tree T. - $|T|$ is the number of leaves in the tree. - α is a regularization parameter that controls the tradeoff between prediction error and model complexity.

- SSE (Sum of Squared Errors): This term measures the total error in the tree. It is calculated by summing the squared differences between the predicted and actual target values in each node. - Number of Leaves: The number of leaves $|T|$ in the tree represents the complexity of the model. More leaves mean a more complex model. - Regularization Parameter (α): This parameter controls the tradeoff between prediction error and model complexity. By adjusting α, you can control the extent of pruning. Higher values of α lead to more aggressive pruning, resulting in a simpler model with fewer leaves and possibly higher bias. Lower values of α allow for more complex models with more leaves, which may lead to better fit but potentially higher variance. Adjusting the alpha value ('ccp_alpha') in cost complexity pruning controls the level of pruning.

5.2.2 Statistics

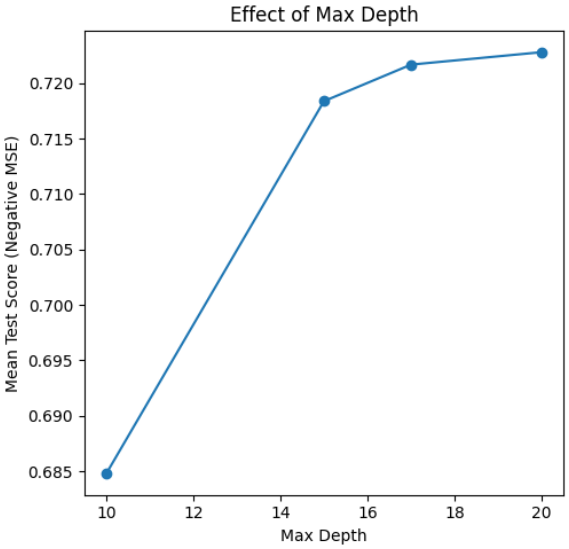
At first after all the feature selection and feature engineering and with basic model and parameters, the predictions generated are scattered compared to actuals as follows,

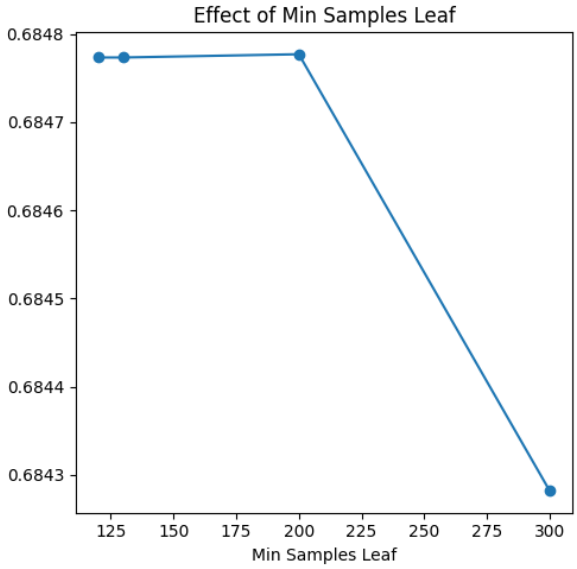


This picture clearly depicts how a lot of plots scattered widely from densed area shows the deviation of predicted prices from actual prices.

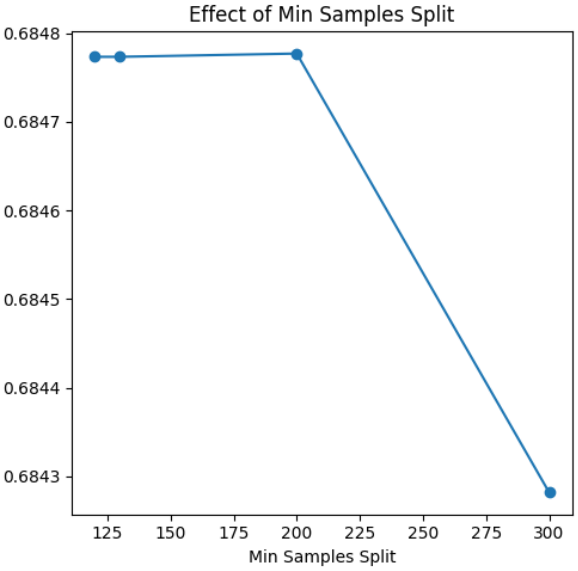
Defined a parameter grid with different values of max_depth, min_samples_split, and min_samples_leaf with 5-fold cross-validation grid search to find the optimal hyperparameters on R^2 evaluation metric.

