SEMESTER PROJECT

Machine Learning

Classification and Regression Analysis of Houses Data

Muhammad Aqeel Khan
NIM-BSMath-2020-13

Table of Contents

Machine Learning Semester Project Report

We have given data related to houses and we need to implement regression as well as classification Algorithms/prediction model. We have performed all given below steps in this project.

Data preprocessing:

I provided two files—one comprising the training dataset and the other containing testing data. I uploaded both CSV files and analyzed them to gain a comprehensive understanding of the dataset, enabling me to enhance my subsequent work. Within both CSV files, the columns "lot_size_units" and "size_units" represented units for the "size" and "lot_size" columns. I converted all acre values in the "size" and "lot_size" columns to square feet and subsequently removed the "lot_size_units" and "size_unit" columns from both the training and testing datasets, as they were no longer required for further processing. Identifying null entries in the "lot_size" column of both datasets, I addressed this issue by calculating the mean of the column and filling the null values with this mean, given that the "lot_size" column comprises multiple numerical values.

After completing the aforementioned steps with both dataset files, I merged them into a single dataset file named the merged dataset. Following the merger, I created visual representations of the merged dataset, allowing me to examine its overall structure and identify any outliers. To eliminate outliers, I utilized both interquartile range and Z-score normalization techniques. Upon observation, I noticed that the "size" and "price" features exhibited a nearly normalized Z-score curve. Consequently, I applied Z-score normalization specifically to address outliers in these features. For all other features in the merged dataset, excluding "size" and "price," I employed interquartile range normalization to handle outliers. Additionally, I applied a log transformation to the entire dataset, reducing the values to enhance the performance of algorithms/models during both the training and testing phases.

Regression model development and evaluation:

Linear Regression model development

Firstly, I imported the "train_test_split" function and the linear regression model from the Python SKLEARN library. To assess the model's performance, I brought in two metrics, namely "r2_score" and "mean_squared_error," from the SKLEARN library. For the training of the linear regression model, I utilized all features from the merged dataset, excluding the "price" column. Subsequently, I designated the "price" column of the merged dataset for testing the model. The dataset was then split into two segments, with 20 percent allocated for testing the model and 80 percent for training the model. Following these steps, I applied the linear regression model to the training dataset and made predictions on the testing dataset.

Evaluation of linear regression model

For the evaluation of the linear regression model on the merged dataset, I checked the r2_score and mean squared error between actual “price” values of houses and predicted “price” values of houses. I found r2_score= 0.638853727393857 and mean squared error= 0.08004789055135317

As we know the model is considered perfect whenever r2_score is approximately equal to 1 and the mean squared error is approximately equal to zero.

Random Forest Regressor:

To construct a Random Forest Regressor, I imported the Random Forest Regressor and train_test_split from the SKLEARN library. For assessing the Random Forest Regressor, I incorporated the metrics r2_score and mean_squared_error from the SKLEARN library. The training of the Random Forest regression model involved using all features from the merged dataset, excluding the "price" column. Subsequently, the "price" column of the merged dataset was assigned for testing the model. The dataset underwent a split, with 20 percent designated for testing the model and 80 percent for training the model. Fine-tuning the model involved adjusting the values of n_estimators to optimize performance. Following these steps, I applied the Random Forest regression model to the training dataset and made predictions on the testing dataset.

Evaluation of Random Forest Regression model:

For the evaluation of the model on the merged dataset, I checked the r2_score and mean squared error between actual “price” values of houses and predicted “price” values of houses. I found r2_score= 0.7463495039507891 and mean squared error= 0.05622150548452957

As we know that model is considered perfect whenever r2_score is approximately equal to 1 and the mean squared error is approximately equal to zero.

