A logo with text and symbols on it

Description automatically generated

**Comparative Analysis for Regression and Classification Tasks: A Case Study on Boston Housing Data**

**Team 3: Kruayatidee Wannida, Shrestha Jarshana, Vorakamolpisit Methapat**

**ALY 6040: Data Mining**

**Prof. Kasun Samarasinghe**

**March 23, 2024**

**Introduction**

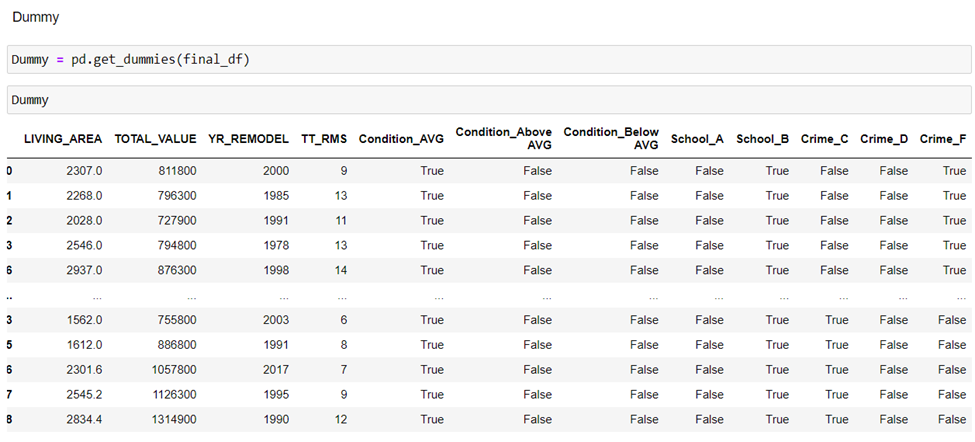
In the realm of real estate analytics, Boston stands out as a vibrant and dynamic market, driven by diverse factors ranging from property size and condition to neighborhood amenities and socioeconomic indicators. Harnessing the power of data analytics, our team delves into the intricate landscape of Boston's residential properties, aiming to forecast their total value with precision and insight.

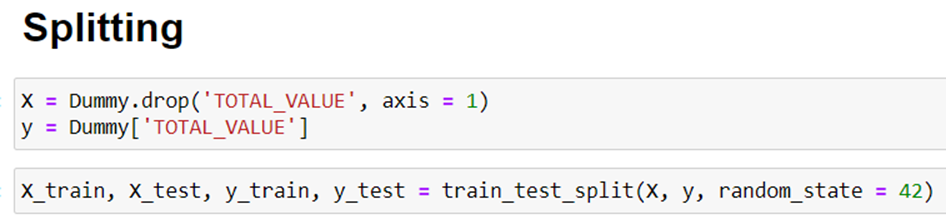
As we navigate through the intricacies of regression and classification analyses, we not only aim for predictive accuracy but also strive to unravel the nuanced relationships between various predictors and the target variable. With each model iteration and evaluation, we gain deeper insights into the factors driving property valuation dynamics, empowering stakeholders with valuable knowledge for strategic planning and investment decisions.

In this report, we present a synthesis of our findings, encapsulating the performance of different models, key metrics of evaluation, and actionable recommendations for stakeholders navigating Boston's real estate landscape. From discerning the impact of neighborhood characteristics to understanding the predictive power of structural attributes, our analysis offers a multifaceted perspective, illuminating the path towards harnessing data-driven insights in the pursuit of real estate excellence.

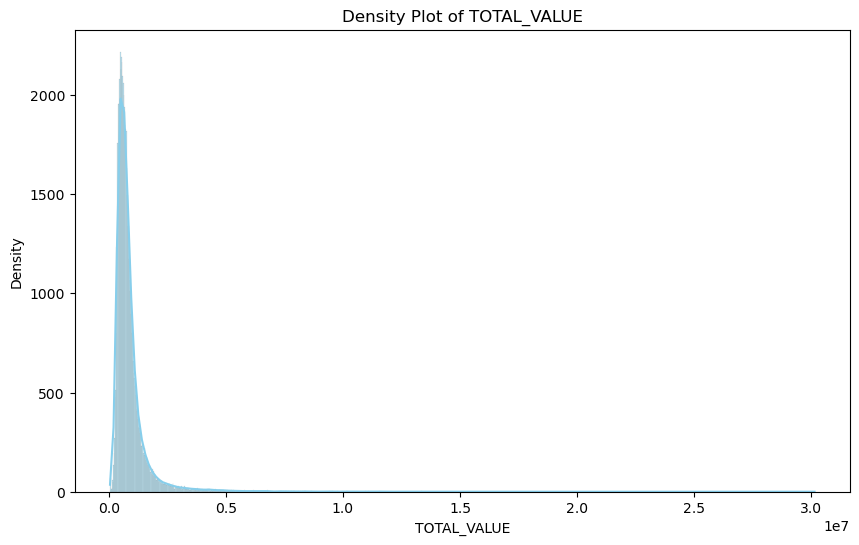
**Data Preparation**

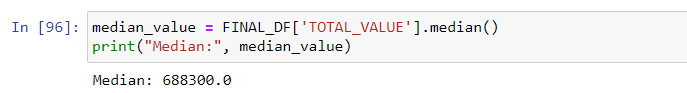
In this analysis, our objective is to forecast the total value of Boston property utilizing various indicators such as living area, remodel years, total rooms, condition, crime grade, and school grade. Originally, our target variable is numerical, we've opted for regression analysis as it aligns better with our goal compared to classifier analysis. However, we also performed classifier analysis by modifying our target variable to binary. To facilitate regression analysis, it's imperative to convert categorical variables into dummy variables. Consequently, we'll transform condition, crime, and school into dummy variables to effectively incorporate them into our regression model.

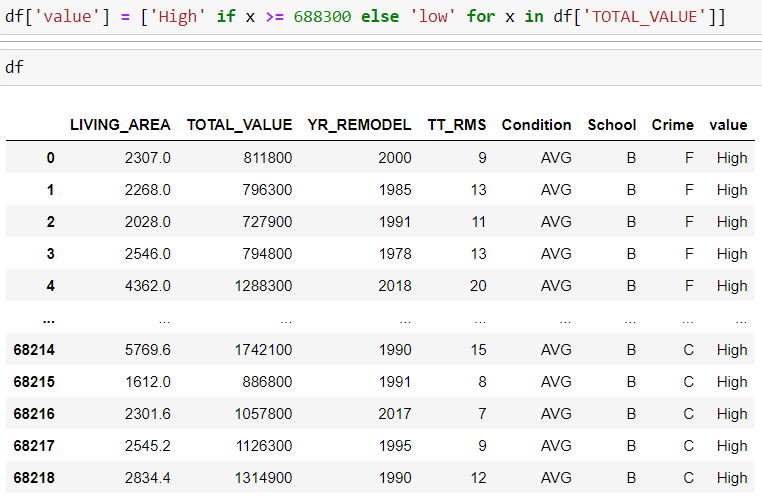
 Once we've obtained our dummy variables, the next step is to split our dataset into training and testing sets. This ensures that we have sufficient data to train our model while also retaining a portion for evaluation purposes. We'll perform this split to facilitate model training on one subset and then assess its performance on unseen data using the other subset.



For classifier analysis, we transformed our target variable from numerical values to categorical values, specifically "High" and "Low". Given that our target variable is "TOTAL\_VALUE" and our data displays a right-skewed distribution, utilizing the median as a measure of central tendency for partitioning appeared to be a fitting approach. Consequently, we defined "High" as instances where the "TOTAL\_VALUE" is greater than or equal to 688,300, while "Low" encompasses values below this threshold. This method enables us to effectively categorize our data while accounting for its skewed distribution.



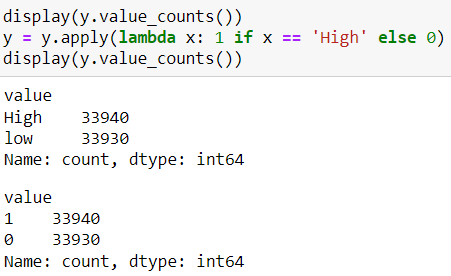




Following the creation of our new variable, termed "value," we made the decision to remove "TOTAL\_VALUE" from our dataset to eliminate redundancy. Additionally, we proceeded to convert the "value" variable into a binary format. Lastly, we also encoded our independent variable to dummy variables as well.



Inserting image...



Upon completing the data preparation phase, we resolved to randomly divide our dataset into training and testing sets. The training set will encompass 80% of the original data, while the remaining 20% will constitute the test set. This random splitting approach was chosen due to the absence of time series information for manual splitting. Additionally, we reasoned that splitting based on zip codes or locations might introduce inaccuracies into our results.