A higher max depth allows the tree to model more complex relationships in the data, but it also increases the risk of overfitting. Ideally, we take a max depth that balances complexity with good model performance on the test set. Below plot shows the effect of max_depth on the mean test score:





Higher value of min_samples_split can prevent the model from creating very small and potentially noisy splits in the data, reducing the risk of overfitting. When it comes to min_samples_split, mean test score changes as follows:

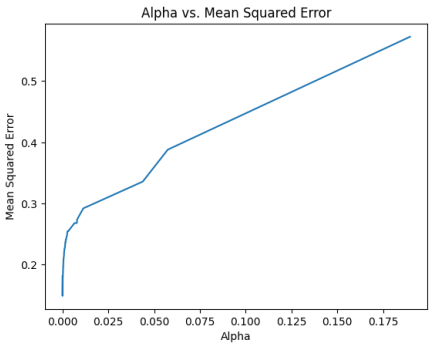


Finally, Larger min_samples_leaf values create more general and smooth splits in the data, which can improve the model's ability to generalize and reduce overfitting. we can visualize the effect of min_samples_leaf on mean test score in our model over various parameters as below:

parameter grid search implementation selects parameter values that lead to high test scores while avoiding overfitting. The selected best parameters and best corresponding r2 score are

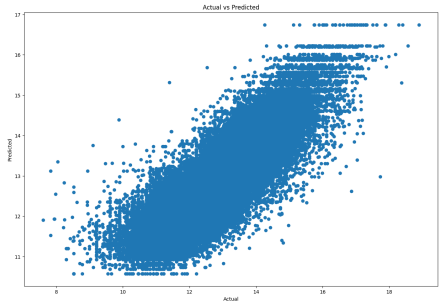
Best parameters: 'max_depth': 20, 'min_samples_leaf': 120, 'min_samples_split': 130
Best score (r2): 0.72

Obtained the cost complexity pruning alpha values and impurities associated with each alpha value. For each alpha value, created a decision tree and calculated MSE values. plotted the alpha values against the corresponding mean squared errors,



Identified the optimal alpha value (1.3944797735713967e-05) that results in the minimum mean squared error and then created a new decision tree with the optimal ccp_alpha and the best pruning parameters found during previous searches.

With these enhancements, the same scatter plot became as below,



We can observe that it is more dense than the begining

5.3 Random Forest Regression

Random Forest is an ensemble learning method for regression tasks that builds a forest of decision trees and averages their predictions to improve model performance. A Random Forest regressor consists of multiple decision trees, each trained on a random subset of the training data and a random subset of features. The final prediction is the average of the predictions from all trees. This approach helps reduce overfitting and improve generalization. The final prediction $\hat{y}$ for an input instance x from a Random Forest regressor can be calculated as:

$$\hat{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

Where,
- n is the number of decision trees in the forest.
- y_i is the prediction of the i^{th} decision tree in the forest.

5.3.1 Parameter choosing

'n_estimators' defines Number of decision trees in the forest.
'max_depth' represents Maximum depth of each decision tree.
'min_samples_split' represents Minimum number of samples required to split an internal node.
'min_samples_leaf' represents Minimum number of samples required at a leaf node.