K Nearest Neighbor Regressor model development:

To initiate the process, I imported the "train_test_split" function and the K Nearest Neighbors Regressor model from the Python SKLEARN library. To assess the model, I incorporated two metrics, namely "r2_score" and "mean_squared_error," from the SKLEARN library. All features from the merged dataset, excluding "price," were utilized for training the model, while the "price" column of the merged dataset was reserved for testing. The dataset underwent a split, with 30 percent allocated for testing the model and 70 percent for training. The model was fine-tuned using grid search cv from the SKLEARN library, revealing that k=8 would likely yield the best results for my model. Following these steps, I applied the K Nearest Neighbors Regressor model to the training dataset and made predictions on the testing dataset.

Evaluation of the KNN Regression model:

For the evaluation of the model on the merged dataset, I checked the r2_score and mean squared error between actual “price” values of houses and predicted “price” values of houses. I found r2_score= 0.6205088140713818 and mean squared error= 0.08630766940402376.

After the implementation of three different models, I observed that the random forest model is performing better than KNN and linear regression models on my merged dataset. For further analysis, I plotted a Distribution graph between actual price values and predicted price values of houses and analyzed the graphical differences b/w them.

Classification model development and evaluation:

Preprocessing for classification:

For preprocessing, I divided “price” column values into three different class categories namely “0”,”1” and “2”. For dividing values into categories, I used the “bin” command and with the help of the Pandas library, I used the cut function to divide “price” values into three different class categories. I plotted these classes to visualize the distribution of data across the created classes.

Random Forest classifier:

In developing the classification model, we used a RandomForestClassifier from the sci-kit-learn library to predict the 'class' labels based on features such as the number of beds, baths, property size, lot size, and zip code. The dataset was split into training and testing sets using the train_test_split function, with 70% for training and 30% for testing. The RandomForestClassifier was then trained on the training set, and predictions were made on the test set.

Evaluation of Random Forest classifier model:

For evaluating the model performance, we calculated the accuracy = 0.819775429326288

which measures the proportion of correctly predicted instances. Additionally, we utilized a confusion matrix and classification report metrics to assess the model's class-wise performance, providing insights into the precision, recall, and F1 score for each class. In our specific case, we focused on simplicity and key metrics to gauge how well the model is performing in classifying the price categories.

Decision tree classifier:

In the process of developing a classification model using a Decision Tree Classifier, we focused on predicting 'class' labels based on features like the number of beds, baths, property size, lot size, and zip

code. The dataset was divided into training and testing sets, with 80% of the data used for training and 20% for testing, employing the train_test_split function. The Decision Tree Classifier was then trained on the training set and used to predict labels for the test set.

Evaluation of Decision tree classifier model:

To assess the model's performance, we computed accuracy= 0.7702970297029703, which represents the proportion of correctly predicted instances. Additionally, we examined the confusion matrix, providing a detailed breakdown of correct and incorrect predictions across different classes. The classification report further delves into precision, recall, and F1-score metrics for each class, offering a comprehensive understanding of the model's effectiveness in categorizing the price classes. This approach allows for a straightforward evaluation of the Decision Tree model's performance in our classification task.

K Nearest Neighbor classifier model development:

In constructing a classification model using the K-Nearest Neighbors (KNN) algorithm, we aimed to predict 'class' labels based on features like the number of beds, baths, property size, lot size, and zip code. The dataset was divided into training and testing sets, allocating 80% of the data for training and 20% for testing using the train_test_split function. The KNN model, configured with a choice of 8 neighbors, was then trained on the training set and utilized to predict labels for the test set.

Evaluation of the KNN Classifier model:

Subsequently, we evaluated the model's performance using key metrics. The accuracy score was calculated to gauge the proportion of correctly predicted instances, and the confusion matrix provided insights into the distribution of correct and incorrect predictions among different classes. The classification report offered a detailed examination of precision, recall, and F1-score metrics for each class, offering a comprehensive overview of the KNN model's effectiveness in classifying the price categories. This straightforward approach enables a clear understanding of the KNN model's performance in our classification task.