**Exploratory Data Analysis**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | LIVING\_AREA | LAND\_VALUE | BLDG\_VALUE | TOTAL\_VALUE | YR\_BUILT | YR\_REMODEL | BED\_RMS | FULL\_BTH | HLF\_BTH | KITCHENS | TT\_RMS |
| frequency | 68219 | 68219 | 68219 | 68219 | 68219 | 68219 | 68219 | 68219 | 68219 | 68219 | 68219 |
| mean | 1612.512153 | 118741.10 | 778621.80 | 897370.70 | 1916.185022 | 2000.985253 | 2.943564 | 1.723479 | 0.264457 | 1.299301 | 6.552661 |
| std | 1101.715956 | 268946.90 | 754506.70 | 887404.30 | 35.652726 | 15.320722 | 2.130875 | 0.93277 | 0.499581 | 0.636564 | 3.984309 |
| min | 100 | 0.00 | 8300.00 | 48300.00 | 1700 | 1900 | 0 | 0 | 0 | 0 | 1 |
| 25% | 795 | 0.00 | 439900.00 | 502400.00 | 1899 | 1986 | 2 | 1 | 0 | 1 | 4 |
| 50% | 1255.2 | 0.00 | 590500.00 | 689900.00 | 1905 | 2004 | 2 | 1 | 0 | 1 | 5 |
| 75% | 2159 | 209300.00 | 829500.00 | 970000.00 | 1927 | 2015 | 4 | 2 | 0 | 1 | 8 |
| max | 21710.8 | 9571500.00 | 21891000.00 | 30143000.00 | 2021 | 2022 | 17 | 17 | 7 | 5 | 20 |

***Table 1: Summary of Residential Property based on different variables***

The table is a thorough overview of residential properties in Boston, summarizing essential attributes such as living area, construction history, and interior features. Spanning from the 1700s to 2021, the construction history showcases the rich architectural heritage of the city, with properties revealing diverse periods of origin. Furthermore, insights into renovation trends unveil a prevalent pattern of remodeling, particularly around the year 2000, indicating a significant phase of property enhancement and modernization.

Moreover, the dataset offers valuable insights into the interior composition of properties, delineating average counts and distributions of bedrooms, bathrooms, and kitchens. On average, properties boast 2.94 bedrooms, with notable variability evident in bedroom counts. Bathroom configurations predominantly feature full bathrooms, averaging approximately 1.72, while half bathrooms are less prevalent, with an average count of 0.26. Additionally, most properties encompass a single kitchen, with an average count of approximately 1.30, indicating a standard layout across residential units in the dataset. This comprehensive portrayal of property attributes lays the groundwork for further analysis, enabling a nuanced understanding of Boston's diverse real estate landscape.

A graph of a person

Description automatically generated with medium confidence

***Graph1: Property Assessment Counts by Neighborhood in Boston***

The bar chart illustrating property assessments across diverse neighborhoods in Boston unveils intriguing findings. Boston, Dorchester, and JP emerge as the leaders with the highest property counts, indicating significant real estate activity and appeal among investors and residents alike. In contrast, Roxbury Crossing, Dedham, and Brookline exhibit minimal counts, underscoring variations in development and investment levels across different areas. Dedham and Brookline, recognized for their residential housing, demonstrate relatively lower property assessments, shedding light on the intricate dynamics influencing development patterns in Boston. This disparity underscores the necessity for in-depth exploration within each neighborhood and housing segment.

A graph of a number of rooms

Description automatically generated

A graph of a bed

Description automatically generatedA graph of different colored bars

Description automatically generated

***Graph 2: Room Types Frequency in FY 2023***

The histogram portraying the distribution of total rooms among housing in the Boston area unveils a positively skewed pattern. The majority of houses cluster within the 4-5 total room range, closely followed by the 3-4 total room category, which exhibits approximately 15,000 and 10,300 occurrences respectively.

Similarly, the second bar chart also demonstrates positive skewness. Notably, housing units with two bedrooms dominate the distribution, with units featuring one bedroom showing a significant count of approximately 14,500 occurrences, followed by around 12,000 instances for three-bedroom units. This distribution indicates a concentration of housing units towards the lower end of the bedroom spectrum, particularly within the 1–2-bedroom range.

Lastly, the final bar chart showcases the distribution of bathroom types. The high frequency of zero half bathrooms suggests that many apartments likely possess full bathrooms. As the number of bathrooms increases, the frequency gradually diminishes, indicating a decreasing trend in occurrence.

A graph of a number of points

Description automatically generated

***Graph 3: Boxplot of Property Values***

Graph 3 presents a box plot illustrating the distribution of property values in Boston. The notation "1.0e7" signifies 10,000,000 dollars. As depicted in the graph, the average house value in Boston is below 5,000,000 dollars. However, notable outliers are observed in the data, with the highest value exceeding 30,000,000 dollars. These outliers represent properties with exceptionally high values, indicating the presence of luxury real estate within the Boston market. The contrast between the average value and the extreme outliers highlights the wide range of property values and the potential for significant variations in the local housing market.

A green squares with white text

Description automatically generated

***Figure 1: Correlation Matrix***

In Figure 1, the correlation matrix showcases the relationships among selected variables concerning the total values of properties. Specifically, we examined the correlation between total property values and three key variables: living area in square feet, year of remodel, and total number of rooms. The analysis reveals that the variable most positively correlated with property values is the "living area," exhibiting a correlation coefficient of 0.49. This indicates that larger properties tend to command higher values. Interestingly, although the number of rooms and the year of property remodel do not display as strong correlations, they still exhibit positive trends, with correlation coefficients of 0.26 and 0.22 respectively.

A collage of graphs

Description automatically generated

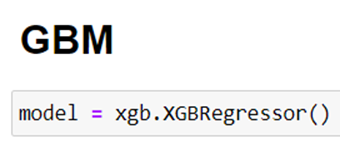
***Figure 2: Pair Plot for Correlation Matrix***

The pair plot shown above corresponds with the correlation matrix in Figure 1, providing visual insights into the relationships between variables. Observing the plot for living area versus property value, a clear upward trend is evident, indicating a positive correlation between the two variables. Similarly, the plot for the year of remodel, while exhibiting a slightly weaker correlation compared to living area, still showcases a positive relationship with property value, aligning with the patterns observed in the correlation matrix. However, the plot for the total number of rooms presents a somewhat ambiguous pattern, despite having a positive correlation coefficient similar to that of the year of remodel.

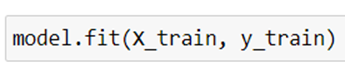
**Code Walk Through**

**1. Regression**

*XGBoost*

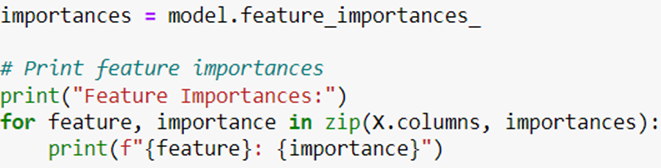


After splitting the data into training and testing sets, we opt to utilize XGBoost for Gradient Boosting Machine (GBM) to predict the value of Boston properties. Given that the target variable, total value, is numerical, employing XGBRegressor is more appropriate than XGBClassifier. This enables us to effectively model the continuous nature of the target variable and make accurate predictions. Let's proceed with implementing XGBoost's regression capabilities to train our model and evaluate its performance.

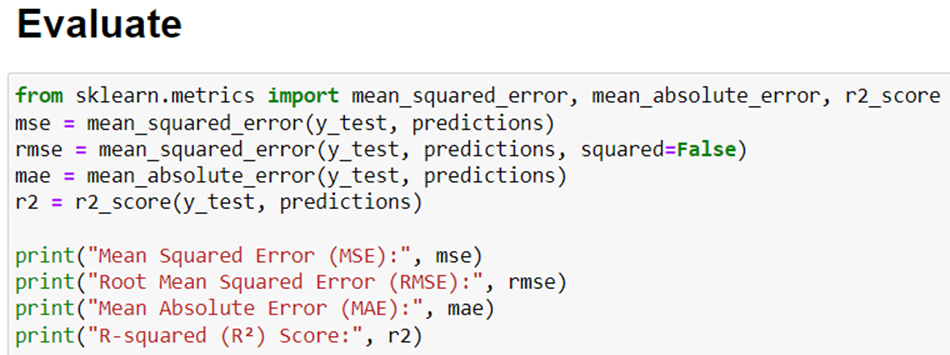




Fitting the model to the training data and making predictions on the testing data (X\_test) using XGBoost's regression. This code initializes the XGBoost regression model, fits it to the training data (X\_train and y\_train), and then uses it to predict the target variable for the testing data (X\_test).

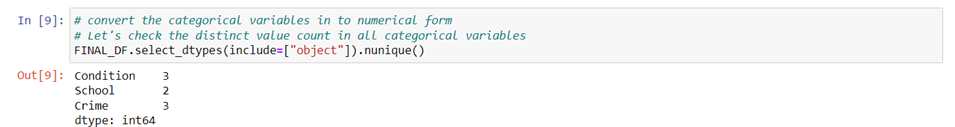


After generating predictions, I've decided to visualize the relationship between the actual and predicted values by plotting a graph. This will provide a clear comparison between the two sets of values, allowing for an assessment of the model's predictive performance. Additionally, I've explored the feature importance of the variables used in the model to understand which features have the most significant impact on predicting the total value of Boston properties. This analysis will help identify key factors driving property values in the Boston area, contributing to a better understanding of the underlying dynamics captured by the model.

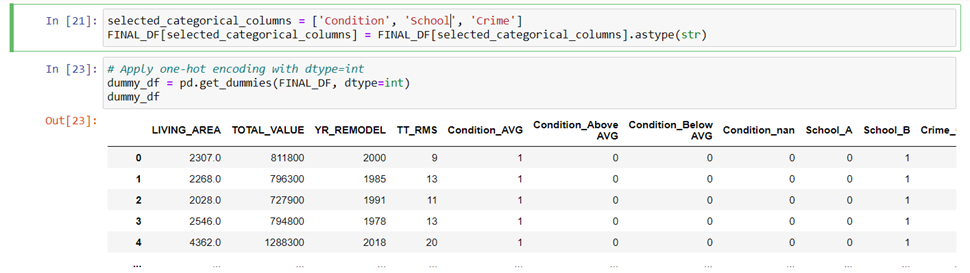


Finally, to comprehensively evaluate the performance of our analysis, we'll utilize several key metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R2). These metrics provide valuable insights into the accuracy and predictive power of our model. MSE, RMSE, and MAE quantify the average difference between the actual and predicted values, with lower values indicating better performance. R-squared measures the proportion of variance in the target variable that is explained by our model, with values closer to 1 indicating a better fit. By comparing these metrics with those from other analyses, we can assess the relative effectiveness of our model and make informed decisions about its deployment.

