**KL/SELANGOR RENTAL PREDICTION USING LOGISTIC REGRESSION AND TREES MODELS**

1. **Defining a Problem**
   1. **Problem statement**

When it comes to setting up a rental price or giving out indications, homeowners, valuers, or realtors, tend to rely on the internet to check the rental market rate from property websites such as PropertyGuru, mudah.my, iProperty, and many others. As easy as it may sound, “the inconsistency of information can easily be found in websites as most of the residential property providers did not own its property data. The objective of most residential property web is more focused on providing a platform for users to share their information. Thus, inconsistent information occurs due to different sets of data may not be entered by the same person (Kee Li Yap, 2020). One can choose their preferred rate suitable for their home from the property websites, but how can one assured on the accuracy of the price per the property’s features? Under-pricing can result in a loss of income, while on the other hand, overpricing can make it difficult to rent the property and lose out on a suitable customer base. It is, therefore, crucial to examine the rental price carefully and suggest a fair rental rate that reflects the property’s value (Dong Xue Ying, 2023).

Commonly used models for predicting housing rental include Logistic Regression, Decision Trees, Random Forests, and Gradient Boosting algorithms.

* 1. **Objectives**

1. To identify the trend of rental rate/price.
2. To identify the most correlated features that influence rental rate/prices
3. To build a model that predicts the monthly rental for high-rise residential property in KL/Selangor using a dataset.
   1. **Tools**

In this project, I will use Jupyter Notebook Python to do the entire modeling process from the data analysis to answer the problem

1. **Acquiring of Data**

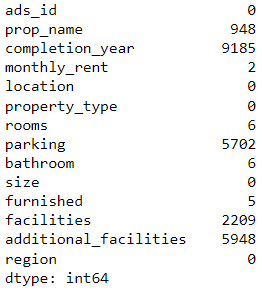
Initially, I was going to use data from JPPH, however, upon evaluating the data the features may not be enough to do the prediction model. Also, the data was seemingly cleaned, so it is not appropriate for this project as it requires a raw dataset.

In this project, I will use a high-rise housing dataset obtained from Kaggel. (<https://www.kaggle.com/datasets/ariewijaya/rent-pricing-kuala-lumpur-malaysi>).

I chose this dataset as it gives more features such as amenities, parking, and furnishing among others (in comparison to the initial dataset), which is important to achieve the objective. Also, it provides monthly rental data as well as I am going to predict the rental with the most accuracy (if possible). This data consists of at least 19k rows and 14 columns from various high-rise developments in KL/Selangor. Thus, the project will focus on KL/Selangor high-rise rental prediction.

1. **Explore the Data**
   1. **Data Cleaning**

In data cleaning, I will evaluate which columns need some cleaning and/or filling by using the **df.isnull().sum()** function in Python. The output shows a total of 9 columns out of 14 columns containing null values.



**df.fillna()**

Null value is a no-no in data analysis. So, these null values must be replaced with some other value by using the df.fillna function.

From the above figure, 8 out of 9 columns (completion\_year, monthly\_rent, rooms, parking, bathroom, furnished, facilities and additional \_facilities) I will fill the null values with the df.fillna().mode() function. This function will use the mode of the column to fill or replace the null values in the very same column.

**df.drop()**

Ads\_id and prop\_name columns are not usable, so both columns is dropped using the df.drop function.

**monthly\_rent**

Since I am going to do a Classification model, I will need to classify the rental data as it is a continuous number. If I am to do a Regression model, this is not needed (for me to note).

Since there is no guidelines to classify the monthly rental, I would have to deduce it myself. As rental in real estate can be a subjective matter (as it depends on various factors), I would classify them into 3 categories that is low/medium/high.

When it comes to paying for a rental, it will be based on your salary, whether can you afford to pay for it or not? So, such classification would be based on the salary range of 3 main categories in Malaysia that is, B40/M40/T20. I will use the range of salary as the guide to classify the rental. This will take some research.

According to DOSM, in 2022, the income across the household groups are as follow:

|  |  |  |
| --- | --- | --- |
| Mean (average) | Median | Threshold |
| B40 – RM3,401  M40 – RM7,971  T20 – RM19,752 | B40 – RM3,440  M40 – RM7,694  T20 – RM15,867. | B40 – RM5,250 and below  M40 – RM 5250 – RM11,819  T20 – RM11,820 and above |

According to Speedhome, there is no strict rule on how much of your income should go to rent as it is all depends on your neighborhood, your place of work, and how much you earn. However, experts have advised to not surpass 30% of your income when spending on a rental.

So, from the threshold, I will take 30% as the mark-up. Below is the simple calculation to do the classifying of rental:

|  |  |  |
| --- | --- | --- |
| b40 | m40 | t20 |
| RM5,250 | RM11,819 | RM19,752 |
| X 30% | X 30% | X 30% |
| RM1,575.00 | RM3,545.70 | RM5,926 |
| 1600 | 3500 | 6000 |

From the above calculation, for B40 and M40 I used the highest amount of the threshold, RM5,520 and RM11,819 respectively. However, for T20 I used the mean as it has a higher amount than its median and also as T20 income starts from 11.8k and above to which there is no limit.

Finally, this new range would be the threshold for the classification. It will be classified as follows:

Low – From RM1600 and below

Medium – Between RM1600 to RM5999

High – From RM6000 and above

For this column, firstly I will have to extract the numbers only using the function **df.str.extract.** This will create a new column (monthly\_rent\_price).Then Isorted the values from lowest to highest using the **df.sort\_values** function as the values are now a series. The output showed that there are some rows with 6-digits figures which is inappropriate to be considered as rental. Therefore, I have to replace the values in these cells using **df.loc** and **np.random.randint** function. But before that, I used the **df.filter** function to take a deeper look at the column like how many rows have at least 6 digits.

After cleaning this particular column, I classified the level of rental based on the range I just formulated using **pd.cut()** function as it is better for a large dataset.

**facilities and additional\_facilities**

For these columns, it has multiple values in a single cell. The values are separated into individual columns by creating a new function with 0 and 1 as the value assigned. 0 represents False and 1 represents True.

**size**

The size column has a mixture of numerical and text, the same method of classifying the monthly\_rent column (**df.str.extract. and pd,cut()**) is used for fixing this column. Since this project will be use a Classification model, the size was categorized it by the interval of 250sf (0-250, 251 -500…until 2,500 sf)

**rooms**

During the visualization of the data, there was a string ‘More than 10’. The value of this cell(s) were replaced by randomly generated values using **np.random.choice** for which Python can choose from the common number of rooms (1,2,3,4).

There are 10 rooms in total, but for a high-rise property with more than 6 rooms can be quite non-sense. So, cells with more than 6 rooms were replaced from randomly selected values (1,2,3,4,5,6).

Why at least 6 rooms? Because the maximum number of rooms we have seen so far usually is 4 rooms, same goes to bathroom. However, owner can renovate their own unit by partitioning the living room into 1 or 2 rooms.

**bathroom**

There are 8 bathrooms in total, but for a high-rise property with more than 4 bathrooms can be quite non-sense. So, cells with more than 4 bathrooms were replaced from randomly selected values (1,2,3,4).

Why 4 bathrooms? The maximum number of bathrooms usually is 4.

