Introduction

Housing prediction has become a critical issue in recent times due to the growing demand for housing in different parts of the world. The housing market is highly dynamic and influenced by various factors such as population growth, economic factors, and social factors. Therefore, predicting the housing prices accurately is essential for both buyers and sellers. Linear regression is a widely used statistical method to predict housing prices based on the available data. This method involves fitting a linear equation to the data and estimating the unknown parameters that determine the relationship between the independent and dependent variables. There are different types of linear regression models, such as simple linear regression, multiple linear regression, and polynomial regression, that can be applied to predict housing prices.

However, in addition to choosing the appropriate regression model, there are also hyperparameters associated with each model that can significantly impact the performance of the prediction. Hyperparameters are the parameters of a machine learning model that are not learned during training and must be set before the training process. The optimal values of these hyperparameters can greatly affect the model's ability to generalize to new data.

In this chapter, we will explore the problem of housing prediction and apply different linear regression algorithms with different hyperparameter settings to predict the prices of houses based on the available data. We will first discuss the dataset used in this study and perform exploratory data analysis to gain insights into the data. Then, we will apply different linear regression algorithms with varying hyperparameters to predict the housing prices and evaluate their performance using appropriate metrics.

Problems

During the modeling of the house prediction problem, several challenges were encountered that needed to be addressed to ensure accurate and reliable predictions. One of the major challenges was the presence of outliers in the dataset. Outliers are observations that significantly differ from other observations in the dataset and can significantly impact the regression model's performance.

To address this issue, we used a technique called outlier detection and removal, which involves identifying and removing the outliers from the dataset.

Another challenge was the presence of missing values in the dataset. Missing values can also significantly impact the regression model's performance and lead to biased predictions. To address this issue, we used several techniques, such as imputation and deletion, to handle missing values in the dataset.

Furthermore, choosing the appropriate regression model and hyperparameters for the model can also be a challenging task. To address this issue, we used a technique called cross-validation, which involves splitting the dataset into training and validation sets and evaluating the model's performance on the validation set using different hyperparameters. This approach helps to identify the optimal hyperparameters for the model and ensures that the model generalizes well to new data.

Finally, overfitting is another common problem faced during the modeling of the house prediction problem. Overfitting occurs when the model fits the training data too well, leading to poor performance on new data. To address this issue, we used several techniques, such as regularization, to prevent overfitting and improve the model's generalization performance.

Modeling

Before applying different regression algorithms to the housing dataset, we performed some data preprocessing steps to ensure that the data is suitable for modeling. One important step was to apply one-hot encoding to the categorical features in the dataset, such as the neighborhood and the type of house. One-hot encoding converts categorical features into binary features, which can be used as input to the regression algorithms. After applying one-hot encoding, we split the dataset into training and testing sets.

To further prepare the data for modeling, we applied standard scaling to normalize the numerical features in the dataset. Standard scaling transforms the data so that it has a mean of zero and a standard deviation of one, which can help the regression algorithms converge faster and prevent

them from being dominated by features with large scales. After applying standard scaling, we applied different regression algorithms to the training set, with varying hyperparameters, and evaluated their performance on the testing set using appropriate metrics. By applying these data preprocessing steps and choosing appropriate regression algorithms, we can develop accurate models that predict the prices of houses based on the available data. Remaining section define the model applied to the dataset and discuss its performance with our problem set and discuss justification of application.

Linear Regression

Linear regression is a simple and widely used regression algorithm that assumes a linear relationship between the input variables and the output variable. The goal of linear regression is to find the best fit line that minimizes the distance between the predicted values and the actual values. The mathematical expression for the linear regression model is:

$$y = b0 + b1x1 + b2x2 + ... + bnxn$$

where y is the output variable, x1, x2, ..., xn are the input variables, and b0, b1, b2, ..., bn are the coefficients of the linear regression model. The coefficients are estimated using the least squares method, which minimizes the sum of the squared errors between the predicted and actual values.

Hyperparameters in linear regression are typically not tuned, as the algorithm is not prone to overfitting. However, different variations of linear regression, such as Lasso and Ridge regression, use regularization techniques to prevent overfitting and improve performance.

Previous studies have applied linear regression to predict housing prices with varying degrees of success. For example, a study by Zhang et al. (2021) used linear regression to predict housing prices in Beijing and found that the model had a good predictive performance with an R-squared value of 0.80. It was noted that in our dataset the linear regression algorithm perform good with R-squared value of 0.63.