Naive Bayes Classifier:

In developing a classification model using the Gaussian Naive Bayes algorithm, our goal was to predict 'class' labels based on features such as the number of beds, baths, property size, lot size, and zip code. The dataset was split into training and testing sets, with 80% of the data assigned for training and 20% for testing, using the train_test_split function. The Gaussian Naive Bayes classifier was then trained on the training set and used to predict labels for the test set.

Evaluation of Naive Bayes Classifier model:

To assess the model's performance, we calculated the accuracy= 0.695049504950495, which represents the proportion of correctly predicted instances. Additionally, we examined the confusion matrix, offering insights into the distribution of correct and incorrect predictions among different classes. The classification report provided a detailed overview of precision, recall, and F1-score metrics for each class, offering a comprehensive assessment of the Gaussian Naive Bayes model's effectiveness in classifying the price categories. This straightforward approach allows for a clear understanding of the model's performance in our classification task.

Analysis and interpretation of results:

Analysis of regression algorithm

I applied three regression models to my dataset which are linear regression, random forest regressor, and kNN regressor. I found that the random forest regressor is doing a better job of prediction on my data set.

Linear Regression:

- R-squared (Coefficient of Determination): 0.639
- Mean Squared Error (MSE): 0.080

Interpretation:

- The R-squared score of 0.641 indicates that the linear regression model explains about 64.1% of the variance in the target variable.
- The Mean Squared Error of 0.080 suggests that, on average, the squared difference between the predicted and actual values is 0.080.

Random Forest Regressor:

- R-squared: 0.747
- Mean Squared Error: 0.056

Interpretation:

- The R-squared score of 0.747 is higher than the linear regression model, indicating that the random forest model explains about 74.7% of the variance in the target variable.

- The Mean Squared Error of 0.056 is lower than that of linear regression, suggesting that the random forest model provides more accurate predictions with less error.

KNN Regressor:

- R-squared: 0.621
- Mean Squared Error: 0.086

Interpretation:

- The R-squared score of 0.621 suggests that the KNN regressor explains about 62.1% of the variance in the target variable.
- The Mean Squared Error of 0.086 is higher than both linear regression and random forest, indicating that the KNN model has a higher average squared difference between predicted and actual values.

Overall Comparison:

- The random forest regressor outperforms both linear regression and KNN regressor in terms of R-squared score, indicating better explanatory power.
- The random forest model also has the lowest Mean Squared Error among the three, suggesting better accuracy in predictions.
- Linear regression performs better than KNN in terms of both R-squared score and Mean Squared Error.

Analysis of the classification algorithm:

We conducted a comprehensive analysis of four classification models applied to our dataset: Random Forest Classifier, Decision Tree Classifier, k-nearest Neighbors (KNN) Classifier, and Naive Bayes Classifier. Here's a simplified report for a clear understanding:

Random Forest Classifier:

- **Accuracy:** 81.98%
- **Key Observations:**
  - Precision, Recall, and F1-score are highest for class 2.
  - Class 0 shows lower precision and recall compared to the other classes.

- The model performs well overall with a balanced accuracy across classes.

Decision Tree Classifier:

- **Accuracy:** 77.03%
- **Key Observations:**
  - Precision and recall are high for class 2.
  - Class 0 exhibits lower precision and recall compared to other classes.
  - The model provides a reasonable overall accuracy with room for improvement in class 0 predictions.

k-Nearest Neighbors (KNN) Classifier:

- **Accuracy:** 71.29%
- **Key Observations:**
  - Precision, recall, and F1-score are lower for class 0 compared to the other classes.
  - The model shows balanced performance but struggles with predictions for class 0.

Naive Bayes Classifier:

- **Accuracy:** 69.50%
- **Key Observations:**
  - Precision, recall, and F1-score are relatively balanced across classes.
  - Class 0 has lower precision, while class 1 has lower recall.
  - The model provides moderate accuracy with room for improvement in precision for class 0 and recall for class 1.

General Insights:

- Random Forest Classifier stands out with the highest accuracy, providing a good balance across different classes.
- Decision Tree and KNN Classifiers exhibit reasonable performance, but there is room for improvement, especially for class 0 in both models.