**parking**

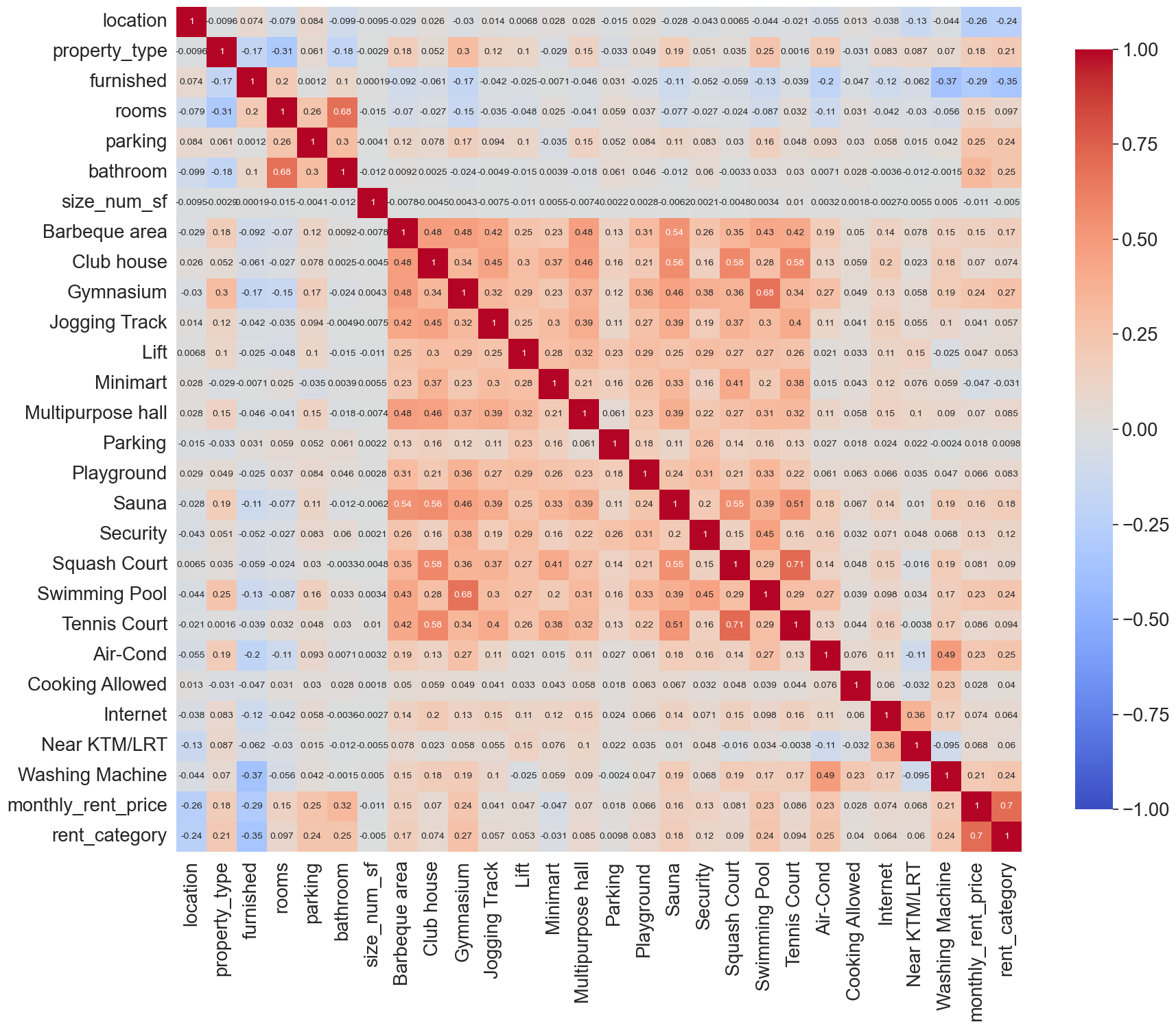
There are total of 10 parking, but for a high-rise property each unit will be allocated minimum of 1 or 2 car parks. But for bigger units may have more allocated car parks, usually 3 or 4 car parks. So, cells with more than 4 car parks were replaced from randomly selected values (1,2,3,4).

**property\_type**

During the visualization of the data, there were some other redundant types of properties and also other labels (Others', 'Bungalow House', 'Houses', 'Soho', 'Residential', 'Condo / Services residence / Penthouse / Townhouse'). These values were replaced with existing common labels ('Condominium', 'Apartment', 'Service Residence', 'Studio', 'Flat', 'Duplex') using **np.random.choice.**

* 1. **Data Visualization**
  2. **Heatmap**

Heatmaps are a valuable tool for EDA, it helps in getting a quick overview of the dataset and identifying potential correlations/relationships/trends between variables.

****

Heatmap of target variable with other variables

**Correlation Coefficient Interpretation:**

* **+1** indicates a perfect positive correlation.
* **-1** indicates a perfect negative correlation.
* **0** indicates no correlation.

#### Interpretation

\*Parking is with or without parking \*\*parking is number of parking

#### Correlations with rent\_category:

##### *Correlations:*

* monthly\_rent\_price (0.70): Strongly correlated with rent\_category, as expected since they are directly related
* size\_bins (0.3): Moderately correlated.
* property\_type (0.21): Slightly correlated.
* bathroom (0.25): Slightly correlated.
* rooms (0.097): Weak negative correlation.
* parking (0.24): Slightly correlated.

##### *Correlations:*

* furnished (-0.35): Weak negative correlation.
* location (-0.24): Weak negative correlation.

#### feature Correlations:

###### *Positive Correlations:*

* Barbeque area and Gymnasium (0.48)
* Air-cond and Washing Machine (0.49)
* Gymnasium and Swimming pool (0.68)
* rooms and bathroom (0.68)
* Sauna and Tennis court (0.51)
* Squash Court and Swimming Pool (0.71)
* Swimming Pool and Tennis Court (0.59)

##### *Negative Correlations:*

* Air-cond and MRT/LRT (-0.11)

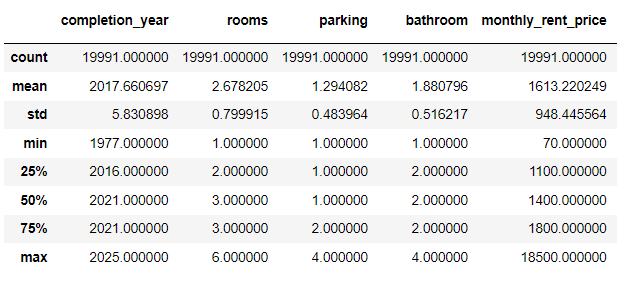
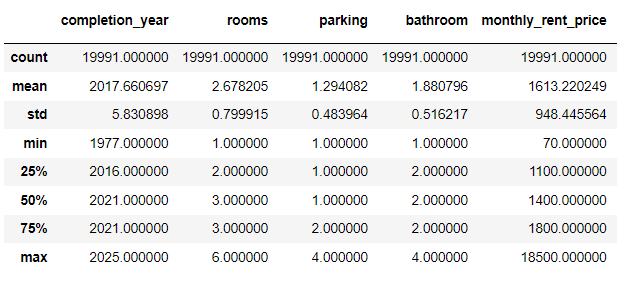
#### Interpreting Specific Features:

* Property\_type: Positively correlated with Gymnasium (0.30 and swimming pool (0.25), but negatively correlated with furnished (-0.17), bathroom (-0.18) and size\_bins (-0.11). Suggesting properties of a certain type (e.g., apartments) might be more likely to have amenities like gyms or swimming pools. Negative correlation, there might be a connection between property type and features like furnished status (-0.17), number of bathrooms (-0.18), or size (-0.11). For instance, apartments might tend to be less furnished, have fewer bathrooms, or be smaller than houses on average.
* Rooms: Positively correlated with size\_bins (0.54), bathroms(0.68), parking(0.25), but negatively correlated with property type (-0.31). For Positive correlations, as expected, there's a strong positive correlation between the number of rooms and the size of the property. Properties with more rooms are also likely to have more bathrooms and parking availability. The negative correlation with property type suggests that properties with more rooms might be less common in certain categories (e.g., apartments) compared to others (e.g., studio).
* Parking: Positively correlated with bathroom (0.30), rooms (0.25), and size\_bins( 0.29) suggesting properties with parking also have more bathrooms and rooms. There's also a positive correlation with property size (0.29).
* Lift: Positively correlated with Gymnasium (0.29), Multipurpose hall (0.32), and Security (0.29), indicating a pattern where properties with lifts tend to have these amenities.
* Furnished: Positively correlated with bathroom 0.10 and rooms (0.20). Although weak, there's a positive correlation between furnished properties and the number of bathrooms (0.10) and rooms (0.20). This might indicate that furnished properties tend to be larger or more luxurious.
* Location: Negatively correlated with size\_bins (0.10) and bathroom (0.10) suggests that there might be a trend of smaller properties with fewer bathrooms in certain locations compared to others.

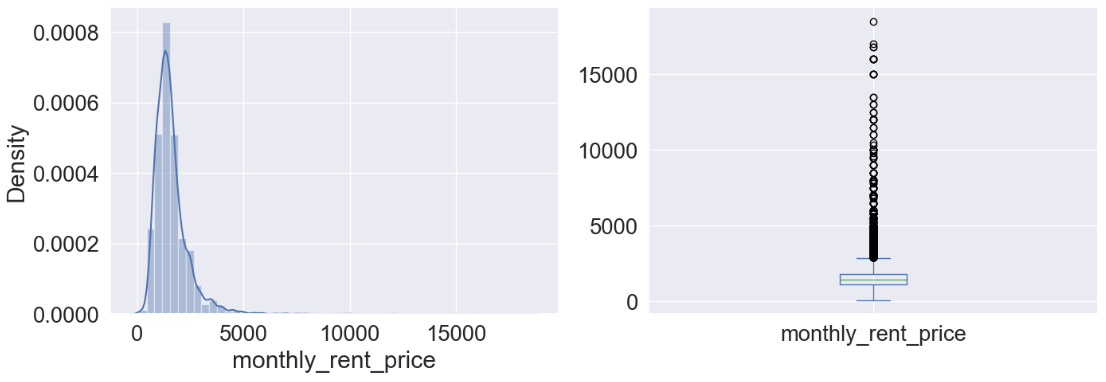
#### Overall: Target Variable (rent\_category)

* The most significant features for predicting rent\_category are size\_bins followed by bathroom, parking, property\_type and rooms.
* The weak correlation with other features suggests that rent\_category is not strongly influenced by individual amenities but by a combination of various features.

**3.3 Outliers**

****

Based on the description, the minimum value is 70 and max value is 18500 and the median is 1400. This means outliers are present in the dataset.

****

Based on the summary statistics provided for monthly\_rent\_price, we can assess the presence of potential outliers, particularly by examining the minimum and maximum values alongside the interquartile range (IQR).

Here's how we can approach identifying outliers:

1. Interquartile Range (IQR):
   * Calculate the IQR, which is the difference between the 75th percentile (Q3) and the 25th percentile (Q1).

IQR=Q3−Q1=1800−1100=700

1. Outlier Detection:
   * Lower Bound: Calculate the lower bound for outliers using the formula Q1−1.5×IQR

Lower Bound=1100−1.5×700=1100−1050=50

* + Upper Bound: Calculate the upper bound for outliers using the formula Q3+1.5×IQRQ

Upper Bound=1800+1.5×700=1800+1050=2850

1. Identifying Outliers:
   * Any value below the lower bound or above the upper bound can be considered a potential outlier.
2. Comparison:
   * Minimum Value: 70 is below the lower bound of 50, suggesting it might be an outlier.
   * Maximum Value: 18,500 is well above the upper bound of 2850, strongly indicating it as an outlier.

Therefore, based on the IQR method with a 1.5 multiplier (a common threshold for outlier detection):

* The minimum value of 70 appears to be an outlier (since it's below the lower bound).
* The maximum value of 18,500 is definitely an outlier (since it's far above the upper bound).

These outliers could potentially impact the models during the fitting since the upper bound is 2850. I did try to fit the models with and without the outliers. In models without the outlier, it removed the ‘High’ category. Hence, the outlier should be kept in this project.

1. **Model the Data**

There are four (4) models used in this project, namely;

* + - 1. Logistic Regression,
      2. Decision Tree,
      3. Random Forest and
      4. Gradient Boosting.

Each model was fitted into three (3) different ways of training;

* + - 1. by default (no parameter),

To see how well the data perform without any parameters by default

* + - 1. with parameters and

To see how well the data perform with specific parameters (adopted by other projects)

* + - 1. with GridSearchCV (hyperparameter)

To see if the performance can be improved by tuning the model with ranges of hyperparameters and from there get the best estimator/parameter.

* 1. **Models**
     1. **Logistic Regression**

Logistic regression is a supervised machine learning algorithm used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not (GeeksforGeeks, 2004). This model can handle both binary or multiclass classification be it binominal, multinomial and ordinal (Gustavo, 2019 and GeeksforGeeks, 2024). Since the target variable (rent\_category) for this project is in the form of ordinal data, we can use this model for the training and testing. However, the data is an imbalanced dataset which may not be suitable to fit into this model, so, a “weighted logistic regression” will be used to address this issue by assigning different weights to each class based on their prevalence in the dataset (GeeksforGeeks, 2024).

* + 1. **Decision Tree**

A decision tree is a flowchart-like structure used to make decisions or predictions (GeeksforGeeks, 2024) and is able to handle both numerical and categorical data (ScikitLearn). Decision trees seem to perform pretty well with imbalanced datasets (Numal Jayawardena, 2020). However, decision trees are prone to overfitting the training data and therefore it is recommended to tune it with hyperparameters (GeeskForGeeks, 2024) while improve its performance.

* + 1. **Random Forest**

Random Forest is an ensemble learning technique that combines the predictions from multiple models to create a more accurate and stable prediction. It is a type of supervised learning algorithm that can be used for both classification and regression tasks. It is effective in handling imbalanced classification problems (GeekforGeeks, 2024) and large datasets that have many attributes (GeeksforGeeks, 2023). It is more accurate than the decision tree algorithm and it can handle missing data, outliers, and noisy features. To increase the accuracy of the training model, it requires the use of the hyperparameter.