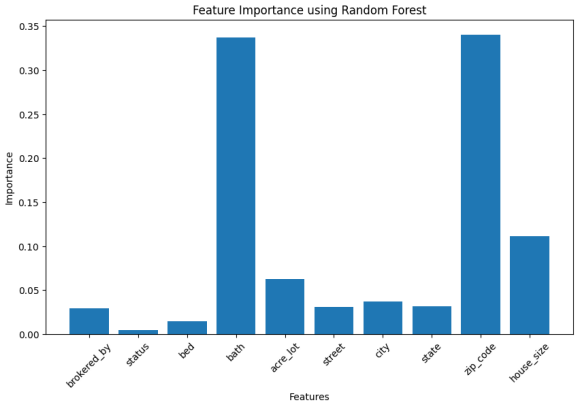
Cost Complexity Pruning:
Cost complexity pruning can be applied to individual decision trees in the Random Forest to control their complexity and avoid overfitting.

Pruning individual trees can enhance the overall Random Forest model by reducing complexity.

Implemented this Random Forest Regression model [7] for predicting the house prices and enhanced model in various ways.

5.3.2 Statistics

Calculated the importance of each feature using the Random Forest model. Below visualization helps understand which features are most important in the model's predictions.



For each tree, calculated the cost complexity pruning path. For each alpha value, each tree is pruned and then evaluated using cross-validation (5-fold) with the scoring metric of negative mean squared error. The code identifies the optimal alpha value that results in the minimum mean squared error. The optimal alpha value is then used to prune each tree in the Random Forest model and retrain it on the training data. The code uses the pruned Random Forest model to predict the prices.

Defined a dictionary with parameter names as keys and lists of possible values as values. Fit the grid search to the training data. Retrieved the best parameters and the best model from the grid search results.

n_estimators=100, max_depth=12, min_samples_split=2, max_features= log2

Used the best model to predict the target variable. calculated various metrics to evaluate the model's performances.

- r2 Score: 0.8099433888653528
- Mean Squared Error: 0.10881620742584765
- Root Mean Squared Error: 0.34987301712302514
- Mean Absolute Error: 0.29509890018086752

5.4 eXtreme Gradient BOOSTing Model

XGBoost, or eXtreme Gradient Boosting, is a powerful machine learning technique that has gained popularity due to its efficiency and effectiveness across a variety of regression and classification tasks [1]. The core technique behind XGBoost, involves constructing new models that predict the residuals or errors of prior models and then combining these models through a weighted sum to make the final prediction. This technique is highly effective in reducing bias and variance, leading to more accurate models. [5] Where The formula we used for updating the model in gradient boosting is:

$$F_t(x) = F_{t-1}(x) + \eta \cdot h_t(x)$$

XGBoost builds upon basic gradient boosting by utilizing a more regularized model formalization to control over-fitting, which is vital for predictive accuracy [1]. The regularized objective in XGBoost is given by:

$$\text{Obj} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k)$$

Since we were working with house prices as our target variable we changed the default behaviour of the model to make it work more better, observed how gradually the loss function value is reduced over the number of boosting rounds, examined the feature importance given by the model to make sure how good the model is in predicting the nuances of the features. To ensure the robustness of our model, we implemented a five-fold cross-validation strategy using the training data (X_train and Y_train). This approach allowed us to fine-tune the model parameters while avoiding overfitting, ensuring that the model's performance was not only optimized on the training data but also capable of performing consistently on unseen data. Finally, we used this optimally tuned model to make predictions on the test dataset (X_test), evaluating its predictive accuracy against the actual values in Y_test. This rigorous methodology not only validated the model's effectiveness but also reinforced the importance of our parameter choices in achieving high prediction accuracy.

5.4.1 Parameter Selection

The 'objective' parameter was set to 'reg squaredlogerror' to minimize the squared logarithm of the prediction error, which is particularly suitable for regression problems with a log-transformed target variable like house prices. This objective helps in reducing the impact of outliers and scaling errors proportionately.[3]. With this parameter the default loss function Mean Squared Error is replaced with the Mean Squared Logarithmic Error eq. (1), and the extreme gradient boosting algorithm works on minimizing this value using the gradient in eq. eq. (2) and hessian in eq. eq. (3) formulas of the loss function.