*Random Forest*



The first step is to check the distinct value count in all categorical variables. According to the result, it indicates that the categorical variable "Condition" has three unique values. Similarly, the categorical variable "School" has two distinct values, while the variable "Crime" encompasses three distinct values.



Secondly, I created categorical variables in my dataset FINAL\_DF for machine learning. I select 'Condition', 'School', and 'Crime' columns and convert them to string type. Then, I apply one-hot encoding using pd.get\_dummies(), ensuring numerical compatibility for machine learning algorithms.



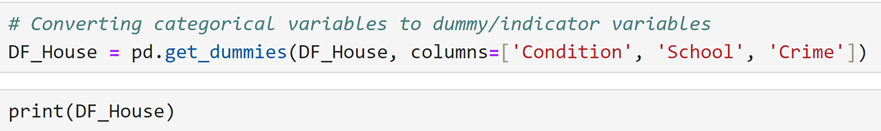
These codes describe my steps in building and training a Random Forest Regressor model (MODEL\_RF) to predict a target variable. I configured the Random Forest with 100 trees (n\_estimators=100) and a consistent random state (random\_state=42) for effective results. After training the model with the provided data (X\_train and y\_train), I made predictions on the testing data (X\_test) using the predict() method. Then, I calculated and stored the feature importances named feature\_imp\_rf, providing insights into the most influential features for predicting the target variable.

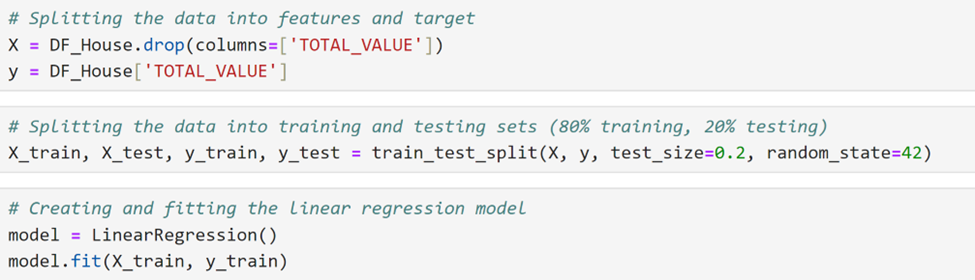


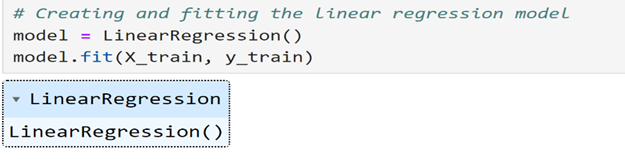
Lastly, I evaluated the performance of the result of random forest model by using metrics such as, MAE, MSE, and R-Squared. Also, I plot the linear line to compare the predicted and actual values.

*Linear Regression*

Similarly for the linear regression model, categorical variables in the dataset are converted into dummy variables using one-hot encoding to facilitate regression analysis. Then the dataset is split into two sets: a training set and a testing set, using an 80-20 ratio. The training set comprises 80% of the data, while the testing set contains the remaining 20%.



A linear regression model is chosen for its simplicity and interpretability. The model is created and trained using the training data with the LinearRegression class from scikit-learn.



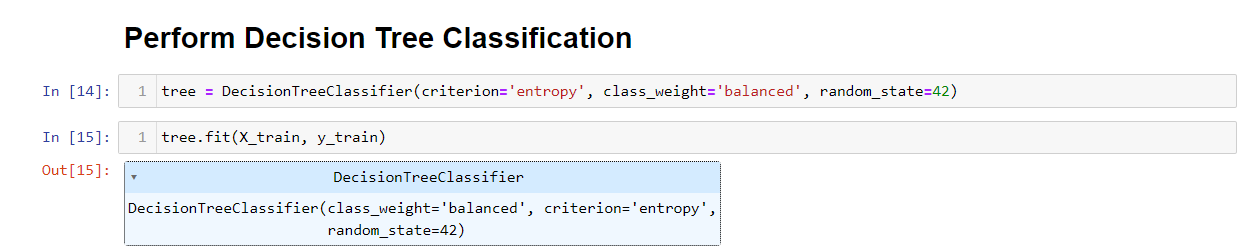
The trained model is used to make predictions on the testing data. Several regression evaluation metrics are calculated to assess the performance of the model, including Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared value.



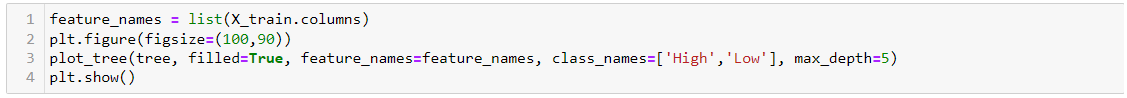
A scatter plot is created to visualize the relationship between the actual and predicted house prices. Additionally, a red line is added to the plot to represent the ideal scenario where predicted values perfectly match the actual values.

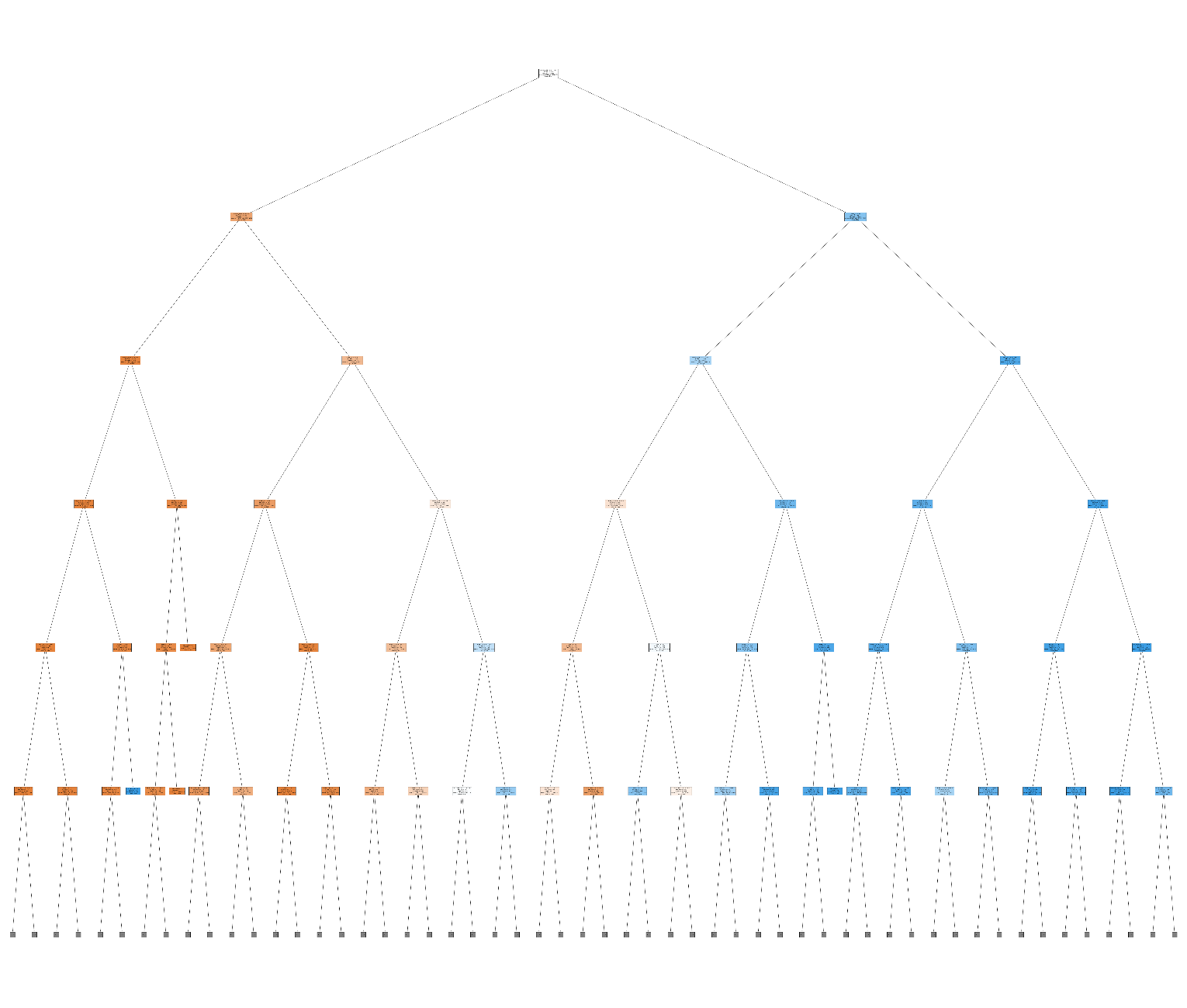
**2. Classifier**

*Decision Tree*

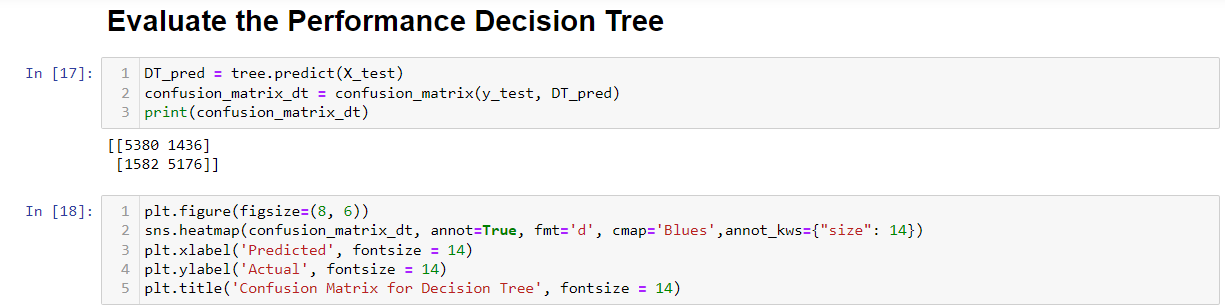


Subsequently, we initialized a decision tree classifier, specifying parameters like criterion='entropy', which guides the model in making optimal splits based on information gain. Additionally, class\_weight='balanced' was set to handle class imbalance by adjusting weights inversely proportional to class frequencies. By setting random\_state=42, we ensured reproducibility of results across different runs. Finally, we trained the decision tree classifier on the training data using the fit method, allowing the model to learn from the features and labels provided.

