House Price Prediction

Team members details

MADA SRINIDHI (RA2011026010355)

# Problem Statement:

The house price prediction problem involves developing a model that can accurately predict the selling price of a house based on various features such as the size of the house, the number of bedrooms and bathrooms, the location of the house, and other relevant factors. The goal is to build a model that can help real estate agents or homeowners determine a reasonable selling price for a house based on its characteristics.

Literature Review

|  |  |
| --- | --- |
| Literature | Review |
| Overfitting  Feature importance  Limited interpretability  Hyperparameter tuning  Limited extrapolation  Limited scalability | Random Forest models can overfit the training data if the number of trees and the maximum depth of each tree are not carefully selected. Regularization techniques such as pruning can help avoid overfitting.  Random Forest models can provide an estimation of the importance of each feature used to make predictions. However, this estimation is based on the training data and may not generalize well to new data.  Random Forest models can be difficult to interpret compared to linear regression models. Understanding how the model arrives at its predictions can be challenging, especially if the model is using a large number of trees  Random Forest models require careful selection of hyperparameters such as the number of trees, maximum depth, and minimum samples per leaf. Choosing the right hyperparameters can have a significant impact on model performance.  Random Forest models are not well suited for extrapolation beyond the range of the training data. Predictions may become less accurate when the model is asked to make predictions for values outside the range of the training data.  Training a Random Forest model on large datasets can be time-consuming and resource-intensive. Additionally, the model may not be able to handle datasets with a large number of features. |

# Limitations with citations:

Lack of data: The accuracy of random forest models is highly dependent on the quality and quantity of data available. If there is a lack of relevant data on house prices, the model may not be able to accurately predict house prices.

Limited feature selection: Random forest models require feature selection to identify the most important predictors for house price prediction. However, this process can be limited by the quality and quantity of data available. If there are not enough relevant features available, the model may not be able to accurately predict house prices.

Overfitting: Random forest models can be prone to overfitting if the model is too complex or if there is noise in the data. Overfitting occurs when the model is trained on the training data too well, resulting in poor performance on the test data.

Lack of interpretability: Random forest models can be difficult to interpret, making it challenging to understand why certain predictions are made. This can be problematic in fields like real estate, where understanding the reasoning behind a particular prediction is important.

Limited extrapolation ability: Random forest models are not well-suited for extrapolation beyond the range of the training data. This means that the model may not be able to accurately predict house prices for properties that are significantly different from the properties in the training data.

Overall, while random forest can be a useful tool for house price prediction, it is important to be aware of its limitations and to carefully consider the quality and quantity of data available when building and interpreting models.

# Objective:

The objective of the house price prediction problem is to develop a model that can accurately predict the selling price of a house based on various features. The model aims to assist real estate agents or homeowners in determining a reasonable selling price for a house based on its characteristics.

The model's accuracy is crucial as an overestimated price may lead to the house remaining on the market for too long, while an underestimated price may result in a significant financial loss for the homeowner. Therefore, the model's primary objective is to minimize the difference between the predicted price and the actual selling price of a house.

# Technical Depth:

House price prediction using random forest is a common application of machine learning in the real estate industry. Random forest is an ensemble learning technique that combines multiple decision trees to improve the accuracy and stability of predictions. In this context, a decision tree represents a set of rules that predict the price of a house based on its features, such as location, size, number of bedrooms, etc.

Here is a technical overview of the steps involved in house price prediction using

random forest:

Data collection and preprocessing: The first step is to collect data on houses and their features, such as location, size, number of bedrooms, etc. This data can be obtained from various sources, such as real estate websites, government databases, or private datasets. Once the data is collected, it needs to be cleaned, preprocessed, and transformed into a format suitable for machine learning. This may involve removing missing values, encoding categorical variables, scaling numerical variables, and splitting the data into training and testing sets.

Feature selection: The next step is to select the most relevant features for predicting house prices. This can be done using various techniques, such as correlation analysis, feature importance ranking, or domain expertise. The goal is to identify the features that have the strongest impact on the target variable (house price) and remove any irrelevant or redundant features.

Random forest training: Once the features are selected, the next step is to train a random forest model on the training data. A random forest is a collection of decision trees, each of which is trained on a random subset of the features and a random subset of the training data. This helps to reduce overfitting and improve the generalization performance of the model. During training, the model learns to predict the house price based on the selected features.

Model evaluation: After training, the model needs to be evaluated on the testing data to measure its performance. This can be done using various metrics, such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or R-squared. The goal is to select the best model that achieves the highest accuracy and the lowest error on the testing data.

Model deployment: Once the model is trained and evaluated, it can be deployed to make predictions on new, unseen data. This can be done using various techniques, such as batch processing, real-time scoring, or API integration. The goal is to provide accurate and reliable predictions to help buyers and sellers make informed decisions about buying or selling houses.