* + 1. **Gradient Boosting**

Gradient Boosting is also one of the Ensemble Learning methods. It is a special type of Ensemble Learning technique that works by combining several weak learners (predictors with poor accuracy) into a strong learner (a model with strong accuracy) ([Vagif Aliyev](https://vagifaliyev.medium.com/?source=post_page-----60cc980eeb3d--------------------------------), 2020). It can be used for both classification and regression tasks.

**4.2 Metrics for Training and Testing Score**

The metrics used in these models are as follows:

* **Train Score:** This represents the model's accuracy on the data it was trained on.
* **Test Score:** This represents the model's accuracy on a separate dataset it wasn't trained on. This is a more realistic measure of how well the model will generalize to unseen data.
* **Cross-Validation Score:** This score is usually an average of the model's performance on multiple splits of the data. It's another way to estimate the model's generalizability.
  1. **Training and Testing**
  2. **Logistic Regression**

LR 1 (no parameter)

train score : .7948349174587294

test score : 0.7911977994498625

cross\_val\_score : 0.7944590056187432

LR 2 (with parameter)

train score : 0.7278014007003502

test score : 0.7229307326831708

cross\_val\_score : 0.7269258040886369

LR 3 (with GridSearchCV)

train score : 0.7946473236618309

test score : 0.7909477369342336

cross\_val\_score : 0.7945215447056724

* The train, test, and cross-validation scores are very close, indicating good generalization and minimal overfitting or underfitting.
* The model is stable and performs consistently across different datasets.
  1. **Decisions Trees**

DT 1 (no parameter)

train score : 0.9869934967483742

test score : 0.8414603650912729

cross\_val\_score : 0.8423591174618554

* This model achieves a very high training score (almost 0.99) but a significantly lower test score (around 0.84) and cross-validation score.
* This large gap suggests significant overfitting.

DT 2 (with parameter)

train score : 0.7718859429714857

test score : 0.7699424856214053

cross\_val\_score : 0.7718859923588536

* This model with specific parameters has lower scores across the board compared to DT 1 and DT 3.
* It might be underfitting due to restrictive parameters.

DT 3 (with GridSearchCV)

train score : 0.8135317658829415

test score : 0.8064516129032258

cross\_val\_score : 0.8063414712344636

* This model uses GridSearchCV for hyperparameter tuning and achieves a balance between training and test score (around 0.8) with decent cross-validation performance.

* 1. **Random Forest**

RF 1 (no parameter)

train score : 0.9869934967483742

test score : 0.8757189297324331

cross\_val\_score : 0.87637602119643

* Similar to Decision Trees (DT 1), RF 1 (no parameter) exhibits a significant gap between training and test scores, indicating overfitting.

RF 2 (with parameter)

train score : 0.8609929964982491

test score : 0.835458864716179

cross\_val\_score : 0.8433591172663597

* The chosen hyperparameters for this model might be too restrictive, hindering its ability to learn complex patterns from the data.
* This can lead to underfitting

RF 3 (with GridSearchCV)

train score : 0.8578039019509754

test score : 0.8377094273568392

cross\_val\_score : 0.8411077492360516

* This model achieves a balance between training and test scores (around 0.837-0.857) with decent cross-validation performance, demonstrating the benefit of hyperparameter tuning.
* The hyperparameters configuration helps the model learn effectively without overfitting significantly.
  1. **Gradient Boosting**

GB 1 (no parameter)

train score : 0.8587418709354677

test score : 0.8459614903725932

cross\_val\_score : 0.847361169189637

* The train, test, and cross-validation scores are very close, indicating good generalization and minimal overfitting or underfitting.
* The model is stable and performs consistently across different datasets.

GB 2 (with parameter)

train score : 0.9048274137068534

test score : 0.870717679419855

cross\_val\_score : 0.8746873876510424

* The highest train score suggests the model fits the training data very well, but there's a risk of overfitting.
* The test and cross-validation scores are also higher compared to other models, indicating strong performance even on unseen data.

GB 3 (with GridSearchCV)

train score : 0.9683591795897949

test score : 0.889472368092023

cross\_val\_score : 0.8896325677133703

* The training score (0.97) is significantly higher than previous models, suggesting substantial overfitting. The model might be learning training data specifics too well and may not generalize well to unseen data.
* The test score (0.89) and cross-validation score (0.89) are similar, indicating potential for reasonable generalizability despite the overfitting.

**Overall:**

* **Gradient Boosting:** Achieves the highest test score (0.8894) and cross-validation score (0.8896), indicating it might be the best performing model overall.
* **Random Forest:** Follows closely with a test score of 0.8757 and a cross-validation score of 0.8763.
* **Logistic Regression:** Has the lowest test score (0.7911) and cross-validation score (0.7944), suggesting it might not be the best choice for this project.
* **Decision Tree:** Performs slightly better than Logistic Regression, but still lower than Random Forest and Gradient Boosting

1. **Evaluate the Model**

The evaluation of all models will be using the Confusion Matrix, Classification Report, F1 score, Matthews Correlation Coefficient, ROC AUC.

**5.1 Metrics**

**a. Confusion Matrix**

A confusion matrix is easily the most popular method of visualizing the quality of classification models. A table that summarizes the performance of a classification model by comparing its predicted labels to the true labels. It displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) of the model's predictions. It is well usable for both binary and multiclass problems (EvidentlyAI). Below is the meaning of each component in the confusion matrix;

* **True Positives (TP)** - The instances where the model correctly predicts the positive class (Spam email) as positive.
* **False Positives (FP)** - The instances where the model incorrectly predicts the negative class (Not a spam email) as positive.
* **True Negatives (TN**) - The instances where the model correctly predicts the negative class as negative.
* **False Negatives (FN)** - The instances where the model incorrectly predicts the positive class as negative.

**b. Classification Report** (precision, recall, and F1-score for each class along with macro and weighted averages)

A classification report is a summary of the key metrics derived from a confusion matrix. A classification report is a text summary that shows the main metrics for each class. It usually includes the precision, recall, F1-score, and support for each class, as well as the weighted average of these metrics across all classes. A classification report gives a detailed breakdown of how well the model performs on each class, and how it balances the trade-off between precision and recall. It also shows the number of instances (support) for each class, which can indicate the class imbalance or the size of the dataset. Below is the meaning of each component in the classification report;

* **Precision:** the number of true positives divided by the sum of true positives and false positives. It measures the accuracy of positive predictions.
* **Recall:** the number of true positives divided by the sum of true positives and false negatives. It measures the completeness of positive predictions.
* **F1-score:** the harmonic mean of precision and recall. It provides a balance between precision and recall.
* **Support:** the number of samples in each class

Generally, higher scores (closer to 1) indicate better performance for a specific metric (precision, recall, F1-score) for a particular class. Lower scores suggest the model struggles with that class.

**c. F1\_score**

The F-1 Score metric is preferable for multi-class classification problem and imbalanced class distribution and to balanced measures between precision and recall (baeldung,2024). Precision and recall are both crucial in different aspects of machine learning. Precision focuses on minimizing false positives, ensuring that the positive predictions made by the model are accurate. On the other hand, recall aims to minimize false negatives, ensuring that the model identifies all positive observations correctly.