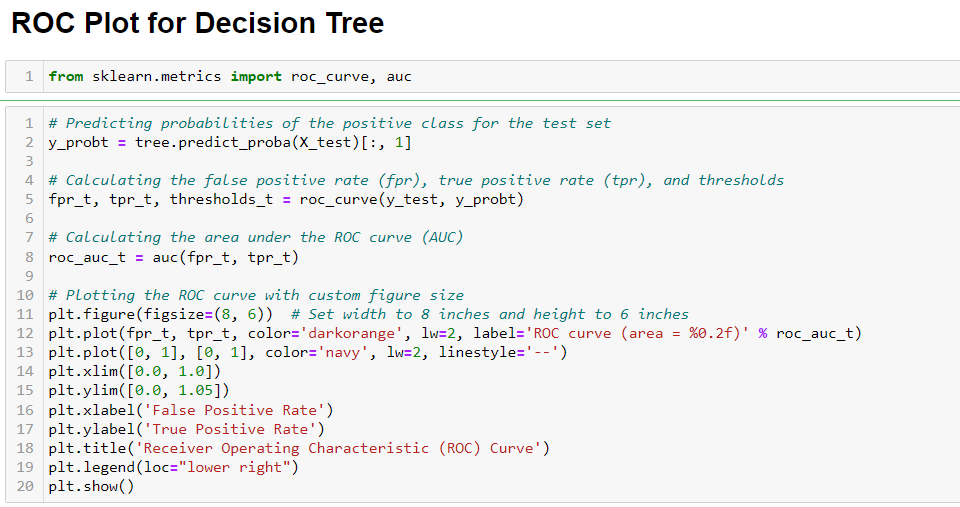
 Next, I created a decision tree figure by using the codes in the above picture and below is the result of the decision tree with depth level of 5.



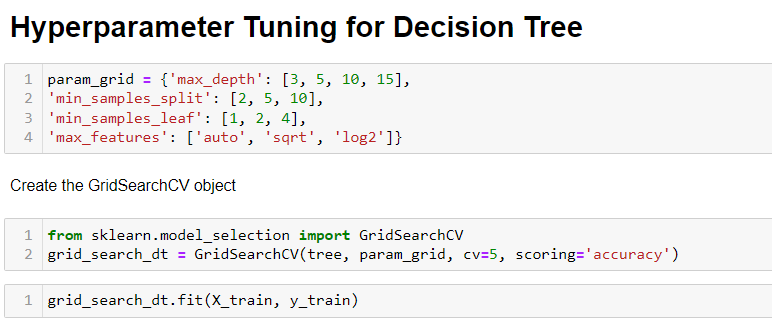
In evaluating our decision tree classifier on the test dataset, we first made predictions using test features. Then, we computed a confusion matrix to compare these predictions with actual labels, revealing true positive, true negative, false positive, and false negative counts. Visualizing this matrix with a heatmap helped us understand classification performance visually.

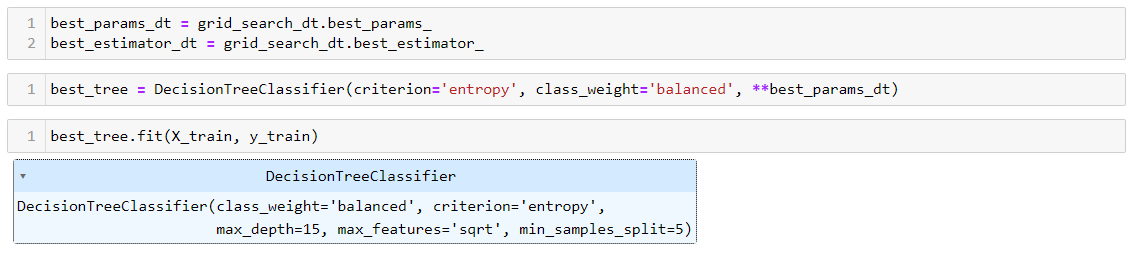


Lastly, we generated the ROC curve to see the performance of the decision tree model. The below picture displays the codes that we utilized to create the ROC.

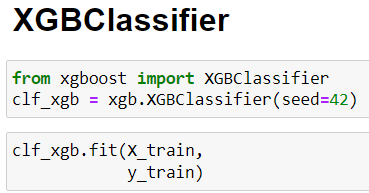


Furthermore, we also conducted hyperparameter tuning for decision tree model to observe if it could derive a better model.

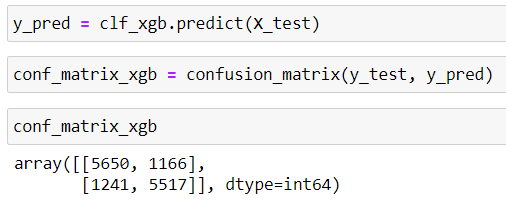


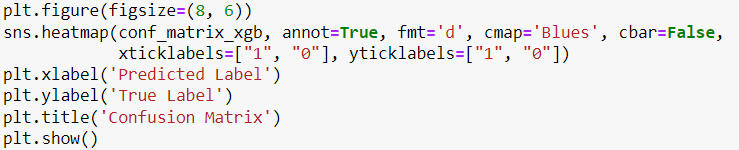


*XGBoost*

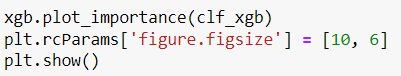
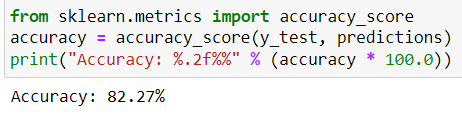


After retrieving our target variable as binary and independent variables as dummy variables, we start importing XGB Classifier from XGBoost packages and set seed equal to 42 to ensure that that the random initialization of certain components within the algorithm (like the random initialization of model parameters or the random shuffling of data during training) is performed in a predictable and reproducible manner. Then we start training our data by fitting X\_train and y\_train into the model.

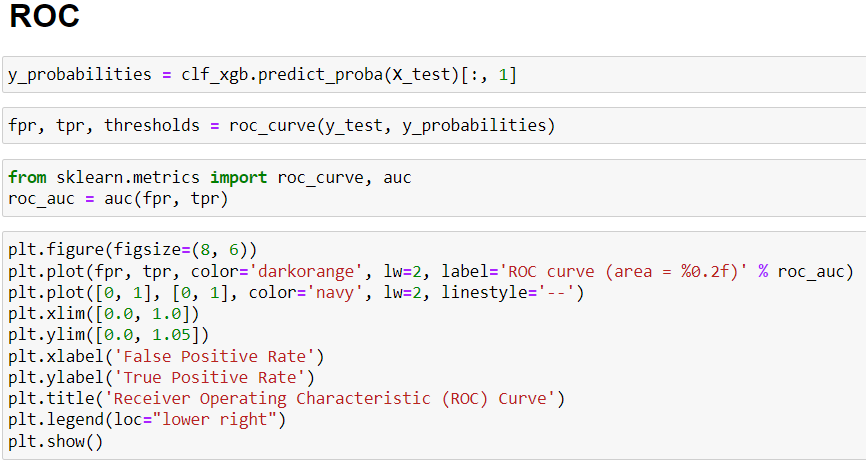




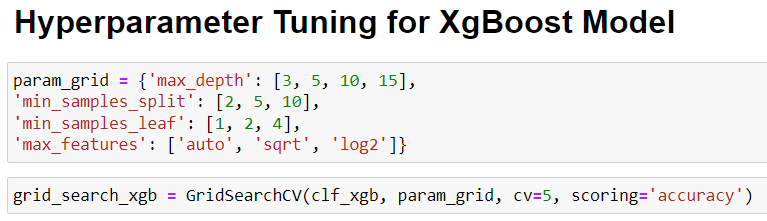
Once the training process on our dataset is complete, we proceed to make predictions on our target variable using the test dataset, X\_test. Subsequently, we compare the predicted values, y\_pred, with the actual values in our test dataset, y\_test, to evaluate the performance of our model. One common method to assess the predictive accuracy is by generating a confusion matrix, which provides a detailed breakdown of the model's predictions compared to the ground truth. Analyzing this matrix allows us to gain insights into the model's performance, including its ability to correctly classify instances into different categories and identify any potential misclassifications. By comparing y\_pred with y\_test and examining the confusion matrix, we can refine our understanding of the model's strengths and weaknesses, ultimately guiding further optimizations or adjustments as necessary.