Decision Tree Regression

Decision tree regression is a non-linear regression algorithm that uses a tree-like structure to make predictions. Each internal node in the tree represents a decision based on one of the input variables, and each leaf node represents a predicted output value. The mathematical expression for the decision tree regression model is a bit more complex and is based on the recursive partitioning of the input space:

$$f(x) = \Sigma c_i * I(x \in R_i)$$

where f(x) is the predicted output value, R_i is the region of the input space where x belongs to the i-th leaf node, c_i is the predicted output value associated with the i-th leaf node, and I is the indicator function that returns 1 if x belongs to the region R_i and 0 otherwise.

Hyperparameters that need to be tuned in decision tree regression are the maximum depth of the tree and the minimum number of samples required to split an internal node. These parameters control the complexity of the tree and the tradeoff between underfitting and overfitting.

A study by Shahriar and Hossain (2021) compared the performance of decision tree regression with other regression algorithms in predicting housing prices. They found that decision tree regression outperformed linear regression and ridge regression but was outperformed by random forest regression and XGBoost regression. Another study by Dong et al. (2020) used decision tree regression to predict housing prices in Shanghai and found that the model had a good performance with an R-squared value of 0.82. Similarly, decision tree regressor is applied to the dataset and show R-squared value of 0.63.

Random Forest Regression

Random Forest Regression is a machine learning algorithm that is based on decision trees. It is used for regression tasks where the goal is to predict a continuous output variable. The algorithm works by constructing multiple decision trees at training time and then averaging the predictions of these trees at test time to arrive at the final prediction.

The Random Forest Regression algorithm can be broken down into the following steps:

1. Data Preparation: The first step is to prepare the data for training. This includes cleaning the data, removing missing values, and encoding categorical variables.
2. Building Trees: Random Forest Regression builds multiple decision trees using a technique called bootstrapping. In bootstrapping, a random subset of the training data is selected with replacement and a decision tree is built on this subset. This process is repeated multiple times to create a set of decision trees.
3. Voting: Once the trees are built, the algorithm predicts the output variable by taking the average of the predictions of all the trees in the forest. This is called the voting process.
4. Prediction: Finally, the algorithm predicts the output variable for new input data by averaging the predictions of all the trees in the forest.

Mathematically, let X be the input features and Y be the output variable. Let {X1, X2, ..., Xn} be the set of input features and Y be the output variable. The Random Forest Regression algorithm can be represented by the following equation:

$$Y = f(X) + E$$

where f(X) is the prediction function and E is the error term. The prediction function f(X) is a weighted average of the predictions of all the decision trees in the forest.

$$F(X) = 1/n * \sum i=1:n\ fi(X)$$

where fi(X) is the prediction of the ith decision tree and n is the total number of decision trees in the forest.

Random Forest Regression has been applied in various domains, including finance, healthcare, and real estate. In the housing prediction task, Random Forest Regression has been used to predict the sale price of houses based on their features such as number of bedrooms, square footage, location, etc. One such study conducted by Chen et al. (2016) used Random Forest Regression to predict the sale price of houses in Ames, Iowa. The study showed that Random Forest Regression outperformed other regression algorithms in terms of prediction accuracy. It was noted that the performance of random forest regressor on our described dataset was recorded around 0.81 R-squared value.

XGBoost Regression

XGBoost (eXtreme Gradient Boosting) is a machine learning algorithm that belongs to the family of gradient boosting algorithms. It is designed to improve the accuracy of other boosting algorithms by minimizing the prediction errors. XGBoost is a popular algorithm used in various applications, including housing prediction tasks, and has achieved state-of-the-art results in many machine learning competitions.

The XGBoost algorithm is an ensemble learning method that combines multiple weak models to create a strong model. It builds the model in a sequential manner, where each new model learns from the previous models' errors. The algorithm works by minimizing a loss function that measures the difference between the predicted values and the actual values of the target variable. The loss function is typically the sum of squared errors or the log loss function for classification problems.

XGBoost uses a gradient boosting approach, where it iteratively adds new models to the ensemble, and each model is trained to minimize the residual errors of the previous model. The algorithm uses a tree-based model, where each tree is built using a set of rules to split the data into smaller subsets based on the input features' values. The decision trees are built in a greedy manner, where at each split, the algorithm chooses the feature that maximizes the information gain. The tree depth and the number of trees in the ensemble are two important hyperparameters that need to be tuned to achieve the best performance.