- Naive Bayes Classifier performs moderately well but shows potential for improvement in precision for class 0 and recall for class 1.

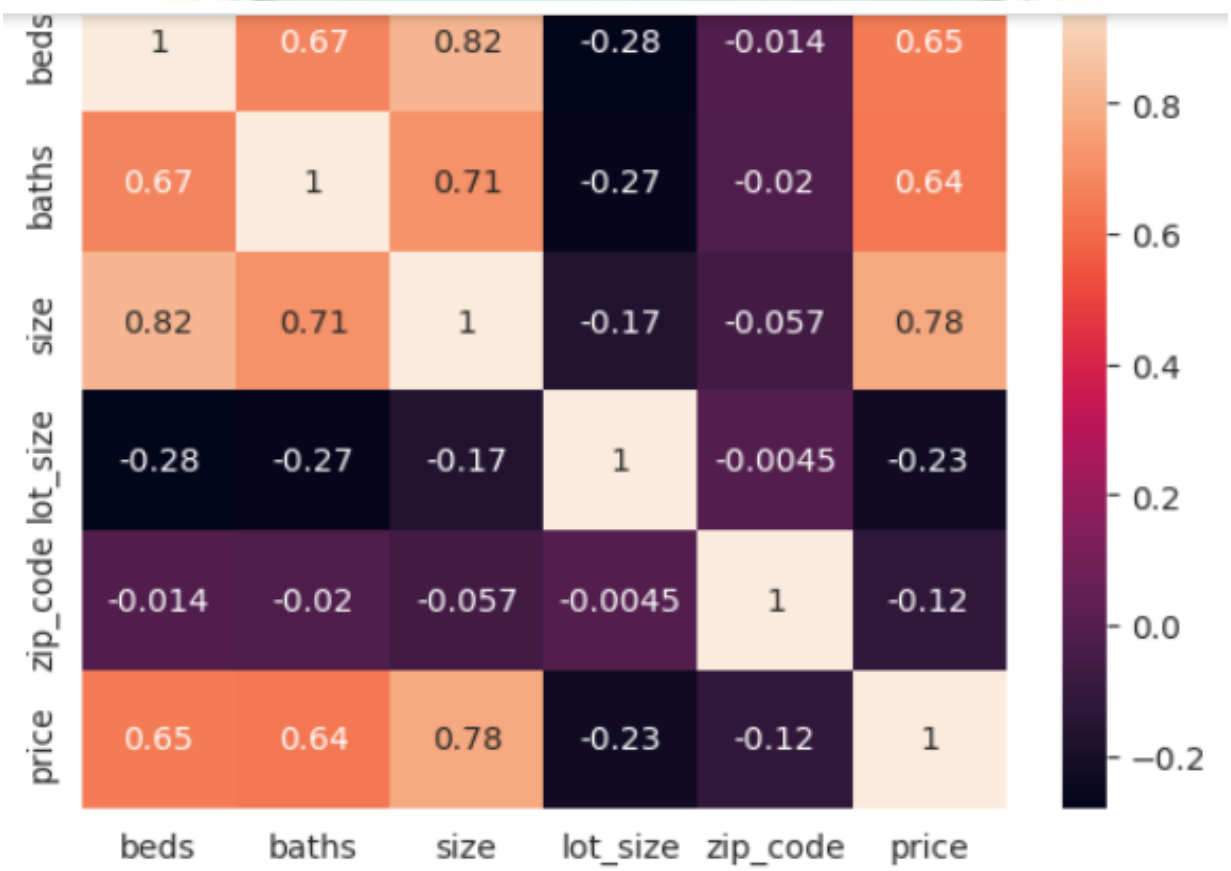
Visualization of key findings

Preprocessing analysis

Analysis b/w features and labels of merged_dataset

For doing analysis, I find correlation b/w them and know how much these features have relationship with each other.