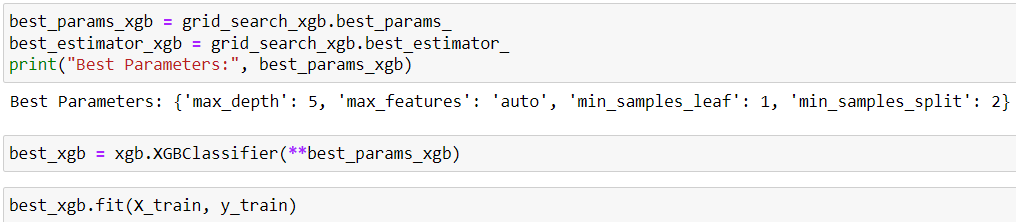


Additionally, we gauge the accuracy of our model by importing the accuracy\_score function from the sklearn.metrics library. This allows us to quantitatively assess the model's performance by comparing the predicted labels with the actual labels in our test dataset. Furthermore, we delve into understanding the significance of different features in our model by examining feature importance. This analysis sheds light on which features contribute the most towards predicting the target variable, offering valuable insights into the underlying patterns learned by the model.

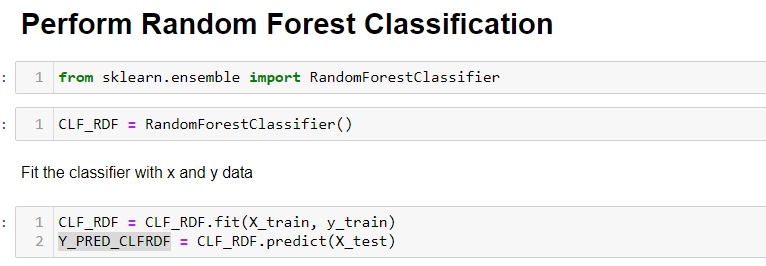


We also evaluate our model by performing the ROC curve and AUC. Moreover, we also do the classification report to see the recall and precision as well.

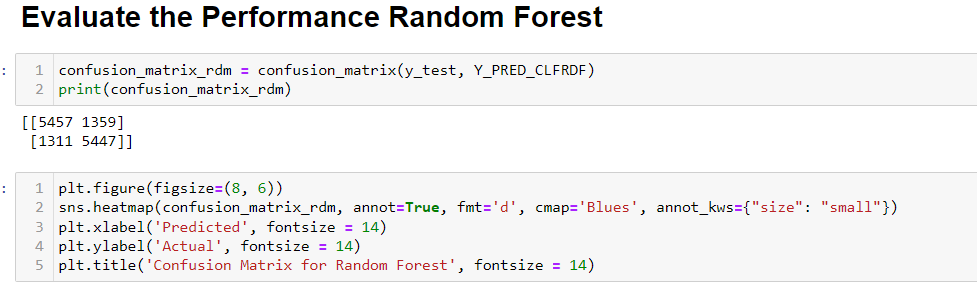


 We also perform Hyperparameter Tuning to enhance our prediction model and increase its accuracy.

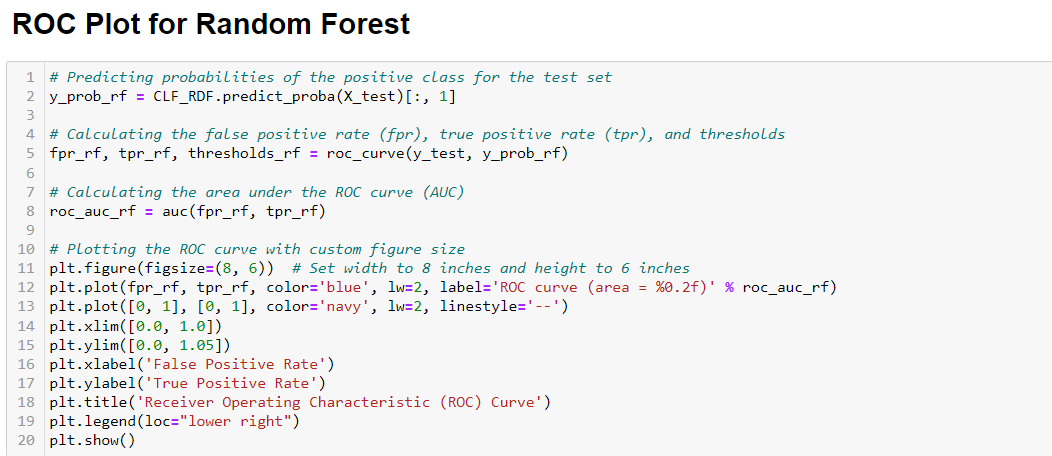
*Random Forest*



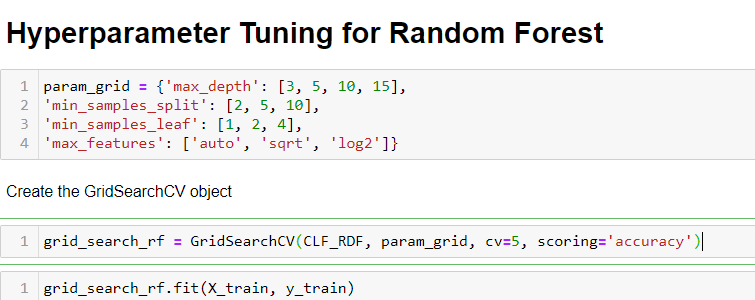
Similarly to Decision Tree model, for the Random Forest Classifier, we use the code RandomcForestClassifier() to predict the outcome of data.



After we created the Random Forest model, nowwe use confusion matrix to evaluate the result and create heatmap to have a clearer picture and better understanding of the performance.



Not only conducting the confusion matrix, but also the ROC curve and AUC, in order to observe the better visualization of the performance of the Random Forest model in predicting the outcome.





Subsequently, as we believed that the model could perform better; therefore, we conducted the Hyperparameter Tuning and test with the “test data” to see if this model could derive a finer result.

*Logistic Regression*

ROC and AUC curve were made then the confusion matrix is calculated to assess the model's performance, showing counts of true positives, true negatives, false positives, and false negatives. The confusion matrix is then visualized using a heatmap, providing a graphical representation of the model's performance in predicting class labels. The classification report is printed, displaying metrics such as precision, recall, and F1-score for each class, as well as overall accuracy.

A screenshot of a computer code

Description automatically generated

A close-up of a report

Description automatically generated

Hyperparameter tuning is performed to optimize the logistic regression model's performance. Through grid search cross-validation, the code systematically explores different combinations of hyperparameters to identify the most effective configuration.

A screenshot of a logistic program

Description automatically generated

Subsequently, the model's performance is evaluated again on the test dataset, with a new confusion matrix and classification report generated. These evaluation metrics provide a clear understanding of the model's accuracy, precision, recall, and F1-score after hyperparameter tuning.

*A screenshot of a computer program

Description automatically generated*

**Analysis**

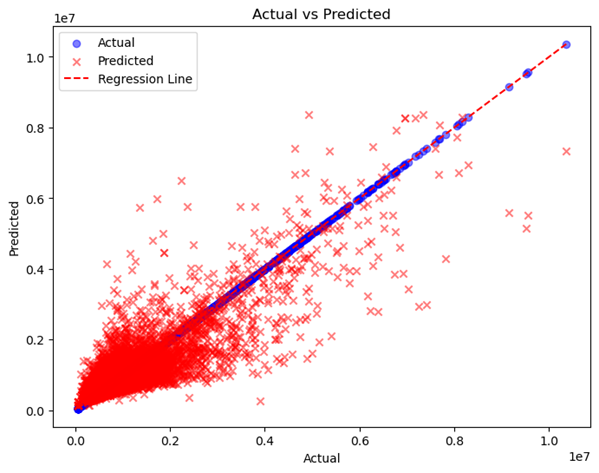
In the following analysis, our team has conducted regression and classification tasks. Initially, our dataset comprised predominantly numerical variables with limited categorical features. Consequently, our initial focus was on regression analysis, aligning with the nature of our dataset.

When it comes to predicting property values from housing data, the choice of machine learning algorithms like decision trees, random forests, XGBoost, and logistic regression involves various factors. For example, decision trees and logistic regression are valued for their interpretability, providing clear insights into the factors influencing predictions. These models are particularly useful for understanding feature importance and direction of influence in property valuation decisions. Conversely, random forests and XGBoost are adept at capturing complex relationships and interactions between features often found in housing data. Their ensemble approach allows them to combine predictions from multiple trees, reducing overfitting and enhancing reliability, which is crucial for handling noisy or high-dimensional datasets typical of housing data.

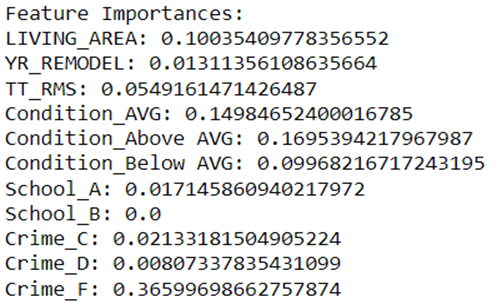
Furthermore, the algorithm choice is influenced by performance metrics and computational constraints. XGBoost, known for its high performance and scalability, tends to outshine other methods in predictive accuracy, especially in structured/tabular data like housing datasets. Random forests are also favored for their resilience to missing values and outliers, offering dependable performance in real-world data scenarios. On the contrary, logistic regression stands out for its simplicity and faster training times, making it suitable for rapid prototyping or resource-limited environments. Ultimately, selecting the most suitable algorithm depends on a comprehensive understanding of the dataset's characteristics, interpretability requirements, performance goals, and computational resources available.