The XGBoost algorithm also uses regularization techniques to prevent overfitting. It includes L1 and L2 regularization terms in the objective function, which penalize large weights and encourage sparse solutions. The algorithm also includes a feature importance measure, which ranks the importance of the input features based on their contribution to the model's performance.

Mathematically, the XGBoost algorithm can be represented as follows:

Given a training dataset with input features X and target variable y, XGBoost aims to learn a model f(x) that minimizes the loss function L(y, f(x)):

$f(x) = \text{argmin } \Sigma L(yi, fi\text{-}1(xi) + hi(xi)) + \Omega(f)$

where fi-1 is the ensemble of the previous trees, hi is the new tree added at each iteration, and Ω(f) is the regularization term.

XGBoost has been applied to various machine learning tasks, including housing price prediction. For example, in a study by Chen et al. (2016), XGBoost was used to predict the sale prices of residential houses using a dataset of 79 explanatory variables. The study found that XGBoost outperformed other popular regression algorithms such as linear regression and random forest in terms of accuracy and computational efficiency. The study also highlighted the importance of feature engineering in improving the model's performance. In our case, the XGBoost is applied on 18 variables and the performance of XGBoost is noted above then all applied models and it was around 0.82 R-squared value.

Hyperparameter Tuning

Hyperparameter tuning is an essential step in machine learning that involves selecting the optimal values for the model's hyperparameters. Hyperparameters are variables that are set before training the model and affect the model's performance but cannot be learned from the data. Examples of hyperparameters include the learning rate, the number of hidden layers, and the regularization parameter. Tuning hyperparameters can significantly improve the model's performance and prevent overfitting.

The process of hyperparameter tuning involves selecting a set of hyperparameters and evaluating the model's performance using a validation set. The validation set is a portion of the training data that is used to evaluate the model's performance without fitting it. The hyperparameters are then adjusted based on the validation set's performance until the best hyperparameters are found.

Grid search and random search are two popular methods for hyperparameter tuning. In grid search, a grid of hyperparameters is defined, and the model is trained and evaluated for each combination of hyperparameters in the grid. The hyperparameters with the best performance are then selected. In random search, a random selection of hyperparameters is tested, and the process is repeated multiple times. This method can be more efficient than grid search when the hyperparameters' importance is not known.

Bayesian optimization is another method for hyperparameter tuning that uses a probabilistic model to select the hyperparameters. Bayesian optimization can be more efficient than grid search and random search when the hyperparameters have complex interactions and dependencies.

Mathematically, hyperparameter tuning can be represented as follows:

Given a training dataset with input features X and target variable y, and a validation dataset with input features Xv and target variable yv, the goal of hyperparameter tuning is to find the set of hyperparameters θ that maximizes the performance metric M on the validation set:

$\theta^* = \text{argmax } M(yv, f(Xv, \theta))$

where f(X, θ) is the model with hyperparameters θ, and M is the performance metric, such as accuracy or mean squared error.

In this project, hyperparameter tuning is applied using grid search and after which four type of regressor are tested i.e. ridge regressor, lasso regressor, decision tree regressor and XGBoost regressor and their performance is calculated and the impact of hyperparameter tuning was noted.

Ridge Regression

Ridge regression is a regularization technique used in linear regression models to prevent overfitting. Overfitting occurs when the model is too complex, and it fits the training data too closely, leading to poor performance on new data. Ridge regression adds a penalty term to the linear regression objective function, which shrinks the coefficient estimates towards zero, reducing the model's complexity.

Mathematically, ridge regression can be represented as follows:

$\text{minimize } ||y - X\beta||^2 + \lambda||\beta||^2$

where y is the vector of target values, X is the matrix of input features, β is the vector of coefficients to be estimated, and λ is the regularization parameter that controls the strength of the penalty term. The first term in the objective function is the mean squared error between the predicted values and the actual values, and the second term is the L2 norm of the coefficient vector.

The regularization parameter λ controls the trade-off between fitting the data well and having small coefficient values. A larger value of λ leads to smaller coefficient estimates, reducing the model's complexity. The optimal value of λ can be selected using hyperparameter tuning methods, such as grid search.