$$\text{MSLE}(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} (\log_e(1 + y_i) - \log_e(1 + \hat{y}_i))^2 \tag{1}$$

$$\frac{\partial}{\partial \hat{y}_j} \text{MSLE}(y, \hat{y}) = \frac{2}{N} \sum_{i=0}^{N-1} (b_i - a_i) \left(\frac{1}{1 + \hat{y}_i} \right) \tag{2}$$

$$\frac{\partial^2}{\partial \hat{y}_i^2} \text{MSLE}(y, \hat{y}) = -\frac{2}{N} \sum_{i=0}^{N-1} \frac{b_i - a_i + 1}{(1 + \hat{y}_i)^2} \tag{3}$$

$$\text{where, } b_i = \log(1 + \hat{y}_i), \tag{4}$$

$$and, a_i = \log(1 + y_i) \tag{5}$$

The 'colsample_bytree' parameter was configured to 0.2, lower than the default, to ensure that each tree uses only 20% of the features when building branches. This reduction helps prevent overfitting and improves model generalization, as it forces the model to not rely heavily on any single or a small group of features, thereby enhancing the robustness of the predictions.

A 'learning_rate' of 1 was chosen, which is aggressive but can be effective in converging quickly to a good solution when combined with other regularization techniques and careful tuning of the tree complexity. It speeds up the learning process, although it requires careful monitoring to avoid overshooting the minimum error.

The 'max_depth' was set extraordinarily high at 100 to allow the model to learn highly specific patterns in the data, capturing complex relationships. While a deep tree risks overfitting, in controlled experiments with sufficient regularization and cross-validation, it can yield insightful results, especially in varied and nuanced datasets like real estate listings.

Lastly, the 'alpha' parameter, which adds L1 regularization on the weights, was set to 0 which is the default, relying solely on the tree structure and other forms of regularization to control overfitting. This choice was made under the assumption that the model's complexity and feature selection are managed adequately by the depth and column sampling settings.

5.4.2 RMSLE Over Boosting Rounds

The Figures fig. 3 and fig. 4 illustrates the convergence of the Root Mean Square Logarithmic Error (RMSLE) for both the training and testing datasets over the number of boosting rounds for regular Dataset and Geospatial dataset respectively. Initially, the RMSLE sharply decreases, indicating that the model rapidly improves its ability to predict the log-transformed prices with each additional boosting round. This rapid improvement in the initial rounds signifies that the model efficiently captures the dominant patterns in the dataset.

The training RMSLE stabilizes quickly and remains flat for most of the boosting process, which suggests that the model fits well to the training data without further significant gains from additional rounds. The test RMSLE, similarly low and stable throughout the process, indicates good generalization of the model to unseen data. This pattern demonstrates the robustness of the XGBoost algorithm in handling overfitting, especially when regularization techniques are applied effectively within the model.

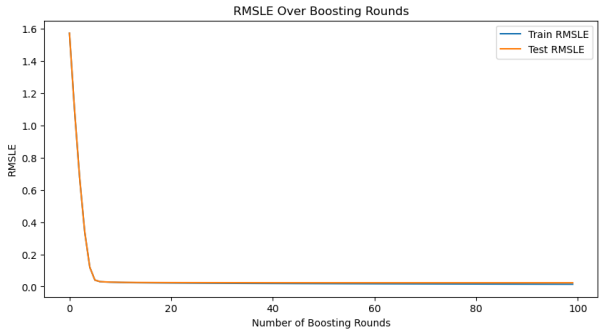


Figure 3: RMSLE over Boosting Rounds for the Regular Dataset

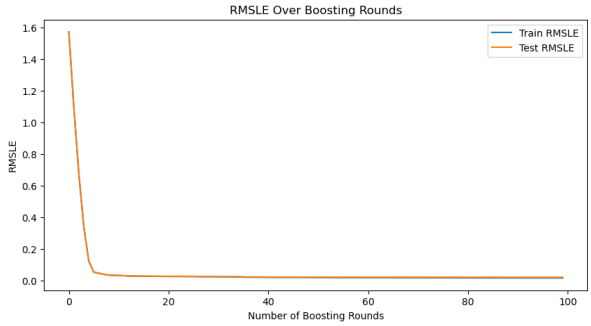


Figure 4: RMSLE over Boosting Rounds for the Geospatial Dataset

5.4.3 Feature Importance Analysis

The Figures fig. 5 and fig. 6 showing feature importance provides insights into which variables have the most impact on the predictions of house prices. In this graph, the features are ranked based on their F scores, a metric that counts the number of times a feature is used to split the data across all trees.