**1. Regression**

*XGBoost*



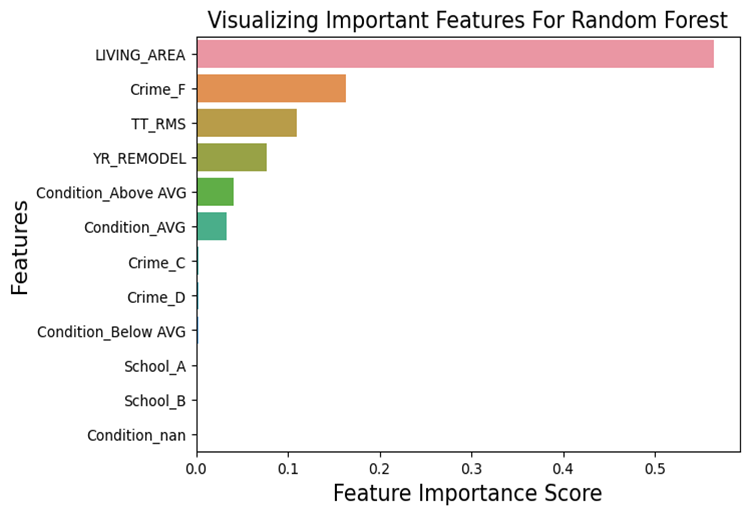
The chart above depicts the relationship between the actual values (blue dots), predicted values (red X), and the regression line generated by the GBM of XGBoost (red line). It's evident that while the actual values closely align with the regression line, indicating a strong fit, the predicted values exhibit some deviation from the regression line. This discrepancy suggests that errors occurred during the prediction process, leading to deviations from the expected trend. Despite these errors, the model still captures the general trend in the data, albeit with some degree of variability.



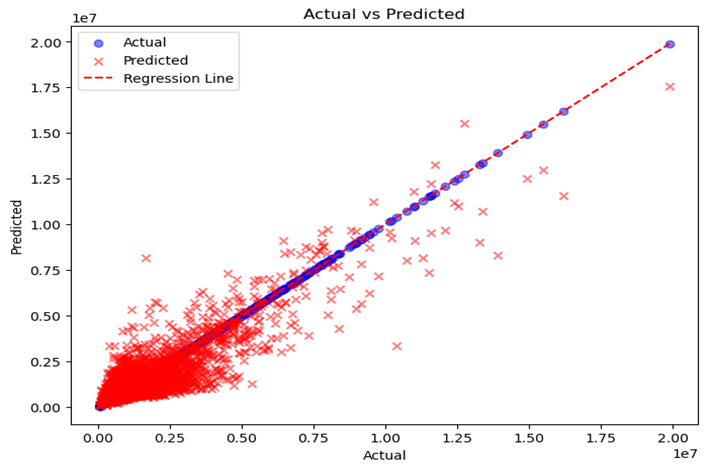
The visualization above illustrates the feature importances of the variables. It is evident that "Crime Grade F" has the highest influence on the value of Boston properties, indicating that areas with a higher crime rate significantly impact property values negatively. Conversely, "Living Area" exhibits moderate importance, suggesting that larger living areas generally contribute positively to property values, but they are not the sole determining factor.

Interestingly, "School Grades A and B" show very low feature importance, implying that the quality of schools in the area has minimal impact on property values. This suggests that other factors, such as crime rate and living conditions, may overshadow the influence of school quality in determining property values in Boston.

*Random Forest*



By conducting feature importance, it provides insights into the importance of each feature in predicting the target variable, as determined by the Random Forest Regressor model. Among the features, "LIVING\_AREA" emerges as the most influential, with a high importance score of approximately 0.564, indicating its significant impact on the prediction task. Following by "Total Rooms" and "Year Remodel," with importance scores of around 0.109 and 0.077, respectively, suggesting their substantial contributions. Other features such as "Condition" and "Crime" categories exhibit varying importance scores, reflecting their relative impacts on the prediction. Notably, features related to property condition and crime rates show discernible effects but with lower importance scores compared to factors like living area and number of rooms.



Then, I plot the linear line to visualize the performance of the random forest prediction. We could observe that the ‘x’ mark represents the predicted values where blue dot represents actual values. According to the graph, although the prediction values are located near the actual values, they get clustered around the very end of the line which indicates that there are discrepancies between predictions and actual outcomes.

*Linear Regression*

A graph with red and blue dots

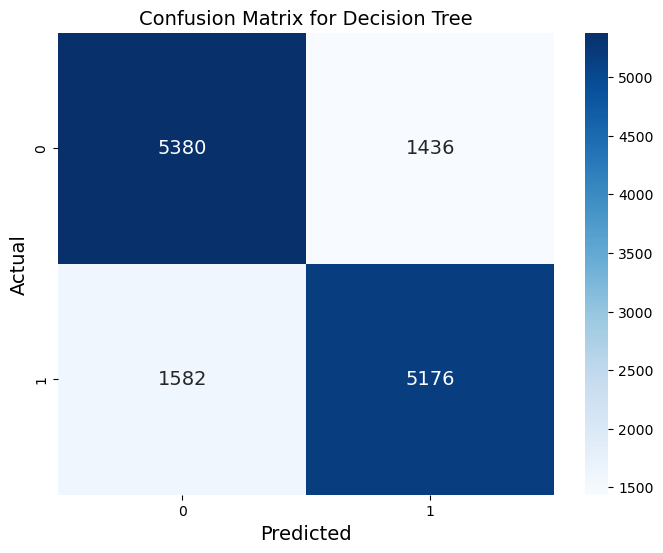
Description automatically generated

The scatter plot illustrates the relationship between the actual total values (represented by blue dots) and the predicted total values (indicated by red 'x' markers) generated by a linear regression model. Notably, both the regression line (dashed red line) and the diagonal formed by the blue dots exhibit a similar diagonal pattern, indicating a positive correlation between the actual and predicted values. However, upon closer examination, it's apparent that while the regression line and the actual values closely follow the diagonal pattern, the predicted points are clustered more towards the corner of the plot, particularly in the range of 0.00 to 0.50. This clustering towards the corner suggests that while the model captures the general trend of the data, it tends to underestimate or overestimate total values, particularly in the mid to high range.

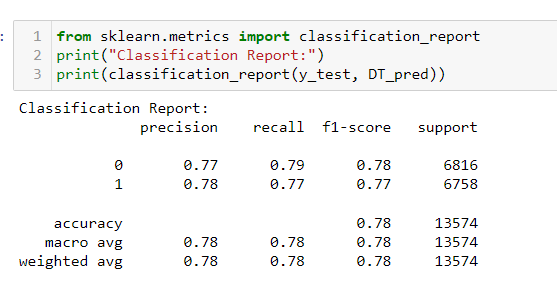
**2. Classifier**

*Decision Tree*

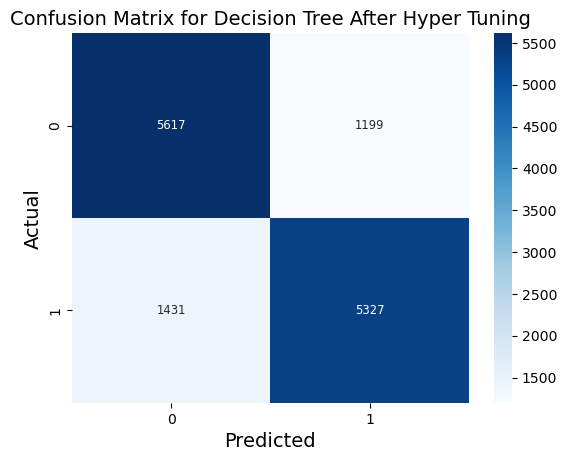
The figure below displays the confusion matrix of decision tree with the counts of true positives, false positives, false negatives, and true negatives, respectively. In this specific instance, we had 5380 true negatives, 5176 true positives, 1436 false positives, and 1582 false negatives.

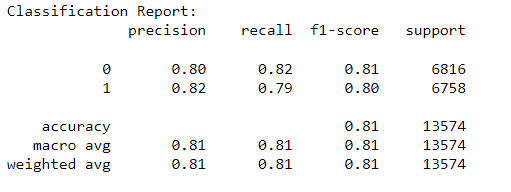


Moreover, we also perform a classifier to see the accuracy of the model as shown in the below figure. This report includes precision, recall, and F1-score metrics for each class, along with overall accuracy and average scores. Precision measures the accuracy of positive predictions, recall indicates the ratio of correctly predicted instances, and F1-score is the harmonic mean of precision and recall. Additionally, the report provides support values, representing the number of occurrences of each class in the test dataset. For class 0, the precision is 0.77, recall is 0.79, and F1-score is 0.78. Similarly, for class 1, the precision is 0.78, recall is 0.77, and F1-score is 0.77. The overall accuracy of the classifier is 78%, indicating its ability to correctly classify instances from both classes.