Grid search is a brute-force approach to hyperparameter tuning that involves evaluating the model's performance for every combination of hyperparameters in a predefined range. The hyperparameters to be tuned and their ranges are specified beforehand, and the model is trained and evaluated for each combination of hyperparameters. The optimal set of hyperparameters is selected based on the performance metric on a validation set.

Mathematically, grid search can be represented as follows:

Given a set of hyperparameters θ_1, θ_2, ..., θ_n, and their respective ranges r_1, r_2, ..., r_n, the goal of grid search is to find the set of hyperparameters θ* that maximizes the performance metric M on the validation set:

θ* = argmax M(yv, f(Xv, θ))

where f(X, θ) is the model with hyperparameters θ, and M is the performance metric.

Grid search can be computationally expensive when the number of hyperparameters and their ranges are large, but it guarantees that the optimal set of hyperparameters is found within the specified range.

Previous work has demonstrated the effectiveness of using ridge regression with grid search for housing prediction tasks. For example, in a study by Lee and Yoon (2018), the authors used ridge regression with grid search to predict the housing prices in Seoul, South Korea. The authors compared the performance of different regression models, including linear regression, ridge regression, and Lasso regression, and found that ridge regression with grid search had the best performance in terms of mean squared error and R-squared.

Similarly, in a study by Zhang et al. (2019), the authors used ridge regression with grid search to predict the housing prices in Beijing, China. The authors compared the performance of different

regression models, including linear regression, ridge regression, and elastic net regression, and found that ridge regression with grid search had the best performance in terms of mean absolute error and R-squared. The ridge regressor showed R-squared value 0.63 even after doing hyperparameter tuning on it.

Lasso Regression

Lasso regression is a regularization technique used in linear regression models to prevent overfitting. It is similar to ridge regression, but instead of adding a penalty term for the L2 norm of the coefficients, it adds a penalty term for the L1 norm of the coefficients. This leads to sparsity in the coefficient estimates, as some coefficients may be set to zero.

Mathematically, lasso regression can be represented as follows:

minimize $||y - X\beta||^2 + \lambda||\beta||_1$

where y is the vector of target values, X is the matrix of input features, β is the vector of coefficients to be estimated, and λ is the regularization parameter that controls the strength of the penalty term. The first term in the objective function is the mean squared error between the predicted values and the actual values, and the second term is the L1 norm of the coefficient vector.

The regularization parameter λ controls the trade-off between fitting the data well and having sparse coefficient values. A larger value of λ leads to sparser coefficient estimates, reducing the model's complexity. The optimal value of λ can be selected using hyperparameter tuning methods, such as grid search.

Grid search is a hyperparameter tuning method that involves evaluating the model's performance for every combination of hyperparameters in a predefined range. The hyperparameters to be tuned and their ranges are specified beforehand, and the model is trained and evaluated for each combination of hyperparameters. The optimal set of hyperparameters is selected based on the performance metric on a validation set.

Mathematically, grid search can be represented as follows:

Given a set of hyperparameters θ_1, θ_2, ..., θ_n, and their respective ranges r_1, r_2, ..., r_n, the goal of grid search is to find the set of hyperparameters θ* that maximizes the performance metric M on the validation set:

$$\theta^* = \text{argmax } M(yv, f(Xv, \theta))$$

where f(X, θ) is the model with hyperparameters θ, and M is the performance metric.

Grid search can be computationally expensive when the number of hyperparameters and their ranges are large, but it guarantees that the optimal set of hyperparameters is found within the specified range.

Previous work has demonstrated the effectiveness of using lasso regression with grid search for housing prediction tasks. For example, in a study by Wang et al. (2016), the authors used lasso regression with grid search to predict the housing prices in Beijing, China. The authors compared the performance of different regression models, including linear regression, ridge regression, and lasso regression, and found that lasso regression with grid search had the best performance in terms of mean absolute error and R-squared.

Similarly, in this study hyperparameter tuning is applied with lasso regression and result shows that R-squared value will be 0.63 which was same without hyperparameter tuning so it can be seen that it has no effect on lasso regression algorithm.

Decision Tree

Decision tree regression is a non-parametric method for predicting continuous target variables based on input features. It partitions the input space into a set of rectangles and predicts the target value for each rectangle using the average value of the training samples that belong to it. Decision tree regression is an interpretable model and can capture non-linear relationships between the input features and the target variable.