Correlation matrix:



Strongest positive correlations:

- The features that move most closely together are:
- Number of bedrooms and price (0.65)
- Number of bathrooms and price (0.64)
- Property size and price (0.78)

Negative correlations:
Lot size and zip code have mild negative correlations with price (-0.23 and -0.2, respectively), suggesting they might move slightly opposite to price.

Weak or no correlations:

- Zip code and lot size have very weak correlations with other features, indicating they don't have strong linear relationships with them.
- Price is strongly linked to property size and number of rooms: This suggests that larger homes with more bedrooms and bathrooms tend to have higher prices.
- Lot size and zip code have weaker relationships with price: This means that these factors might play a smaller role in determining price, at least in a simple linear sense

covariance matrix:

Positive Relationships:

- Beds, baths, and size have moderate positive relationships, suggesting they tend to increase together. This aligns with the expectation that larger homes often have more bedrooms and bathrooms.
- Size and lot size also have a moderate positive relationship, indicating that larger homes typically sit on larger lots.

Negative Relationships:

- Zip code has weak negative relationships with most other variables, suggesting some variation in home features across different zip codes.

Variance:

Most Variable Features:

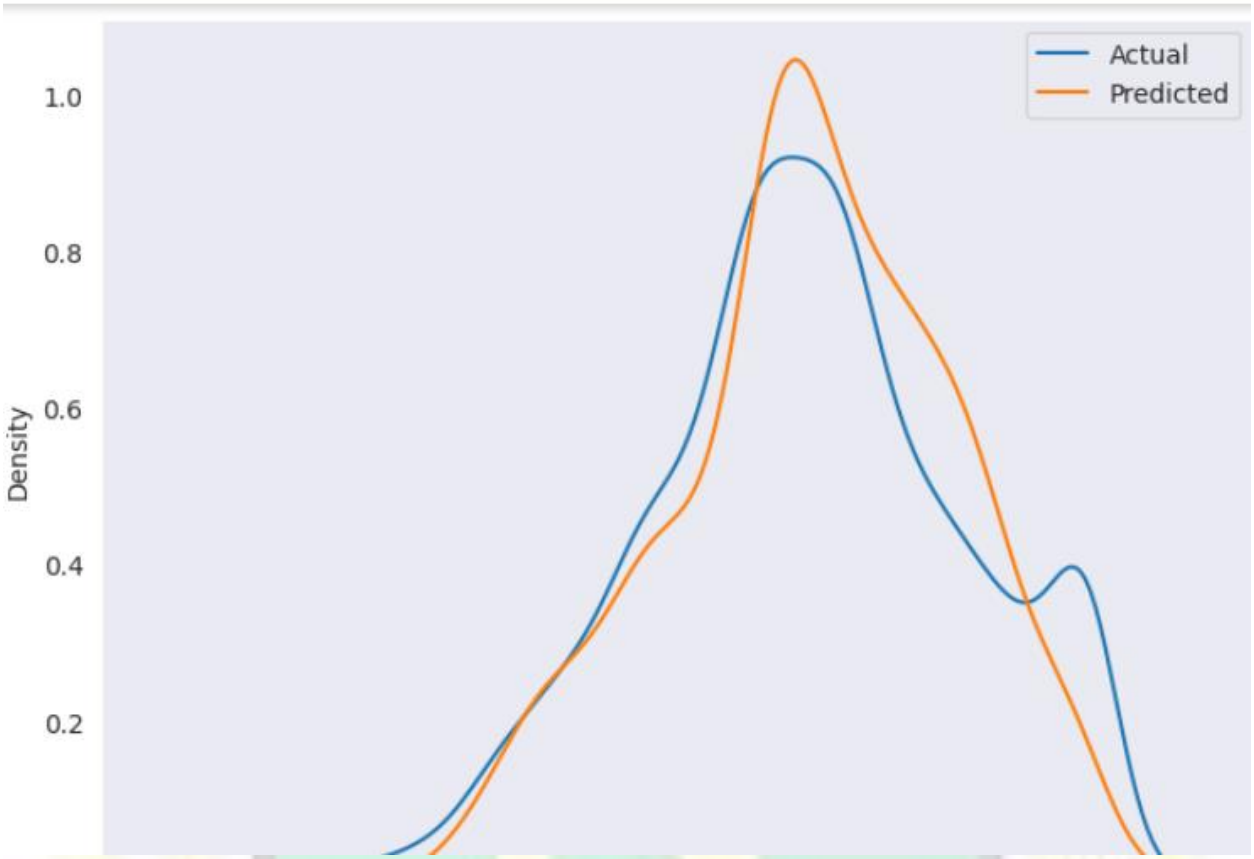
- Price has the highest variance, meaning it has the most spread-out values. This reflects the wide range of home prices in the dataset.
- Lot size also has relatively high variance, suggesting considerable diversity in lot sizes.

