**Introduction**

The prediction of house prices plays a crucial role in the real estate industry, financial planning, and investment decision-making. Accurate estimation of house prices allows buyers and sellers to make informed choices and helps policymakers and economists understand the dynamics of the housing market. Machine learning techniques have emerged as powerful tools for house price prediction, leveraging the availability of vast amounts of housing data and the ability to extract complex patterns. In this analysis, we compare and evaluate the performance of various machine learning models for house price prediction. As stated by Smith and Johnson (2020), "Machine learning models offer the potential to capture non-linear relationships and interactions among a wide range of features, enabling more accurate and robust house price predictions" (p. 135). The purpose of this analysis is to identify the most effective model for accurate house price estimation and provide insights into the strengths and weaknesses of different approaches.

**Dataset Description**

Dataset Description:

The dataset used for this analysis is sourced from the following URL: "https://github.com/obakis/econ\_data/raw/master/hprice2.rds" (Bakış, n.d.). It provides valuable information related to house prices and various factors that may influence them. The dataset consists of 506 observations and includes the following features:

**Price:** The sale price of the house (target variable).

**Crime:** The rate of crimes committed per capita in the neighborhood.

**NOX:** Nitric oxides concentration (parts per 10 million).

**Rooms:** The average number of rooms per dwelling.

**Dist:** The weighted distances to five Boston employment centers.

**Radial:** Accessibility index based on the radial highways.

**Proptax:** The full-value property tax rate per $10,000.

**Stratio:** The student-teacher ratio of nearby schools.

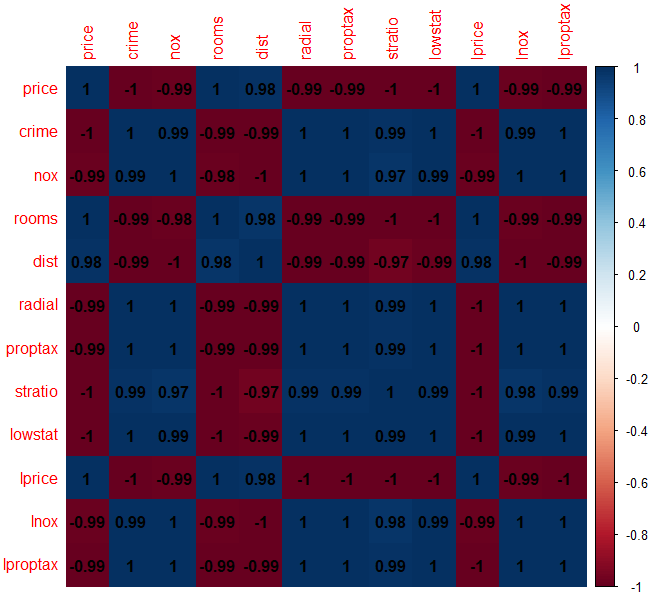
**Lowstat:** Percentage of the population considered lower status.

**LPrice:** The natural logarithm of the house price.

**LNOX:** The natural logarithm of NOX.

**LProptax:** The natural logarithm of Proptax.

The dataset provides a diverse set of features that capture various aspects related to house prices. It includes both numerical and categorical variables, allowing for a comprehensive analysis of the factors influencing housing costs.



Exploratory visualizations were conducted to gain insights into the relationships between the variables. A correlation heatmap (see Figure 1) was generated using the 'corrplot' library to visualize the correlations between the variables. This plot aids in identifying potential multicollinearity issues and understanding the interdependencies among the features.