The brokered_by feature leads in importance, followed closely by street and house_size. These features likely contain critical information about the property's location and size, which are crucial determinants of house prices. The significant role of city and zip_code underscores the geographical factors that typically influence real estate values due to differing neighborhood qualities, accessibility, and amenities.

The lesser importance of features like status, bed, and bath suggests that while these factors are relevant, they do not contribute as crucially to price variability compared to location and property size. This insight can help real estate analysts and investors focus on key property attributes when evaluating investment opportunities or developing pricing models.

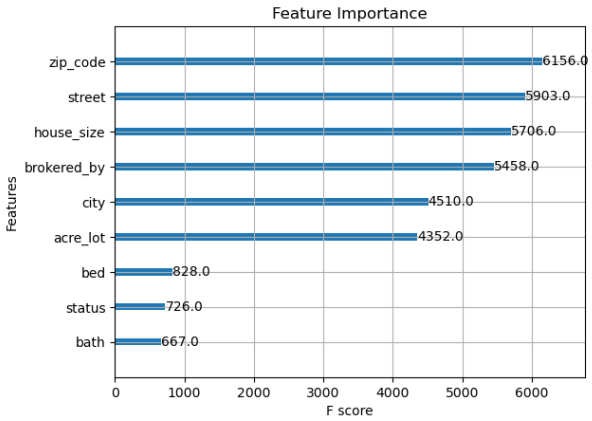


Figure 5: Feature Importance with Regular Dataset

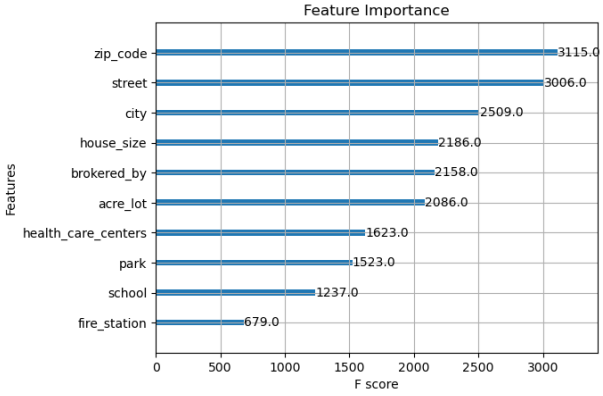


Figure 6: Feature Importance with Geospatial Dataset

The RMSLE curves and Feature Importance

analysis we can clearly see the model is able to find the nuances in the data and select the best feature to predict the prices and it take very less rounds of boosting to reach the goal which is expected, although more rounds gives better results if the dataset is very large this feature can be handy.

6. Results

6.1 Evaluation Metrics

We evaluated the performance of our regression models using two primary metrics: Root Mean Squared Logarithmic Error (RMSLE) and R-Squared (R^2). The Mean Squared Logarithmic Error (MSLE) is a statistical metric utilized to assess the precision of a prediction model, especially useful when dealing with data that spans a broad spectrum of values. It calculates the mean of the squared discrepancies between the logarithms of the forecasted and observed values [3]. R-Squared (R^2) represents the proportion of variance explained by the model, ranging from 0 to 1, where higher values indicate a better fit.These metrics provide insights into the accuracy and goodness of fit of our models.

Below are the tables comparing the Root Mean Squared Logarithmic Error (RMSLE) and R-squared values for four different regression algorithms: linear regression with ridge regularization, decision trees, random forest, and XGBoost. These tables provide insights into the performance of each algorithm with and without the inclusion of geospatial data. By comparing the RMSLE and R-squared values across different models and data configurations, we can evaluate the impact of geospatial data on the predictive accuracy of each regression algorithm.