An ideal machine learning model should have both high precision and high recall. However, there is often a trade-off between these two measures. As precision increases, recall tends to decrease, and vice versa. The F1 score takes this trade-off into account and provides a single metric to evaluate the model’s overall performance. However, due to imbalanced dataset, it needs to be weighted by “average = weighted”.

The core range from 0 to 1, given a higher score (closer to 1) indicates better model performance in terms of balancing precision and recall and lower score (closer to 0) indicates a model struggling to correctly classify instances, with potential issues in either precision or recall or both.

**d. Matthews’ Correlation Coefficient (MCC)**

The Matthews correlation coefficient is used in machine learning as a measure of the quality of binary and multiclass classifications. It takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes (ScikitLearn). Generally, an MCC score closer to 1 indicates better classifier performance, while a score closer to -1 suggests poor performance. A score of 0 implies that the classifier is performing no better than random chance. In practice, an MCC score above 0.3 is considered moderate, and a score above 0.5 is considered strong(activeloop.ai).

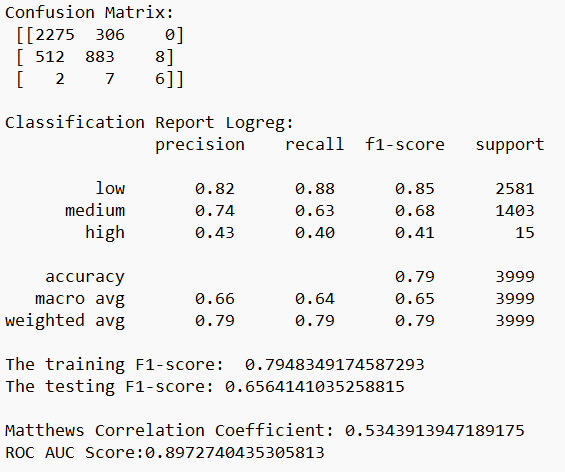
**e. ROC AUC score**

ROC AUC means Receiver Operating Characteristic Area Under the Curve, it begins with the confusion matrix, a foundational tool to assess classification model performance (MisunSong, 2023). It also can be used with binary, multiclass and multilabel classification (ScikitLearn). It uses components or metrics from the Confusion Matrix to derive the model's performance. In summary they show us the separability of the classes by all possible thresholds, or in other words, how well the model is classifying each class ([Vinícius Trevisan](https://medium.com/@vinicius_trevisan?source=post_page-----294fd4617e3a--------------------------------), 2022). In imbalanced data, a model might achieve high accuracy simply by predicting the majority class all the time. However, ROC AUC considers the model's performance across all thresholds, providing a more robust evaluation (developers.google).

The score is usually from 0 to 1 (Perfect Score), which represents an ideal scenario where the model perfectly distinguishes between positive and negative classes. The ROC curve would follow the left border and top of the ROC graph. Between 0 to 0.5 (Random Guessing), indicates the model performs no better than random chance.

**5.2 Evaluation**

* 1. **Logistic Regression**



**Logistic Regression without paramater (1) Evaluation Result**

**Class-Level Performance (Confusion Matrix & Classification Report):**

* **High Imbalance:** The confusion matrix highlights a significant class imbalance, with the "low" class having the most instances (2275) and the "high" class having the fewest (15).
* **Dominant Class:** The model performs well on the majority class ("low") with a precision of 0.82 and recall of 0.88. However, the performance drops for the minority classes ("medium" and "high").

**F1-Score:**

* **Training vs Testing:** The training F1-score (0.7948) is higher than the testing F1-score (0.6564), suggesting potential overfitting. The model might be memorizing the training data and not generalizing well to unseen examples.
* **Interpretation:** Despite the accuracy scores, the F1-score emphasizes the class imbalance. A lower F1-score for minority classes indicates the model struggles to accurately classify them.

**Matthews Correlation Coefficient (MCC):**

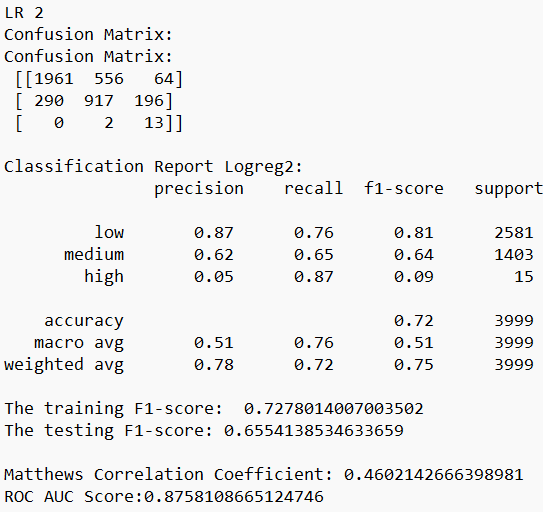
* **Moderate Performance:** The MCC of 0.53 suggests a moderate relationship between the true and predicted classifications. While not a perfect score, it aligns with the observations from accuracy and F1-score.

**ROC AUC Score:**

* **Good Discrimination:** The high ROC AUC score (0.897) indicates good overall discrimination between rent categories. The model can effectively rank instances based on their likelihood of belonging to a particular category.

**Overall Observations:**

* The model shows promise in classifying rent categories with decent overall accuracy and good discrimination ability (ROC AUC).
* However, the class imbalance significantly impacts performance on minority classes ("medium" and "high"). The model struggles to accurately classify them despite good performance on the majority class ("low").
* The overfitting evident from the F1-score difference suggests the model might benefit from regularization techniques to improve generalization.

****

**Logistic Regression -with parameters (2) Evaluation Result**

**Class-Level Performance:**

* **Improved Performance on Medium Class:** The precision and recall for the "medium" class have improved in LR2 compared to LR1. This indicates better handling of this class.
* **Significant Drop for High Class:** However, the performance on the "high" class has significantly declined. LR2 achieves high recall (87%) but very low precision (5%) for this class. This means it might be classifying many other instances as "high" even though they don't belong there (high false positives).

**F1-Score:**

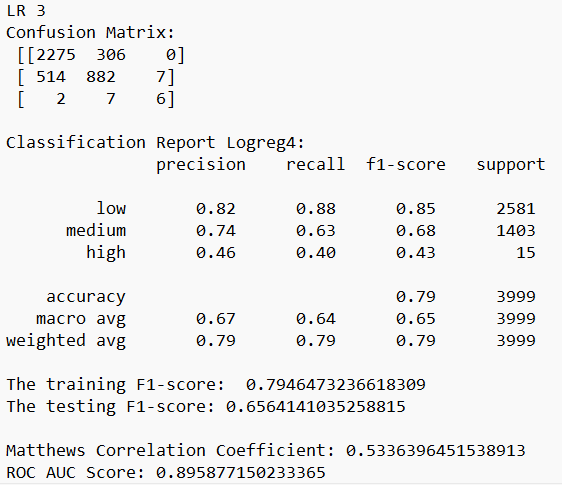
* **Similar Training vs Testing F1:** The training and testing F1-scores are closer in LR2 compared to LR1, suggesting less overfitting on the training data. However, the overall F1-score remains similar (around 0.65) due to the significant drop in the "high" class.