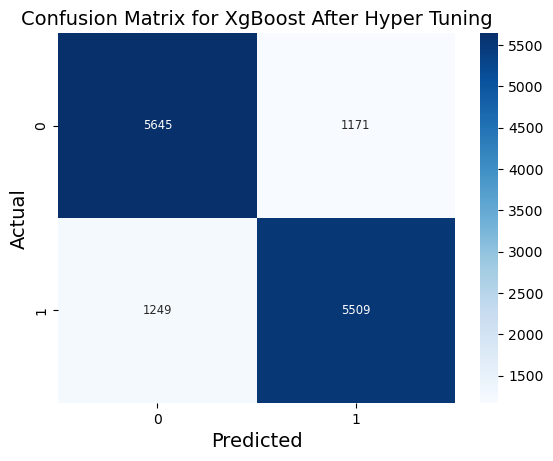
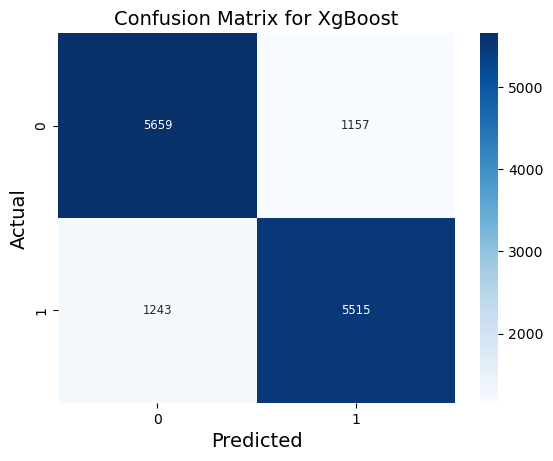


After we created a new model for decision tree by using hyperparameter tuning, we repeatedly evaluated the performance by conducting the confusion matrix. In this instance, we have 5617 true negatives, 5327 true positives, 1199 false positives, and 1431 false negatives. Based on the result below, we could observe that this one shows an overall improvement in classification performance. The classifier achieved better results in terms of correctly identifying instances from both classes and reducing misclassifications with an increase to 81% accuracy.

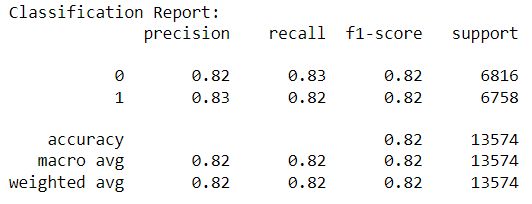


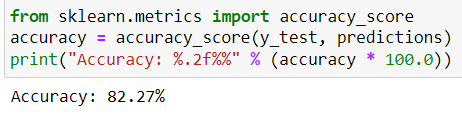


*XGBoost*



When comparing the confusion matrices of the XGBoost Classifier (XGBClassifier) and the XGBoost Classifier with hyperparameter tuning (XGBClassifier with hyper tuning), several observations can be made. In the case of XGBClassifier, it correctly classified 5515 positive instances (True Positives) and 5659 negative instances (True Negatives), while incorrectly classifying 1157 negative instances (False Positives) and 1243 positive instances (False Negatives). Following hyperparameter tuning, there were marginal changes observed. The number of True Positives decreased slightly to 5509, while False Positives increased to 1171. Additionally, False Negatives increased marginally to 1249, and True Negatives decreased slightly to 5645. These alterations suggest that hyperparameter tuning had a minimal impact on the classification performance of the XGBoost Classifier, with only slight fluctuations observed in the classification of positive and negative instances. Since the XGB classifier provides a slightly higher number of true positive and true negative, and the other results are very similar, thus, we would like to provide further analysis of XGB classifier solely.

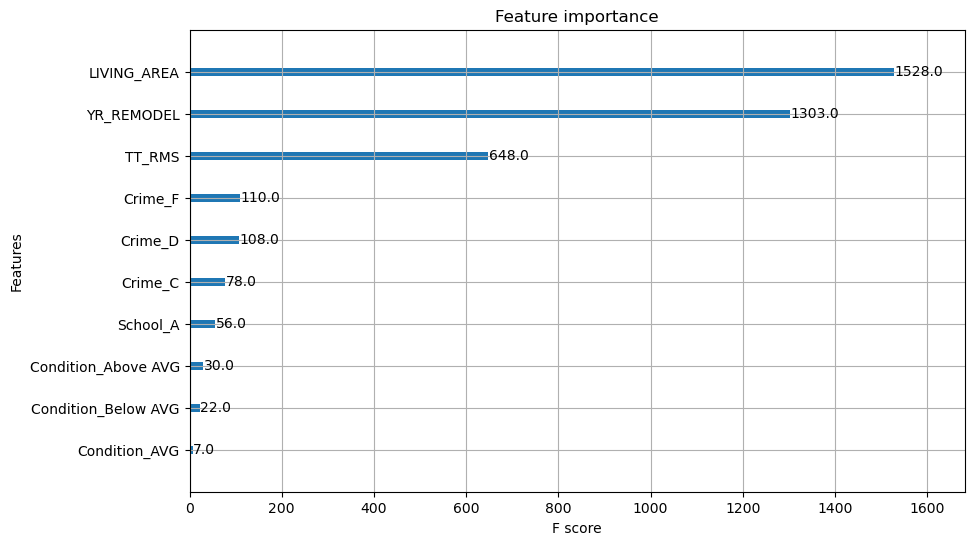




The classification report reveals our precision, recall, and accuracy of the model. Precision measures the proportion of true positive predictions among all instances predicted as positive. In this report, for class 0, the precision is 0.82, indicating that 82% of the instances predicted as class 0 were class 0. Similarly, for class 1, the precision is also 0.83, indicating 83% precision for class 1 predictions.

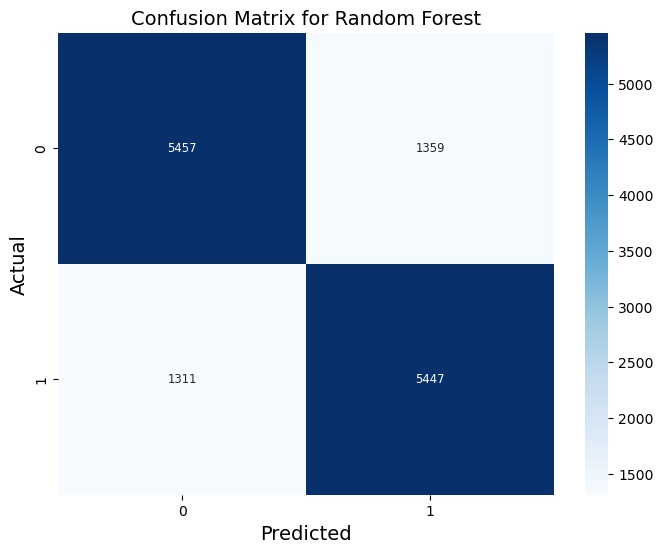
Recall, also known as sensitivity, measures the proportion of true positive instances correctly identified by the model. In this report, for class 0, the recall is 0.83, indicating that 83% of all actual class 0 instances were correctly classified as class 0. Likewise, for class 1, the recall is also 0.82, indicating 82% recall for class 1 instances.

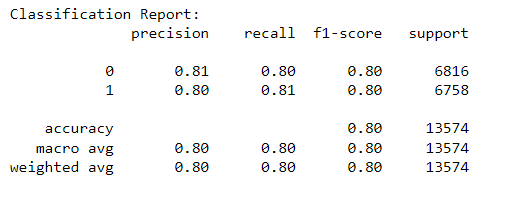
The overall accuracy of the model is also provided, which is calculated as the proportion of correctly classified instances out of the total number of instances. In this report, the model achieves an accuracy of 82%. To be precise, our accuracy of this model is 82.27%.

 Feature importance is a crucial aspect of understanding how a machine learning model makes predictions. It helps identify which features (or variables) contribute the most to the model's predictions. As you can see above, the LIVING\_AREA is the variable that contributes most of the prediction following with YR\_REMODEL (Year Remodel), TT\_RMS (Total rooms), Crime rate, School Grade, and Condition.

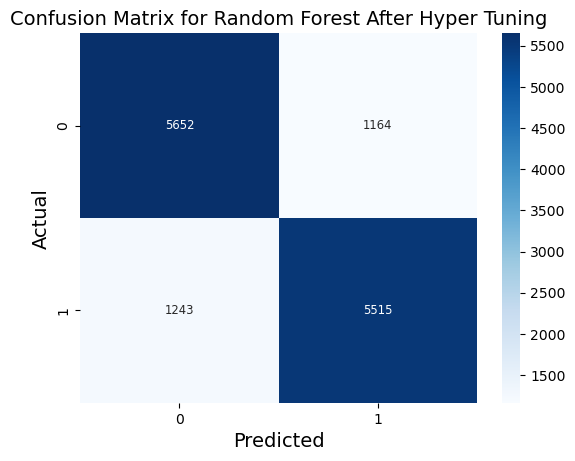
*Random Forest*

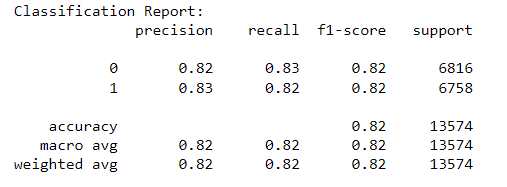
Similarly, we did confusion matrix to see the performance of the model. Where in Random Forest, we had 5457 true negative, 5447 true positive, 1359 false positives, and 1311 false negatives. Moreover, according to the classification report, the accuracy of this model is around 80%.





However, as we chose to perform Hyperparameter Tuning, surprisingly, it derived a better result with an increase in accuracy to around 82%. Comparing the confusion matrices of Before and After Hyperparameter Tuning, it reveals that, with the hyperparameter tuning, it performs better overall. It correctly identifies more instances of the positive class (higher true positives) and fewer instances are incorrectly classified as positive when they are negative (lower false positives). Additionally, after conducting Hyperparameter Tuning, the result has fewer instances incorrectly classified as negative when they are positive (lower false negatives) and correctly identifies more instances of the negative class (higher true negatives).





*Logistic Regression*

The evaluation of the logistic regression model for binary classification revealed promising performance metrics based on the confusion matrix and classification report. The confusion matrix showcased a balanced distribution of true positives and true negatives, with 5,770 instances correctly classified as negative (True Negatives) and 5,020 instances correctly classified as positive (True Positives).