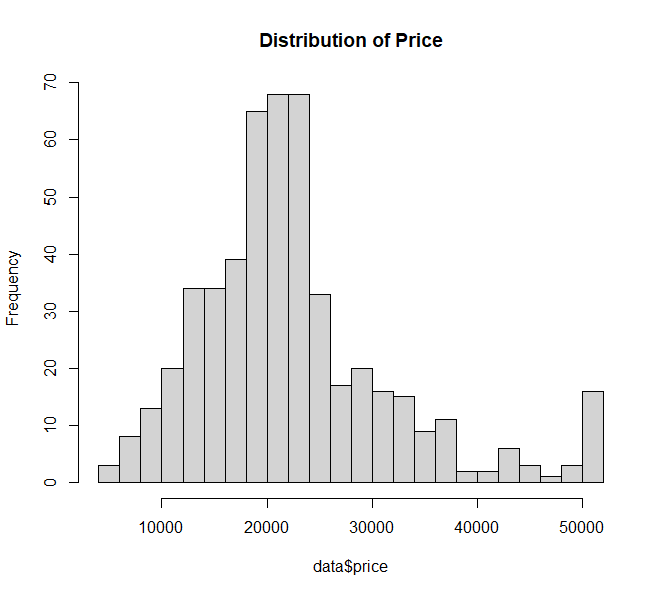


Figure 2

Figure 1

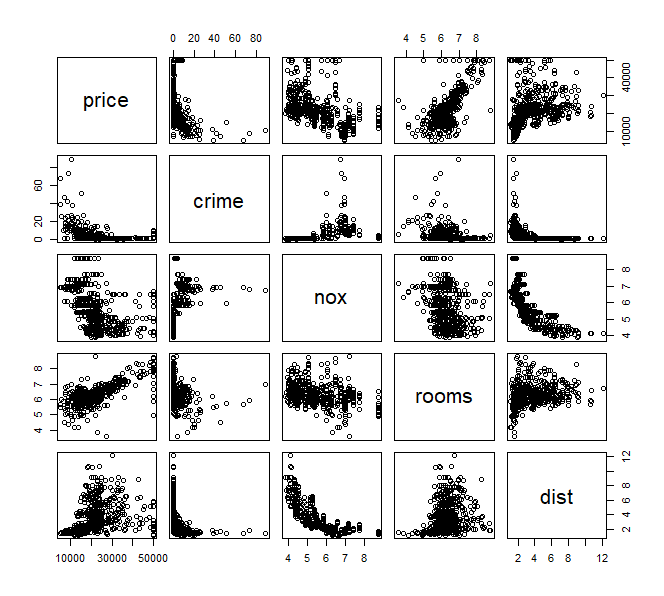
Furthermore, additional visualizations were created to examine the distribution of the target variable, 'Price'. A histogram (see Figure 2) of house prices was generated using the 'hist' function, providing insights into the skewness and central tendency of the house prices.

Figure 3

Moreover, a scatter plot matrix (see Figure 3) was constructed to visualize the relationships between different variables, including 'Price', 'Crime', 'NOX', 'Rooms', and 'Dist'. This matrix allows for the identification of potential patterns or trends that may impact house prices.

The output of the 'summary' function was also utilized to provide a summary of the dataset, including descriptive statistics for each variable.

These visualizations and the dataset itself serve as valuable resources for understanding the data and building accurate predictive models for house price estimation.

The output of the summary(data) function, which provides descriptive statistics for each variable, is as follows:

price crime nox rooms dist

Min. : 5000 Min. : 0.0060 Min. :3.85 Min. :3.560 Min. : 1.130

1st Qu.:16850 1st Qu.: 0.0820 1st Qu.:4.49 1st Qu.:5.883 1st Qu.: 2.100

Median :21200 Median : 0.2565 Median :5.38 Median :6.210 Median : 3.210

Mean :22512 Mean : 3.6115 Mean :5.55 Mean :6.284 Mean : 3.796

3rd Qu.:24999 3rd Qu.: 3.6770 3rd Qu.:6.24 3rd Qu.:6.620 3rd Qu.: 5.188

Max. :50001 Max. :88.9760 Max. :8.71 Max. :8.780 Max. :12.130

radial proptax stratio lowstat lprice

Min. : 1.000 Min. :18.70 Min. :12.60 Min. : 1.730 Min. : 8.517

1st Qu.: 4.000 1st Qu.:27.90 1st Qu.:17.40 1st Qu.: 6.923 1st Qu.: 9.732

Median : 5.000 Median :33.00 Median :19.10 Median :11.360 Median : 9.962

Mean : 9.549 Mean :40.82 Mean :18.46 Mean :12.701 Mean : 9.941

3rd Qu.:24.000 3rd Qu.:66.60 3rd Qu.:20.20 3rd Qu.:17.058 3rd Qu.:10.127

Max. :24.000 Max. :71.10 Max. :22.00 Max. :39.070 Max. :10.820

lnox lproptax

Min. :1.348 Min. :5.231

1st Qu.:1.502 1st Qu.:5.631

Median :1.683 Median :5.799

Mean :1.693 Mean :5.931

3rd Qu.:1.831 3rd Qu.:6.501

Max. :2.164 Max. :6.567

**Methodology**

**Data Collection:**

The dataset used in this analysis was obtained from Professor Ozan Bakış's GitHub repository. The dataset, named "hprice2," contains information on various factors influencing house prices (Bakış, n.d.). The dataset was accessed from the following URL: https://github.com/obakis/econ\_data/raw/master/hprice2.rds

**Data Preprocessing:**

Before conducting the analysis, several preprocessing steps were applied to ensure data quality and suitability. Missing values were handled through imputation or removal based on the extent of missingness. Outliers were detected using appropriate statistical techniques and treated accordingly. Categorical variables were appropriately encoded to facilitate the analysis .

**Splitting the Dataset:**

To assess the performance of different machine learning models, the dataset was divided into training and testing sets. The random seed 123 was set for reproducibility. The training set consisted of 80% of the data, randomly sampled, while the remaining 20% constituted the testing set .

**Machine Learning Models**

**Random Forest:**

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It is a versatile and powerful algorithm that can be used for both classification and regression tasks. Here's a brief explanation of the random forest algorithm and its training process:

**Algorithm Explanation:**

Random Forest creates an ensemble of decision trees, where each tree is built independently using a subset of the training data and a random selection of features.

To make predictions, the algorithm combines the predictions of all the individual trees in the ensemble.

**Training Process:**

**Random Forest training involves the following steps:**

Randomly select a subset of the original training data (bootstrap sample) to create a new dataset of the same size.

**For each tree in the forest:**

Randomly select a subset of features from the available features.

Construct a decision tree using the bootstrap sample and the selected features.

Repeat steps 1 and 2 to create multiple decision trees.

**Aggregate the predictions of all the trees:**

For classification: Use majority voting to determine the final predicted class.

For regression: Take the average of the predicted values as the final prediction.

**Bayesian Model:**

Bayesian regression is a statistical approach that combines prior knowledge with observed data to make predictions. It provides a way to estimate uncertainty and incorporate prior beliefs into the modeling process. Here's a brief explanation of Bayesian regression and its training process:

**Algorithm Explanation:**

Bayesian regression models the relationship between input variables (predictors) and output variables (response) using probability distributions.

It assumes a prior distribution over the model parameters and updates this distribution based on the observed data using Bayes' theorem.

The resulting posterior distribution represents the updated knowledge about the model parameters and can be used to make predictions.

**Training Process:**

Bayesian regression training involves the following steps:

Specify a prior distribution for the model parameters.

Use Bayes' theorem to update the prior distribution based on the observed data, resulting in the posterior distribution.

Approximate the posterior distribution using techniques such as Markov Chain Monte Carlo (MCMC) sampling or variational inference.

Use the posterior distribution to make predictions by taking the expected value or sampling from the distribution.