**MCC and ROC AUC:**

* **Decreased MCC:** The MCC score (0.46) is lower in LR2 compared to LR1 (0.53), indicating a weaker relationship between true and predicted classifications.
* **Slightly Lower ROC AUC:** The ROC AUC score (0.87) is slightly lower in LR2 compared to LR1 (0.89), suggesting a minor decrease in the model's ability to discriminate between rent categories.

**Overall Observations:**

* LR2 seems to have addressed the issue of overfitting on the majority class ("low") seen in LR1.

****

**Logistic Regression – with GridSearchCV (3) Evaluation Result**

Logistic Regression model 3 (LR3) reveals very similar performance to LR1:

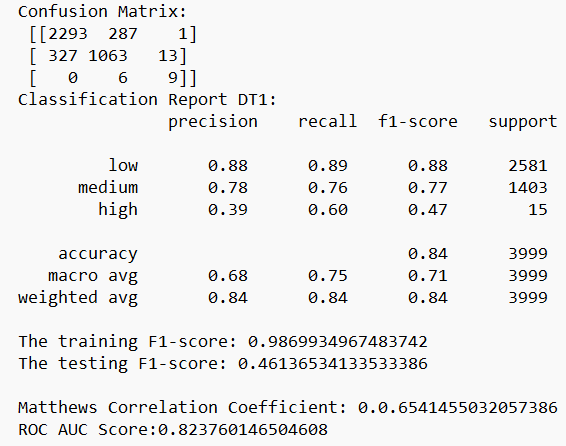
* **Accuracy:** The overall accuracy (79%), training F1-score (0.7946), testing F1-score (0.6564), MCC (0.5336), and ROC AUC score (0.8958) are all very close to the values observed in LR1.
* **Confusion Matrix and Classification Report:** The confusion matrix and classification report also show a very similar pattern to LR1, with good performance on the majority class ("low"), lower performance on the minority classes ("medium" and "high"), and a significant class imbalance.

**Interpretation:**

LR3 appears to be very similar to LR1, suggesting the changes made might not have had a significant impact on the model's performance. It's possible the modifications you made in LR3 didn't address the core issues of class imbalance and potential overfitting on the training data.

* 1. **Decision Tree**

**DT1**

****

**Decision Tree - without paramater (1) Evaluation Result**

**Overall Performance:**

* **High Training Score:** The training score of 0.98 suggests the model perfectly classifies the training data.
* **Large Gap Between Training and Test Score:** However, the test score (0.84) and cross-validation score (0.84) are significantly lower, indicating severe overfitting.

**Class-Level Performance (Confusion Matrix & Classification Report):**

* **Decent Performance on Majority Class:** The model performs well on the majority class ("low") with a precision of 0.88 and recall of 0.89.
* **Struggles with Minority Classes:** Similar to Logistic Regression models, DT1 struggles with the minority classes ("medium" and "high"). It has lower precision and recall for these classes.

**F1-Score:**

* **Huge Discrepancy:** The training F1-score (0.98) is much higher than the testing F1-score (0.46), highlighting the issue of overfitting. The model memorizes the training data and performs poorly on unseen examples.

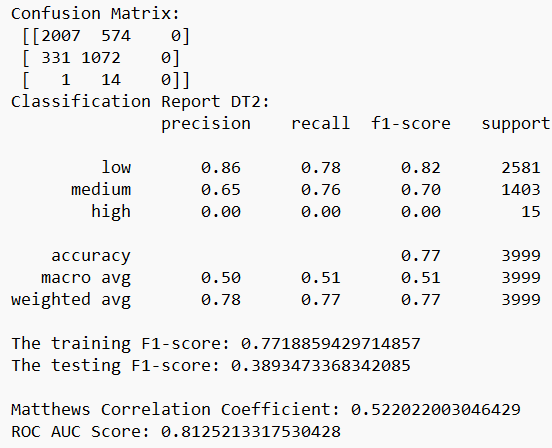
**MCC and ROC AUC:**

* **Moderate Performance:** The MCC (0.65) and ROC AUC score (0.82) suggest moderate overall performance, but the overfitting issue significantly impacts the model's reliability.

**Overall Observations:**

* DT1 achieves high accuracy on the training data but suffers from severe overfitting. It memorizes training data specifics and performs poorly on unseen data.
* The model struggles with minority classes, similar to the Logistic Regression models.

**DT 2**

****

**Decision Tree -with parameters (2) Evaluation Result**

**Overfitting Addressed:**

* **Reduced Training Score:** The training score (0.77) is significantly lower than DT1 (0.98), indicating successful reduction in overfitting. The model is less likely to memorize training data specifics.
* **Similar Test and CV Scores:** The test score (0.77) and cross-validation score (0.77) are close, suggesting the model generalizes better to unseen data compared to DT1.

**Class-Level Performance:**

* **Improved Performance on Medium Class:** Precision and recall for the "medium" class have improved in DT2 compared to DT1.
* **Complete Miss on High Class:** However, DT2 completely misses all instances of the "high" class, resulting in a precision and recall of 0 for this class.

**F1-Score:**

* **Lower Overall F1:** The training and testing F1-scores (around 0.77 and 0.39, respectively) are lower than DT1. This reflects the trade-off made to address overfitting and the missed "high" class.

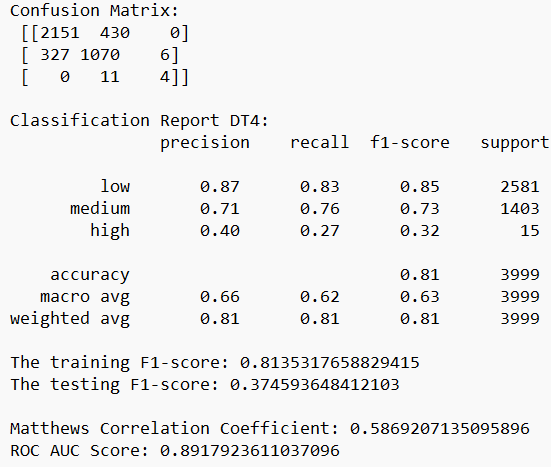
**MCC and ROC AUC:**

* **Moderate Performance:** The MCC (0.52) and ROC AUC score (0.81) suggest moderate overall performance, similar to DT1. However, the missed "high" class significantly impacts the model's ability to correctly classify all categories.

**Overall Observations:**

* DT2 successfully addresses the overfitting issue in DT1. It generalizes better to unseen data.
* However, the changes made resulted in the model completely missing all instances of the "high" class, which is a critical issue.

**DT 3**

****

**Decision Tree – with GridSearchCV (3) Evaluation Result**

**Overfitting Addressed:**

* **Reduced Training Score:** The training score (0.81) is lower than DT1 (0.98) and indicates successful control of overfitting. The model is less likely to overfit to the training data.
* **Similar Test and CV Scores:** The test score (0.81) and cross-validation score (0.81) are close, suggesting the model generalizes reasonably well to unseen data.

**Class-Level Performance:**

* **Improved Performance on Medium Class:** Precision and recall for the "medium" class have slightly improved compared to DT1 and remained similar to DT2.
* **Partial Improvement on High Class:** While still not perfect, DT3 shows some improvement on the "high" class compared to DT2. It achieves a precision of 0.40 and recall of 0.27, indicating it's classifying some "high" class instances correctly.