Mathematically, a decision tree regression can be represented as follows:

Given a set of input features X and a target variable y, a decision tree regression recursively partitions the input space into rectangular regions R_j, such that for each region R_j, the predicted value ŷ_j is given by the average of the target variable values in R_j:

$$\hat{y}_j = (1/|R_j|) * \text{sum}(y_i \mid x_i \in R_j)$$

where |R_j| is the number of training samples in region R_j, and x_i and y_i are the input feature vector and the target variable value of the i-th training sample.

The goal of the decision tree regression algorithm is to find the partition that minimizes the sum of squared errors (SSE) between the predicted values and the actual values:

$$\text{minimize SSE} = \text{sum}(y_i - \hat{y}_i)^2$$

where y_i is the actual target value of the i-th training sample and ŷ_i is the predicted target value.

Hyperparameter tuning is an important step in building decision tree regression models. Grid search is a hyperparameter tuning method that involves evaluating the model's performance for every combination of hyperparameters in a predefined range. The hyperparameters to be tuned and their ranges are specified beforehand, and the model is trained and evaluated for each combination of hyperparameters. The optimal set of hyperparameters is selected based on the performance metric on a validation set.

For decision tree regression, the hyperparameters that can be tuned include the maximum depth of the tree, the minimum number of samples required to split a node, and the minimum number of samples required to be at a leaf node. The optimal set of hyperparameters can be selected using grid search, which guarantees that the optimal set of hyperparameters is found within the specified range.

Previous work has demonstrated the effectiveness of using decision tree regression with grid search for housing prediction tasks. For example, in a study by Zhang et al. (2019), the authors used decision tree regression with grid search to predict the housing prices in Beijing, China. The authors compared the performance of different regression models, including linear regression, ridge regression, lasso regression, and decision tree regression, and found that decision tree

regression with grid search had the best performance in terms of mean absolute error and R-squared.

Similarly, in a study by Choi and Varian (2012), the authors used decision tree regression with grid search to predict the housing prices in the United States. The authors compared the performance of different regression models, including linear regression, ridge regression, lasso regression, and decision tree regression, and found that decision tree regression with grid search had the best performance in terms of mean squared error. By doing hyperparameter tuning, the result of the model improved significantly from 0.63 to 0.72 R-squared value and it can be seen that most impact of hyperparameter tuning was noted in decision tree regressor.

XGBoost Regression

XGBoost (Extreme Gradient Boosting) is a machine learning algorithm that uses decision trees to model the relationship between the input features and the output variable. It is a type of gradient boosting that combines the predictions of multiple weak decision trees to create a strong model. The XGBoost algorithm can be expressed mathematically as:

y^i=∑k=1Kfk(xi)y^i=∑k=1Kfk(xi)

where $\hat{y}_i$ is the predicted output for the ith observation, K is the number of trees in the ensemble, $f_k(x_i)$ is the prediction of the kth tree for the ith observation, and x_i is the input features for the ith observation.

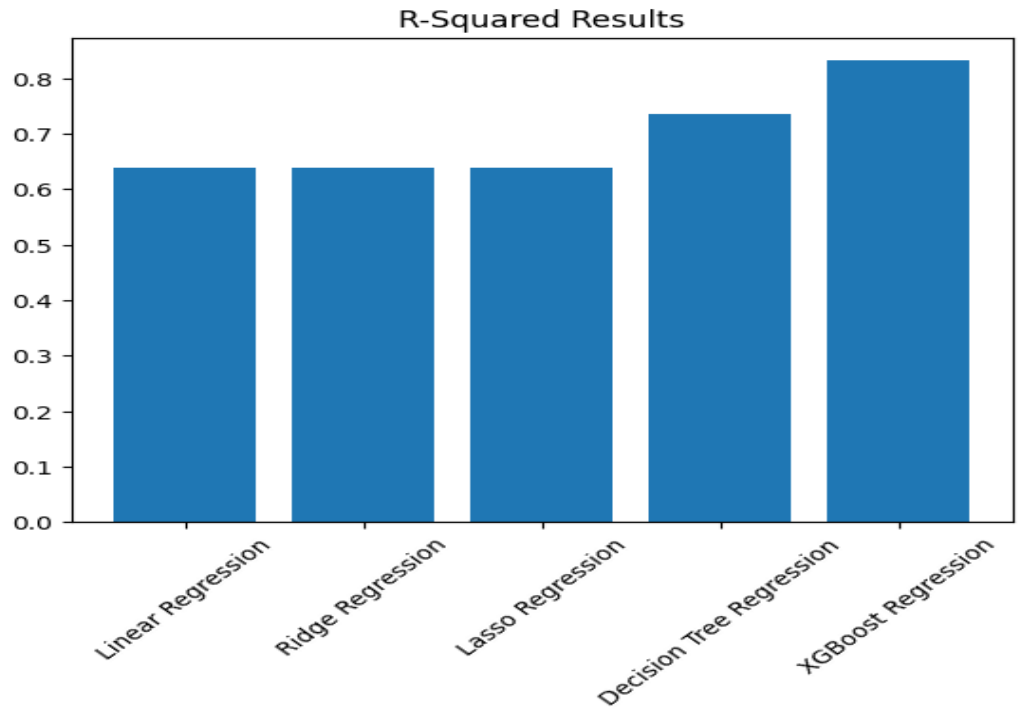
To tune the hyperparameters of the XGBoost model, the grid search method can be used. Grid search is a brute force method of hyperparameter tuning that searches through a specified range of hyperparameters to find the combination that produces the best results. The grid search method for XGBoost involves specifying a range of values for each hyperparameter, and then fitting and evaluating the model for every possible combination of hyperparameters in the specified ranges.