Table 1: Comparison of RMSLE for Regression Models with and without Geospatial Data.

Model	Without Geospatial Data	With Geospatial Data
Linear Regression	0.52	0.50
Decision Trees	0.45	0.41
Random Forest	0.35	0.32
XGBoost	0.32580	0.30229

Table 2: Comparison of R-Squared for Regression Models with and without Geospatial Data.

Model	Without Geospatial Data	With Geospatial Data
Linear Regression	0.55	0.57
Decision Trees	0.71	0.73
Random Forest	0.81	0.83
XGBoost	0.82128	0.84614

6.2 Scatter Plot

Additionally, it's essential to check for assumptions such as linearity, homoscedasticity, and normality of residuals to ensure the validity of the linear regression model. These aspects play a crucial role in understanding the data and building a reliable predictive model.

A scatter plot is used to visualize the relationship between the measured log-scaled prices and the predicted log-scaled prices. This plot helps in assessing the model's predictive performance by comparing the actual values with the predicted values. The diagonal dashed line represents perfect predictions, where the measured and predicted values are identical.

Model without Geospatial Data:

- This model shows a broader dispersion of data points around the diagonal, reflecting less accuracy in predictions.
- The absence of geospatial data leads to a less robust model, lacking the spatial dependencies crucial for accurately predicting real estate prices.

Model with Geospatial Data:

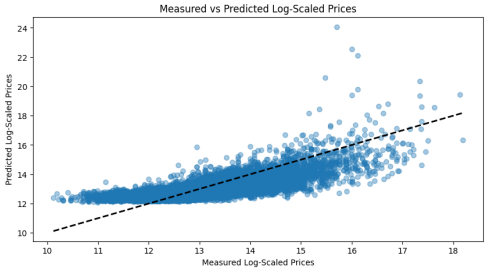
- The inclusion of geospatial data in the model results in a tighter clustering of predicted values around the diagonal, indicating higher predictive accuracy.
- This suggests that geospatial factors are significant predictors of real estate prices, contributing to more precise estimations.

These graphs in fig. 7 and fig. 8 provide valuable insights into the model's performance and feature importance, aiding in the interpretation and evaluation of the results.

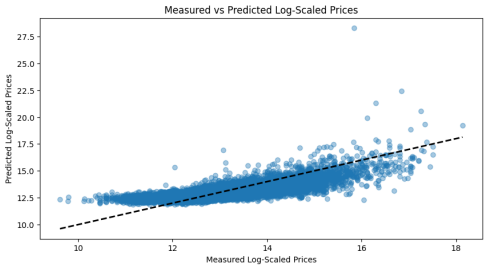
7. Conclusion

In addressing the challenge of price prediction, which inherently involves regression, we initiated our approach with Linear Regression. To enhance its predictive capabilities, we applied L2 regularization, specifically Ridge Regression with cross-validation. Despite our efforts, the results fell short of expectations. Subsequently, we ventured into more sophisticated techniques, namely Decision Trees and Random Forests, seeking improved performance and accuracy.

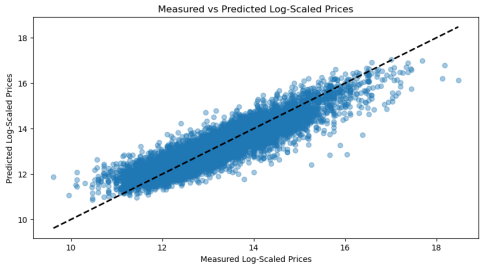
The findings demonstrate that the XGBoost model surpasses other regression models in terms of performance. Not only is XGBoost faster compared to its counterparts, but its effectiveness is significantly enhanced by the use of gradient descent and regularization techniques. Furthermore,