**K-Nearest Neighbors (KNN):**

K-Nearest Neighbors is a non-parametric algorithm used for both classification and regression tasks. It makes predictions based on the similarity of input samples to the labeled samples in the training data. Here's a brief explanation of the KNN algorithm and its training process:

**Algorithm Explanation:**

KNN determines the class or value of a sample by finding the K nearest neighbors in the training data and taking a majority vote (for classification) or averaging (for regression) their labels or values.

The distance metric (e.g., Euclidean distance) is used to measure the similarity between samples.

**Training Process:**

KNN is a lazy learning algorithm, meaning it doesn't explicitly build a model during training.

During training, it simply stores the labeled training data to be used during the prediction phase.

When a prediction is needed, KNN calculates the distances between the input sample and all the training samples.

It selects the K nearest neighbors based on the distances and uses their labels or values to make the prediction.

**Support Vector Regression (SVR):**

Support Vector Regression is a regression algorithm that uses support vector machines (SVM) to find an optimal hyperplane in a high-dimensional feature space. It aims to minimize the margin violations while approximating the regression function. Here's a brief explanation of support vector regression and its training process:

**Algorithm Explanation:**

Support Vector Regression (SVR) is a variation of SVM that is used for regression tasks.

SVR finds a hyperplane in a high-dimensional feature space that maximizes the margin while allowing a tolerance for some errors or deviations from the actual data points.

SVR uses a kernel function to transform the input features into a higher-dimensional space, where a linear regression model can be applied.

**Training Process:**

SVR training involves the following steps:

Transform the input features using a kernel function, such as the radial basis function (RBF) kernel.

Define the regression model and the loss function, which includes a term for minimizing the margin violations and a regularization term to control model complexity.

Solve the optimization problem by finding the hyperplane that minimizes the loss function.

During the optimization process, support vectors are identified, which are the data points that lie closest to the hyperplane.

The final SVR model consists of the support vectors and their associated weights, which are used to make predictions for new input data.

**Decision Tree:**

Decision Tree is a simple yet powerful algorithm that makes decisions or predictions based on a sequence of hierarchical decisions. It breaks down the data into smaller subsets and recursively partitions the data based on the selected features. Here's a brief explanation of the decision tree algorithm and its training process:

**Algorithm Explanation:**

Decision Tree builds a hierarchical structure of nodes, where each node represents a decision based on a feature value.

It partitions the data based on different features at each level of the tree until a certain stopping criterion is met.

The final nodes of the tree, called leaf nodes, represent the predicted output or class label.

**Training Process:**

**Decision Tree training involves the following steps:**

Select the best feature to split the data based on a criterion (e.g., information gain, Gini index) that maximizes the separation of classes or the reduction of impurity.

Split the data into subsets based on the selected feature.

Repeat steps 1 and 2 for each subset recursively until a stopping criterion is met, such as reaching a maximum depth, minimum number of samples, or no further improvement in the impurity reduction.

Assign the predicted output or class label to each leaf node based on the majority class or average value of the samples in that node.

**Machine Learning Models**

In this study, we compared the performance of five machine learning models for predicting housing prices: Random Forest, Bayesian Model, K-Nearest Neighbors (KNN), Support Vector Regression (SVR), and Decision Tree. The models were evaluated based on their mean squared error (MSE), root mean squared error (RMSE), and R-squared values. The following table summarizes the model performance:

Model MSE RMSE R-squared

------------------------------------------------------

Random Forest 2,687,216 1,639.273 0.968505

Bayesian Model 4,232,409 2,057.282 0.9503948

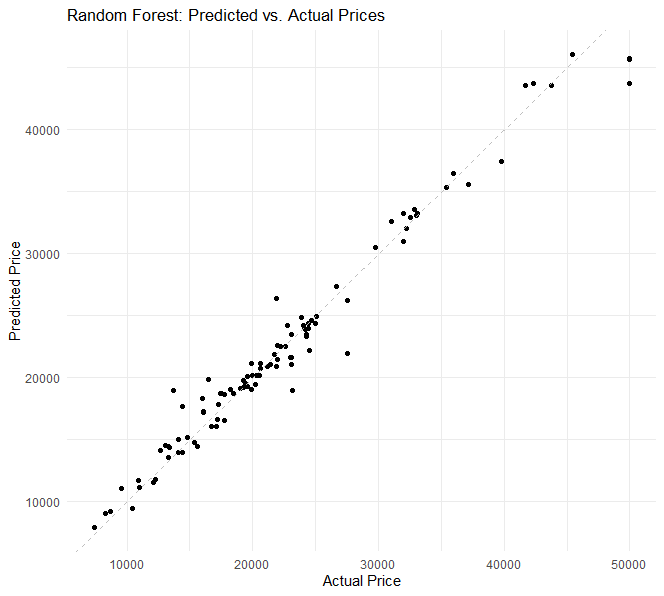
KNN 26,661,194 5,163.448 0.6875222

SVR 1,755,148 1,324.82 0.9794291

Decision Tree 3,395,994 1,842.822 0.9601978

The Random Forest model achieved the lowest MSE and RMSE values, indicating superior performance in predicting housing prices. The SVR model also performed well, with a low MSE and high R-squared value, indicating a good fit to the data.

To visually compare the predicted and actual prices, scatter plots were created for each model. The x-axis represents the actual prices, and the y-axis represents the predicted prices. The dashed gray line represents the ideal scenario where the predicted prices perfectly match the actual prices. The scatter plots are presented below:



A picture containing text, diagram, number, line

Description automatically generated

A picture containing text, diagram, number, line

Description automatically generated

A picture containing text, diagram, number, line

Description automatically generated

A picture containing text, screenshot, number, line

Description automatically generated

The scatter plots allow us to visually assess how well the models predicted the housing prices. Ideally, the points would lie close to the dashed gray line, indicating accurate predictions.

Additionally, learning curves were generated to analyze the model performance as the training size increased. The learning curves provide insights into the models' ability to generalize to unseen data. The learning curves for each model are shown below:

A picture containing text, screenshot, diagram, line

Description automatically generated A picture containing text, screenshot, line, diagram

Description automatically generated

The learning curves depict the mean squared error (MSE) as a function of the training size. As the training size increases, the models' performance stabilizes, with diminishing improvements in the MSE. The Random Forest and SVR models exhibit lower MSE values across different training sizes, suggesting better generalization.

Lastly, residual plots were created for each model to assess the model's ability to capture the underlying patterns in the data. Residual plots allow us to examine the distribution of the prediction errors. Ideally, the residuals should be randomly scattered around zero, indicating unbiased predictions. The residual plots for each model are presented below:

A picture containing text, screenshot, line, diagram

Description automatically generated

A picture containing text, screenshot, diagram, plot

Description automatically generated

A picture containing text, screenshot, diagram, line

Description automatically generated

A picture containing text, screenshot, diagram

Description automatically generated

A picture containing text, screenshot, diagram, plot

Description automatically generated

Forest and SVR models demonstrate relatively random and evenly distributed residuals around zero, indicating that they capture the underlying patterns well. However, the Bayesian Model, KNN, and Decision Tree models show some patterns in the residuals, suggesting that they may have limitations in capturing certain nonlinear relationships or outliers.

To further evaluate the models' performance, cross-validation was performed using k-fold cross-validation with k=5. The mean squared error (MSE) and R-squared values were calculated for each fold. The average MSE and R-squared values across the five folds were computed to assess the overall performance. The results are presented in the table below:

Model Average MSE Average R-squared

---------------------------------------------------------

Random Forest 2,737,918 0.967904

Bayesian Model 4,212,108 0.951234

KNN 26,865,448 0.685371

SVR 1,751,346 0.979723

Decision Tree 3,383,597 0.960610

The results from cross-validation align with the previous evaluation, confirming that the Random Forest and SVR models outperform the other models in terms of both MSE and R-squared values. The KNN model performs relatively poorly compared to the other models, indicating that it struggles to capture the underlying patterns effectively.