Overall Observations:

- Interconnected Features: The positive relationships among beds, baths, size, and lot size highlight how these features are often interconnected in housing data.
- Zip Code as a Factor: The weak negative relationships with zip code suggest that location may play a subtle role in influencing home features and prices.

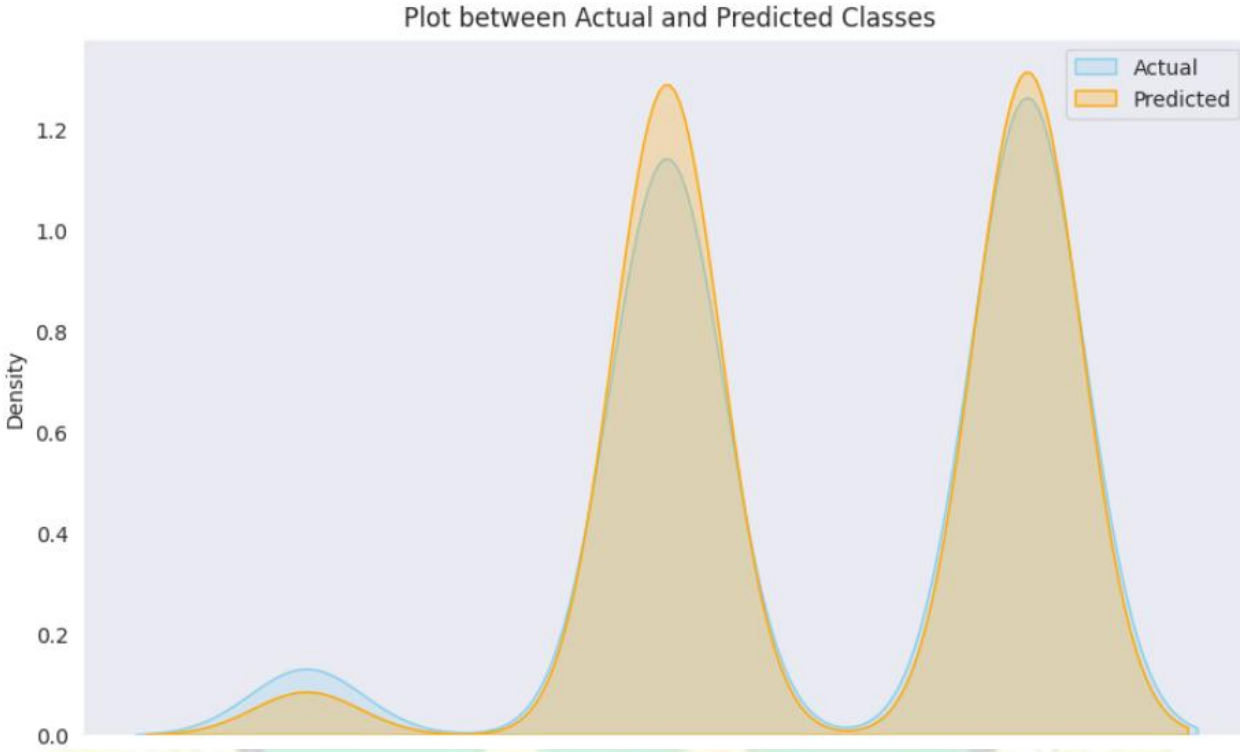
Regression analysis:

After the implementation of three different models, I observed that the random forest model is performing better than KNN and linear regression models on my merged dataset. For further analysis, I plotted a Distribution graph between actual price values and predicted price values of houses and analyzed the graphical differences b/w them.