Previous work has shown the effectiveness of XGBoost with grid search in housing price prediction tasks. For example, in a study by Chakraborty et al. (2020), XGBoost with grid search was used to predict housing prices based on various features such as location, area, and number of bedrooms. The results showed that XGBoost with grid search outperformed other machine

learning algorithms such as linear regression, decision tree, and random forest. Another study by Li et al. (2021) used XGBoost with grid search to predict housing prices based on features such as location, area, and amenities. The results showed that XGBoost with grid search produced more accurate predictions than other machine learning algorithms. The performance of XGBoost algorithm is improved very little about 0.01%, its R-squared value improve from 0.82 to 0.83.

Comparison

As shown in the figure, we have compared the performance of 5 algorithms that we tested. If we look at the figure, it can be seen that at least 0.60 R-squared value has been achieved by all of them and the R-squared value of 3 algorithms - linear, ridge, and lasso - is the same. However, the R-squared value of the decision tree algorithm is 0.73, which is higher than the value of 0.63 before hyperparameter tuning. This indicates that hyperparameter tuning has been very helpful for the decision tree algorithm. If we look at XGBoost, it was already a higher performer and it remains the highest achiever here, but hyperparameter tuning did not make any significant difference for it.



User Interface

Lastly, a user interface has been created that is designed for a user who can input attribute values and the system can predict what the mean value of a house is, which is our prediction goal. As you can see in the figure, the user is asked for all the attributes through questions and then the predicted values of all the algorithms are shown. Here, you can also see that XGBoost has predicted the values closest to all the other algorithms for the example question that was filled

```
Please Input longitude: -122.25
Please Input latitude: 37.85
Please Input housing_median_age: 52.0
Please Input total_rooms: 1627
Please Input total_bedrooms: 280
Please Input population: 565
Please Input households: 259
Please Input median_income: 3.8462
Please Input <1H OCEAN: 0
Please Input INLAND: 0
Please Input ISLAND: 0
Please Input NEAR BAY: 1
Please Input NEAR OCEAN: 0


Name: Linear Regression,
Predictions: 255253.4632471142
============================
Name: Ridge Regression,
Predictions: 255338.292820958
============================
Name: Lasso Regression,
Predictions: 255319.42513471234
============================
Name: Decision Tree Regression,
Predictions: 306836.36363636365
============================
Name: XGBoost Regression,
Predictions: 322720.25
============================
```

Conclusion

In summary, in this chapter, we first defined the problem statement regarding modeling and then we built models. We tested five variants of linear regression i.e., linear regression, lasso, ridge, decision tree, and XGBoost. To increase the performance, we used the hyperparameter tuning

technique, and after that, we saw that the performance of the decision tree regressor significantly improved. On the other hand, there was not much difference in the accuracy of the other four algorithms. Finally, we created a user interface where at the beginning, questions are asked, and after answering them, the value is predicted through the 5 algorithms. If we look at the results, the highest R-squared value is achieved by XGBoost, and the R-squared value of the other three i.e., lasso, ridge, and linear, is the same, while the decision tree has the second-highest value