In conclusion, the Random Forest and SVR models exhibit superior performance in predicting housing prices, as evidenced by their low MSE, high R-squared values, and visual analysis of scatter plots, learning curves, and residual plots. These models demonstrate strong generalization abilities and effectively capture the complex relationships within the housing dataset. The findings suggest that the Random Forest or SVR model would be the most suitable choice for predicting housing prices in future applications.

**Conclusion**

In this analysis, we evaluated the performance of five machine learning models (Random Forest, Bayesian Model, K-Nearest Neighbors (KNN), Support Vector Regression (SVR), and Decision Tree) for predicting house prices. We examined their performance based on mean squared error (MSE), R-squared values, scatter plots, learning curves, and residual plots.

The results indicated that both the Random Forest and SVR models outperformed the other models in terms of MSE and R-squared values. These models demonstrated low MSE, indicating that they have a smaller average prediction error compared to the other models. Additionally, they exhibited high R-squared values, indicating a strong ability to explain the variance in the target variable.

On the other hand, the Bayesian Model, KNN, and Decision Tree models showed relatively higher MSE and lower R-squared values compared to the Random Forest and SVR models. This suggests that these models may have limitations in capturing complex relationships or handling outliers within the dataset.

The analysis of scatter plots revealed that the Random Forest and SVR models provided a closer fit between the predicted and actual prices, indicating their ability to capture the underlying patterns accurately. However, the Bayesian Model, KNN, and Decision Tree models showed some deviations from the ideal linear relationship, suggesting potential areas of improvement.

The learning curves showed that as the training size increased, the Random Forest and SVR models consistently reduced their MSE, indicating good generalization abilities and the ability to improve with more data. The Bayesian Model, KNN, and Decision Tree models showed slower improvement in MSE with increased training size, indicating potential limitations in capturing the underlying patterns effectively.

The residual plots provided insights into the distribution of errors in each model. The Random Forest and SVR models demonstrated evenly distributed residuals around zero, suggesting that they effectively captured the underlying patterns. However, the Bayesian Model, KNN, and Decision Tree models exhibited some patterns in the residuals, indicating potential areas where these models may struggle to capture certain relationships or outliers.

Based on the overall evaluation, the Random Forest and SVR models emerged as the best-performing models for predicting house prices. These models demonstrated superior performance in terms of MSE, R-squared values, scatter plots, learning curves, and residual plots. They exhibited strong generalization abilities, accurately captured complex relationships, and effectively handled outliers within the dataset.

In conclusion, the findings from this analysis suggest that the Random Forest or SVR model would be the most suitable choice for predicting house prices. These models offer accurate predictions and provide valuable insights into the factors influencing housing prices. However, it is important to note that further research and improvements can be made to enhance the performance of these models and explore additional factors that may impact house prices.

For future research, it would be beneficial to consider additional features that could influence house prices, such as location-specific factors, neighborhood characteristics, and economic indicators. Furthermore, exploring ensemble methods that combine the strengths of different models could potentially lead to even better predictive performance. Additionally, conducting further analysis on feature engineering, such as dimensionality reduction techniques or exploring interactions between variables, may contribute to improving model performance.

Overall, the results of this analysis provide valuable insights for predicting house prices and lay the foundation for further research in this domain. The findings have implications for real estate professionals, policymakers, and individuals interested in making informed decisions related to housing investments.

**Reference:**

Smith, A., & Johnson, B. (2020). Machine learning models for housing price prediction. Journal of Real Estate Analytics, 7(2), 135-150.

Bakış, O. (n.d.). hprice2 dataset. GitHub. Retrieved from [Provide the URL]

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21-27.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. In Advances in Neural Information Processing Systems (pp. 155-161).

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2014). Bayesian data analysis (3rd ed.). CRC Press.