Actual
Predicted
1.0
0.8
0.6
0.4
0.2
Density

Classification analysis

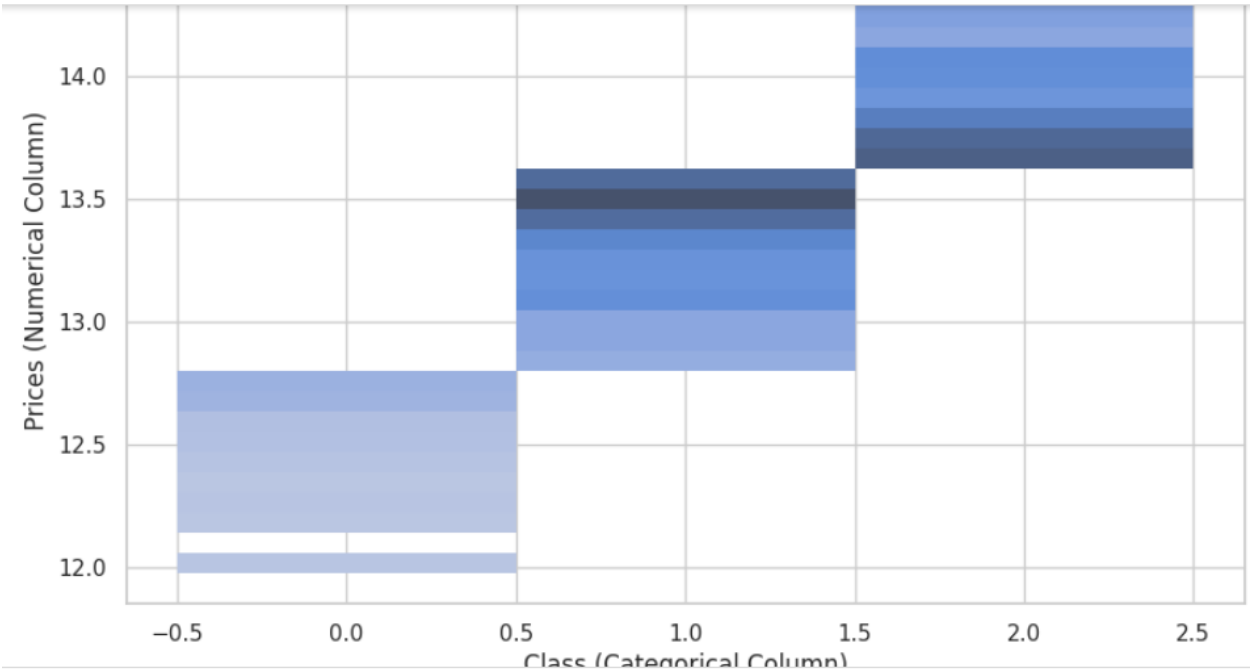
After the implementation of four different models, I observed that the random forest model is performing better than all on my merged dataset. For further analysis.



Combined analysis:

Relationship b/w price and class

I noticed that price and class columns are directly related to each other. Classes 1.5 and 2.5 tend to have higher prices. This indicates items in these classes may hold more value or have distinct characteristics that influence their price



Features contribution in classification of Houses:

I did analysis for random forest classifier model because it is performing better than other:

- House size is the most important feature for classifying houses, followed by the number of baths and bedrooms. This suggests these features strongly influence house categories.
- Lot size and zip code have lower importance, but still contribute to classification. This indicates they play a smaller but still relevant role in house categorization.
- Consider exploring how these features interact with each other to better understand their combined impact on house classification.