**F1-Score:**

* **Lower Overall F1:** Similar to DT2, the F1-scores (training: 0.81, testing: 0.37) are lower than DT1 due to the trade-off with overfitting and the still-imperfect "high" class performance.

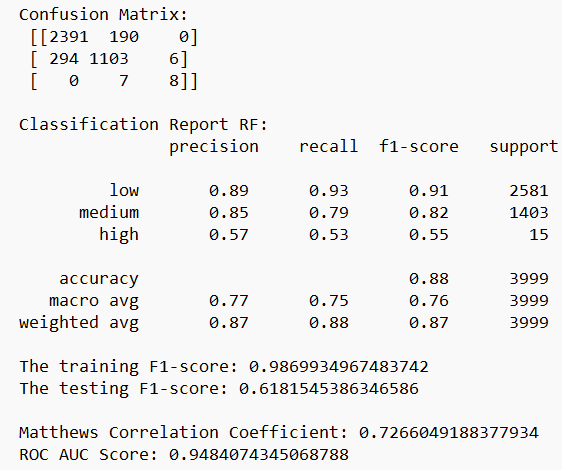
**MCC and ROC AUC:**

* **Improved Performance:** The MCC (0.59) is slightly better than DT1 and DT2, suggesting a stronger correlation between true and predicted classifications. The ROC AUC score (0.89) is also high, indicating good overall discrimination between rent categories.

**Overall Observations:**

* DT3 effectively addresses overfitting while achieving reasonable overall accuracy and good discrimination ability (ROC AUC).
* The model shows improvement in handling the minority "high" class compared to DT2, although there's still room for improvement.
  1. **Random Forest**

**RF 1**

****

**Random Forest - without paramater (1) Evaluation Result**

**Class-Level Performance (Confusion Matrix & Classification Report):**

* **Decent Performance on Majority Class:** The model performs well on the majority class ("low") with a precision of 0.89 and recall of 0.93.
* **Improved Performance on Minority Classes:** Compared to Logistic Regression and Decision Trees, RF1 shows improvement in handling minority classes ("medium" and "high"). It has higher precision and recall for these classes, demonstrating better ability to classify them.

**F1-Score:**

* **High Training F1-Score:** The training F1-score (0.99) is high, but remember it's likely due to overfitting on the training data.
* **Testing F1-Score Better Than Other Models:** The testing F1-score (0.62) is significantly better than Decision Trees (around 0.3-0.4) for minority classes. This indicates some generalization to unseen data.

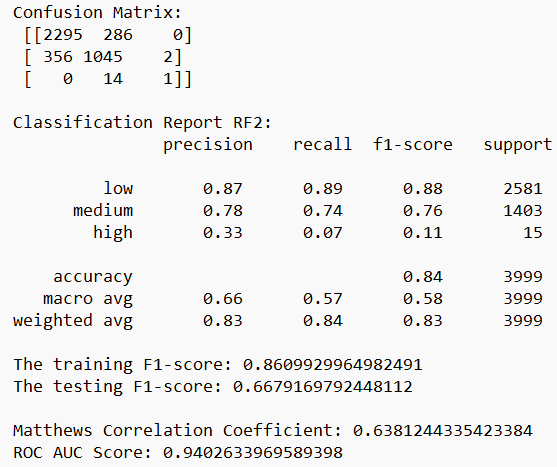
**MCC and ROC AUC:**

* **Moderate Overall Performance:** The MCC (0.73) and ROC AUC score (0.95) suggest moderate overall performance, but it's an improvement over previous models in handling class imbalance.

**Overall Observations:**

* RF1 shows signs of overfitting but less severely compared to Decision Trees.
* It demonstrates a clear advantage in handling class imbalance, achieving better performance on minority classes compared to Logistic Regression and Decision Trees.

**RF 2**

****

**Random Forest -with parameters (2) Evaluation Result**

**Class-Level Performance:**

* **Consistent Performance on Majority Class:** Precision and recall for the "low" class are similar between RF1 and RF2.
* **Mixed Performance on Minority Classes:**
  + There's a slight improvement in precision for the "medium" class in RF2.
  + However, RF2 significantly misses instances of the "high" class, with a precision of 0.33 and recall of 0.07.

**F1-Score:**

* **Lower Overall F1:** The training and testing F1-scores (around 0.86 and 0.67, respectively) are lower than RF1. This reflects the trade-off made to address overfitting and the missed "high" class.

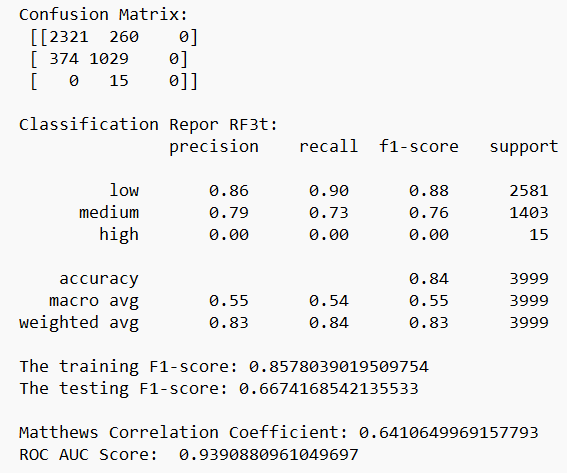
**MCC and ROC AUC:**

* **Moderate Performance:** The MCC (0.64) and ROC AUC score (0.94) are moderately similar to RF1. However, the missed "high" class can negatively impact the model's ability to correctly classify all categories.

**Overall Observations:**

* RF2 addresses the overfitting issue in RF1, leading to better generalization.
* However, the changes made resulted in the model missing a significant portion of the "high" class instances.

**RF 3**

****

**Random Forest – with GridSearchCV (3) Evaluation Result**

**Class-Level Performance:**

* **Consistent Performance on Majority Class:** Similar to RF1 and RF2, precision and recall for the "low" class remain consistent.
* **Mixed Performance on Minority Classes:**
  + There's a slight improvement in precision for the "medium" class compared to RF1.
  + However, similar to RF2, RF3 completely misses all instances of the "high" class, resulting in a precision and recall of 0 for this class.

**F1-Score:**

* **Lower Overall F1:** The training and testing F1-scores (around 0.86 and 0.67, respectively) are lower than RF1. This reflects the trade-off with overfitting and the missed "high" class.

**MCC and ROC AUC:**

* **Moderate Performance:** The MCC (0.64) and ROC AUC score (0.94) are similar to RF1 and RF2. However, the missed "high" class significantly impacts the model's ability to classify all categories accurately.

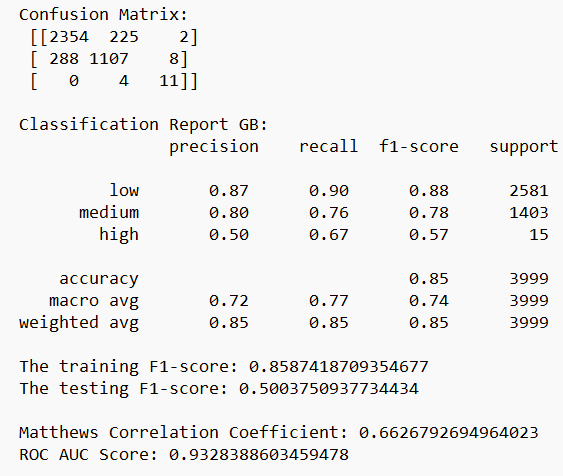
**Overall Observations:**

* RF3, like RF2, addresses overfitting while achieving reasonable overall accuracy and good discrimination ability (ROC AUC).
* Similar to RF2, it suffers from completely missing instances of the "high" class, a crucial issue for your task.