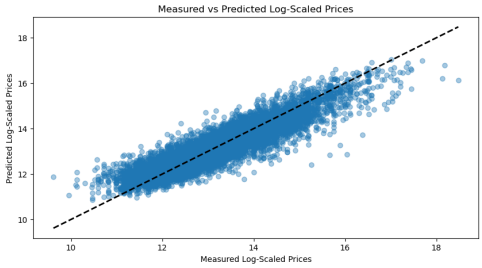
(a) Ridge Regression



(b) Decision Tree

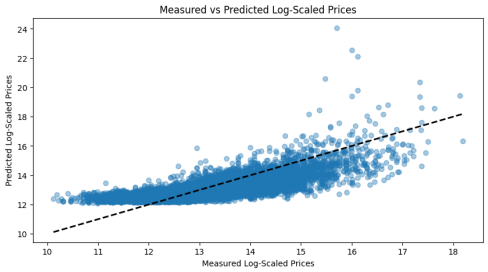


(c) Random Forest Regression

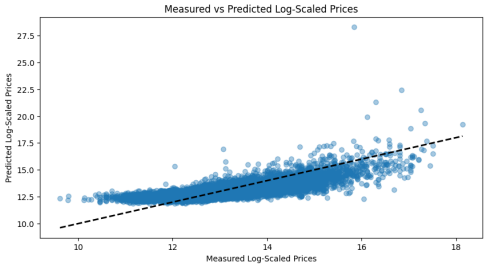


(d) XGBoost

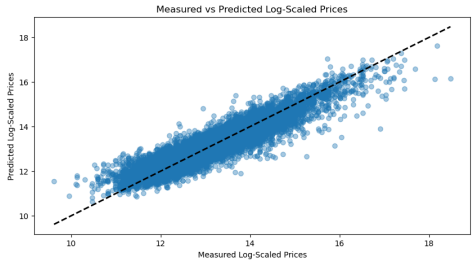
Figure 7: Scatter Plot of each model without geospacial data



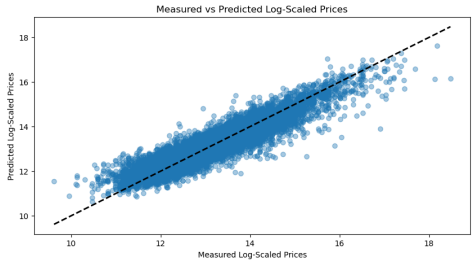
(a) Ridge Regression



(b) Decision Tree



(c) Random Forest Regression



(d) XGBoost

Figure 8: Scatter Plot of each model with geospacial data

incorporating geospatial data into the model further improves accuracy, providing a substantial and non-trivial method for enhancing performance with real-time data.

References

[1]T. Chen and C. Guestrin, *Xgboost: a scalable tree boosting system*, 2016.

[2]Florida Geographic Data Library, *FGDL Map*, https://fgdl.org/fgdlmap/, Accessed: ¡date¿.

[3]T. O. Hodson, T. M. Over, and S. S. Foks, «Mean squared error, deconstructed», Journal of Advances in Modeling Earth Systems **13**, e2021MS002681 (2021) 10.1029/2021MS002681.

[4]S. Malik, M. A. Khan, M. Ramzan, and I. Ur Rehman, «Regularization of Logistic Regression for Forecasting: A Comparative Study», SAGE Open **11**, 21582440211059047 (2021) 10.1177/00368504211059047.

[5]A. Natekin and A. Knoll, «Gradient boosting machines, a tutorial», Frontiers in Neurorobotics **7**, 21 (2013) 10.3389/fnbot.2013.00021.

[6]A. S. Sakib, *Usa real estate dataset*, https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset.

[7]J. L. Speiser, M. E. Miller, J. Tooze, and E. Ip, «A comparison of random forest variable selection methods for classification prediction modeling», Expert Systems with Applications **134**, 93–101, ISSN: 0957-4174 (2019) https://doi.org/10.1016/j.eswa.2019.05.028.

[8]Z. Zhang, «Decision trees for objective house price prediction», in 2021 3rd international conference on machine learning, big data and business intelligence (mlbdbi) (2021), pp. 280–283, 10.1109/MLBDBI54094.2021.00059.