A screenshot of a computer

Description automatically generated

However, the model also misclassified 1,046 instances as positive (False Positives) and 1,738 instances as negative (False Negatives). This indicates that the model effectively distinguished between the two classes but may require refinement to improve sensitivity towards positive instances.

A screenshot of a computer

Description automatically generated

Precision and recall values further supported the model's proficiency, with class 1 (positive class) exhibiting slightly lower recall compared to class 0 (negative class). However, both precision and recall scores remained notably high, with precision values of 0.77 for class 0 and 0.83 for class 1, and recall values of 0.85 for class 0 and 0.74 for class 1.

The F1-scores indicated a satisfactory balance between precision and recall for each class, with F1-scores of 0.81 for class 0 and 0.78 for class 1. Overall, the logistic regression model achieved an accuracy of 79%, with 10,790 instances correctly classified out of 13,574 instances.

A screen shot of a computer

Description automatically generated

The ROC curve and accuracy score provide insights into the performance of the logistic regression model. The ROC curve illustrates the model's ability to differentiate between the two classes, with an Area Under the Curve (AUC) of approximately 0.77 indicating moderate discriminatory power. However, it's essential to note that the curve doesn't reach the upper-left corner, suggesting room for improvement.

The accuracy score of around 76% indicates that the model correctly predicts class labels for a considerable portion of the test data. While this accuracy is reasonable, it may not suffice for applications requiring higher precision or recall. To enhance model performance, further exploration could involve experimenting with feature engineering, alternative algorithms, or hyperparameter tuning. Additionally, understanding the specific context and objectives of the application is crucial for interpreting these findings effectively and making informed decisions regarding model refinement and implementation.

After applying hyperparameter tuning to the logistic regression model, the performance was evaluated using the newly tuned model. The confusion matrix obtained from the evaluation illustrates the following results:

A screenshot of a computer

Description automatically generated

Comparing this confusion matrix to the original one, it's evident that hyperparameter tuning has led to improvements in the model's performance, particularly in reducing false positives and false negatives. The true positives and true negatives remain consistent with the original model, indicating that the model's ability to correctly classify instances has been maintained.

**Comparison**

**Regression**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **XgBoost** | **Random Forest** | **Linear Regression** |
| **Mean Absolute Error** | 214770.09 | 224551.25 | 338884.67 |
| **Mean Squared Error** | 142,185,212,517 | 173,796,717,314 | 326,764,002,409.7 |
| **R-squared** | 0.71 | 0.776 | 0.574 |

The comparison of Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared values across three different regression models, XGBoost, Random Forest, and Linear Regression, provides valuable insights into their respective performances in predicting total values.

XGBoost exhibits the lowest MAE of $214,770.09, indicating the smallest average absolute difference between the actual and predicted total values. Random Forest follows closely with an MAE of $224,551.25, while Linear Regression demonstrates a higher MAE of $338,884.67. This suggests that XGBoost and Random Forest models are more accurate in predicting total values compared to Linear Regression.

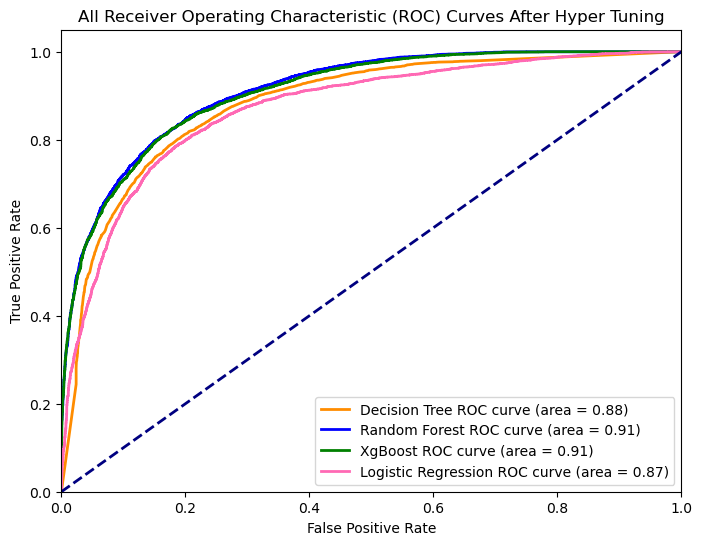
Similarly, XGBoost and Random Forest outperform Linear Regression in terms of MSE, with XGBoost having the lowest MSE of $142,185,212,517 and Random Forest following with $173,796,717,314. Linear Regression exhibits the highest MSE of $326,764,002,409.7, indicating larger squared differences between the actual and predicted values.

Furthermore, R-squared values provide insights into the proportion of variance explained by the models. While XGBoost achieves the highest R-squared value of 0.71, indicating that approximately 71% of the variance in total values is explained by the model, Random Forest follows closely with an R-squared value of 0.776. In contrast, Linear Regression demonstrates a lower R-squared value of 0.574, suggesting that it explains less variance in the total values compared to XGBoost and Random Forest.

**Classifier**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **XgBoost** | **Random Forest** | **Logistic Regression** | **Decision Tree** |
| **Accuracy** | 82% | 82% | 76% | 81% |

The provided accuracy scores for four different models, XGBoost, Random Forest, Logistic Regression, and Decision Tree, offer valuable insights into their respective performance levels. Both XGBoost and Random Forest models demonstrate identical accuracy rates of 82%, signifying their consistent and robust predictive capabilities. These ensemble methods leverage the power of multiple decision trees to make accurate predictions, making them particularly effective for a wide range of classification tasks. The Decision Tree model follows closely behind with an accuracy of 81%, showcasing its ability to make informed decisions based on the input features. While slightly lower than the ensemble methods, its performance remains commendable. In contrast, Logistic Regression yields the lowest accuracy among the models, achieving a rate of 76%. Despite its simplicity and interpretability, Logistic Regression may struggle to capture complex nonlinear relationships present in the data compared to the more sophisticated ensemble techniques.

 The comparison of the Area Under the Curve (AUC) values among the Decision Tree, XGBoost (XGB), Random Forest, and Logistic Regression models provides insights into their respective predictive capabilities. XGBoost and Random Forest emerge as the top performers with identical AUC values of 0.91, signifying strong discriminatory power and effective classification of positive and negative instances. The Decision Tree closely follows with an AUC value of 0.88, indicating slightly lower but still robust discriminatory ability. Meanwhile, Logistic Regression exhibits the lowest AUC value of 0.87, implying comparatively weaker discriminatory power among the models assessed. Overall, XGBoost and Random Forest demonstrate similar and superior performance in terms of AUC, followed closely by the Decision Tree, while Logistic Regression, while still effective, lags slightly behind in discriminatory ability.

**Recommendation**

In terms of the regression, XGBoost and Random Forest models outperform Linear Regression in terms of predictive accuracy and ability to explain variance in total values. These findings underscore the potential of ensemble methods like XGBoost and Random Forest for accurate and robust prediction tasks, particularly in scenarios involving complex relationships and high-dimensional datasets. Thus, we recommend selecting Random Forest Regressor to predict the numerical outcomes.

Similarly, in terms of classification, we recommend choosing either Random Forest Classification or XgBoost as models could perform with the AUC values of 0.91 which indicates a significant amount of accuracy in predicting the Boston Housing values while the other two models, Decision Tree and Logistic Regression underperform with AUC values of 0.88 and 0.87, respectively.

**Conclusion**

In this comprehensive analysis, we evaluated various machine learning models for both regression and classification tasks using Boston Housing dataset. For regression, we compared XGBoost, Random Forest, and Linear Regression models, while for classification, we assessed XGBoost, Random Forest, Logistic Regression, and Decision Tree models.

Our findings suggest that XGBoost and Random Forest, outperform Linear Regression in regression tasks, exhibiting higher predictive accuracy and better ability to explain variance in total values. Therefore, we recommend employing Random Forest Regressor for predicting numerical outcomes in similar scenarios.

In terms of classification, both XGBoost and Random Forest demonstrate superior performance with identical AUC values of 0.91, indicating significant accuracy in predicting Boston Housing values. On the other hand, Decision Tree and Logistic Regression models, while still effective, exhibit slightly lower AUC values of 0.88 and 0.87, respectively. Hence, we recommend selecting either Random Forest Classification or XGBoost for classification tasks based on their robust performance.

Overall, this comparative analysis underscores the importance of selecting appropriate machine learning models tailored to the specific task at hand. By leveraging ensemble methods like XGBoost and Random Forest, practitioners can achieve accurate predictions and effectively address regression and classification challenges in real-world scenarios.

**References**

Boston Police Incidents (n.d.). Retrieved from <https://boston-pd-crime-hub-boston.hub.arcgis.com/datasets/d42bd4040bca419a824ae5062488aced/explore>

Scikit-learn: Machine Learning in Python. (n.d.). Retrieved from <https://scikit-learn.org/stable/index.html>

XGBoost: Introduction to XGBoost Algorithm in Machine Learning (Jan 16, 2024). Analytic Vidhya. <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/#:~:text=High%20accuracy%3A%20XGBoost%20is%20known,millions%20of%20rows%20and%20columns>

CrimeGrade.org | The Safest Places and Best Home Security. (n.d.). Crimegrade.org. <https://crimegrade.org/>

‌How is Splitting Decided for Decision Trees? | Displayr. (2018, November 7). Displayr. <https://www.displayr.com/how-is-splitting-decided-for-decision-trees/>

Massachusetts ZIP Codes with the Best Public Schools. (n.d.). Niche. Retrieved March 6, 2024, from <https://www.niche.com/places-to-live/search/zip-codes-with-the-best-public-schools/s/massachusetts/>