In summary, house price prediction using random forest involves collecting and preprocessing data, selecting relevant features, training a random forest model, evaluating its performance, and deploying it to make predictions. This requires a combination of domain expertise, statistical knowledge, and programming skills to build a robust and accurate model.

# Proposed Methodology:

Data collection: Collect a dataset of housing features and corresponding prices from a reliable source. The dataset should include a variety of features such as the number of bedrooms, square footage, location, and age of the property.

Data preprocessing: Perform data cleaning, normalization, and feature selection on the dataset. This step involves removing irrelevant features, handling missing data, and scaling the features to a common range.

Splitting data: Split the dataset into training and testing sets. The training set will be used to train the random forest model, while the testing set will be used to evaluate the model's performance.

Building the model: Build a random forest model using the training data. The model should be optimized to reduce overfitting by using cross-validation techniques.

Model evaluation: Evaluate the model's performance on the testing set. This step involves calculating metrics such as mean squared error (MSE), mean absolute error (MAE), and R-squared to assess the accuracy of the model.

Model tuning: Adjust the hyperparameters of the random forest model to improve its performance. Hyperparameters such as the number of trees, the maximum depth of the trees, and the minimum number of samples required to split a node can be adjusted to optimize the model.

Final evaluation: Evaluate the final model's performance on a holdout dataset. This step involves using a separate dataset that was not used for training or testing to assess the model's generalizability.

Deployment: Deploy the model for use in real-world applications. The model can be used to predict house prices for new properties based on their features.

Overall, this methodology involves collecting and preprocessing data, building and tuning a random forest model, and evaluating the model's performance on testing and holdout datasets. The final model can be deployed for use in real-world applications.

# Architecture Diagram:

------------------------------------------------------------------------------------------+

| Input Data |

| (House features such as area, number of bedrooms, etc.) |

+-------------------------------------------------------------------------------------------+

|

|

v

+-------------------------------------------------------------------------------------------+

| Data Preprocessing |

| (Cleaning data, handling missing values, feature engineering, etc.) |

+-------------------------------------------------------------------------------------------+

|

|

v

+-------------------------------------------------------------------------------------------+

| Splitting the Data |

| (Splitting the data into training and testing sets for model evaluation and validation) |

+-------------------------------------------------------------------------------------------+

|

|

v

+-------------------------------------------------------------------------------------------+

| Random Forest |

| (Building and training the Random Forest model on the data) |

+-------------------------------------------------------------------------------------------+

|

|

v

+-------------------------------------------------------------------------------------------+

| Model Evaluation |

| (Evaluating the performance of the model on the testing set, adjusting model parameters) |

+-------------------------------------------------------------------------------------------+

|

|

v

+-------------------------------------------------------------------------------------------+

| Prediction |

| (Using the trained model to make predictions on new, unseen data points) |

+-------------------------------------------------------------------------------------------+

|

|

v

+-------------------------------------------------------------------------------------------+

| Output |

| (Predicted house prices based on the input features) |

|  |
| --- |
|  |

# Modules Description :

Bottom of Form

Data Loading: This module is responsible for loading the input data, which contains information about the houses, such as their size, number of bedrooms, location, etc.

Data Preprocessing: This module is responsible for cleaning the data and preparing it for training the model. It may involve tasks such as handling missing values, converting categorical features into numerical features, scaling the data, and feature engineering.

Data Splitting: This module is responsible for splitting the data into training and testing sets. The training set is used to train the Random Forest model, while the testing set is used to evaluate the performance of the model.

Random Forest: This module is responsible for building and training the Random Forest model. It involves defining the number of trees in the forest, the maximum depth of the trees, the minimum number of samples required to split a node, and other hyperparameters.

Model Evaluation: This module is responsible for evaluating the performance of the trained Random Forest model on the testing set. It may involve calculating various performance metrics, such as mean absolute error, mean squared error, and R-squared.

Model Tuning: This module is responsible for tuning the hyperparameters of the Random Forest model to improve its performance. It may involve techniques such as grid search, random search, and Bayesian optimization.

Prediction: This module is responsible for using the trained Random Forest model to make predictions on new, unseen data points. It involves passing the input data through the trained model and obtaining the predicted house prices.

Output: This module is responsible for outputting the predicted house prices to the user in a suitable format, such as a CSV file or a graphical visualization.

By using these modules, a complete pipeline can be constructed for house price prediction using Random Forest, which can be used to automate the process of predicting house prices and make it more efficient and accurate.

# References:

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: with Applications in R. Springer.

Kaggel. (2021). House Prices: Advanced Regression Techniques. Retrieved from https://www.kaggle.com/c/house-prices-advanced-regression-techniques

Python Data Science Handbook. (2021). Supervised Learning: Random Forests. Retrieved from https://jakevdp.github.io/PythonDataScienceHandbook/05.08-random-forests.html

Scikit-learn. (2021). Ensemble methods: Random Forest. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html