**Comparison with RF2:**

* GridSearchCV in RF3 doesn't seem to have a significant impact compared to the specific hyperparameter choices made in RF2. Both models achieve similar performance.
  1. **Gradient Boosting**

**GB 1**

****

**Gradient Boosting - without paramater (1) Evaluation Result**

**Class-Level Performance:**

* **Consistent Performance on Majority Class:** Similar to other models, precision and recall for the "low" class remain consistent.
* **Improved Performance on Minority Classes:**
  + There's a slight improvement in precision and recall for the "medium" class compared to previous models (except tuned Random Forests).
  + Most importantly, GB1 shows significant improvement in handling the "high" class. It achieves a precision of 0.50 and recall of 0.67, suggesting it's correctly classifying some "high" class instances that previous models missed entirely.

**F1-Score:**

* **Lower Overall F1:** Similar to tuned Random Forests, the training and testing F1-scores (around 0.86 and 0.50, respectively) are lower than models without overfitting control. This reflects the trade-off with overfitting and the still-imperfect "high" class performance.

**MCC and ROC AUC:**

* **Moderate Overall Performance:** The MCC (0.66) and ROC AUC score (0.93) are similar to tuned Random Forests. The improvement in handling the "high" class contributes to a better overall performance compared to untuned models.

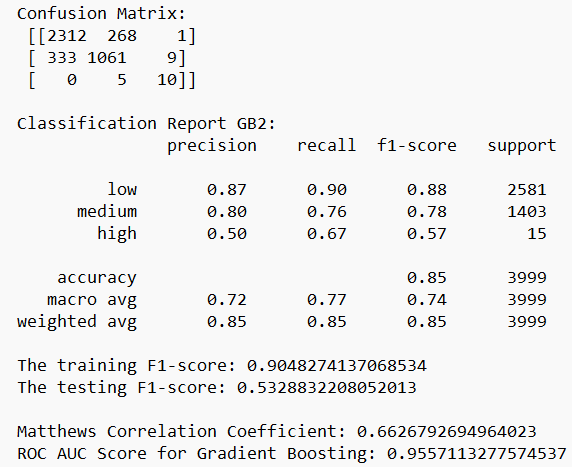
**Overall Observations:**

* GB1 demonstrates a good balance between overfitting control and generalizability.
* It significantly improves upon previous models in handling the minority "high" class, correctly classifying some instances that were previously missed entirely.

**Advantages over Tuned Random Forests (RF2 & RF3):**

* GB1 achieves similar overall performance and generalizability but shows a clear advantage in handling the "high" class.

**GB 2**

****

**Gradient Boosting -with parameters (2) Evaluation Result**

**Class-Level Performance:**

* **Consistent Performance on Majority Class:** Precision and recall for the "low" class are similar between GB1 and GB2.
* **Improved Performance on Minority Classes:**
  + There's a slight improvement in precision and recall for the "medium" class in GB2 compared to GB1.
  + Most importantly, GB2 shows significant improvement in handling the "high" class**.** It achieves a precision of 0.50 and recall of 0.67, suggesting it's correctly classifying some "high" class instances that previous models missed entirely.

**F1-Score:**

* **Improved Overall F1:** The testing F1-score (0.53) is higher than GB1 (0.50). This reflects the improvement in generalizability and "medium" class performance.

**MCC and ROC AUC:**

* **Moderate Overall Performance:** The MCC (0.66) remains similar to GB1 and tuned Random Forests. The ROC AUC score (0.95) is higher than all previously analyzed models, indicating strong overall discrimination ability.

**Overall Observations:**

* GB2 seems to have addressed the overfitting concern to a certain extent, achieving better generalizability (higher test score) than GB1.
* It significantly improves upon previous models in handling the minority "high" class, correctly classifying more instances and achieving a better F1-score.

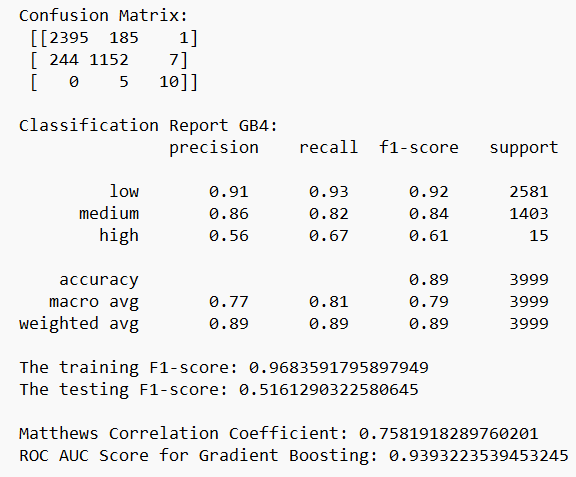
**Compared to GB1:**

* GB2 offers similar or slightly better performance on the majority and "medium" classes.
* It demonstrates a clear advantage in handling the "high" class, which might be crucial depending on your specific needs.

**Recommendation:**

* Given the improvement in generalizability and "high" class performance, GB2 appears to be the strongest model among those explored so far.

**GB 3**

****

**Gradient Boosting – with GridSearchCV (3) Evaluation Result**

**Class-Level Performance:**

* **Improved Performance on Majority Class:** Precision and recall for the "low" class are slightly better in GB3 compared to previous models.
* **Mixed Performance on Minority Classes:**
  + There's a slight improvement in precision and recall for the "medium" class compared to some models (not GB2).
  + GB3's performance on the "high" class is similar to GB1 and GB2. It achieves a precision of 0.56 and recall of 0.67.

**F1-Score:**

* **Lower Overall F1:** The training F1-score (0.97) is high but likely inflated due to overfitting. The testing F1-score (0.52) is similar to other models.

**MCC and ROC AUC:**

* **Moderate Overall Performance:** The MCC (0.76) is the highest among the models analyzed so far. However, this might be partially inflated by the overfitting. The ROC AUC score (0.94) is similar to GB1 and slightly lower than GB2.

**Overall Observations:**

* GB3 achieves good overall accuracy and a high MCC, but the high training score suggests significant overfitting.
* It shows some improvement in handling the "high" class compared to untuned models but performs similarly to GB1 and GB2, which have less overfitting.

**Comparison with GB1 and GB2:**

* GB1 and GB2 seem to offer a better balance between overfitting control and "high" class performance.

**5.3 Model Selection**

From the evaluation of models, in Logistic Regression, the **first model is preferred** for now.

* **Similar Overall Accuracy:** All three models have similar overall accuracy (around 79%).
* **Better Performance on All Classes:** LR1 shows the best balance between precision and recall for all three classes ("low," "medium," and "high"). While it might not be perfect for minority classes, it doesn't have the significant drop in performance for the "high" class seen in LR2.
* **Higher F1-Score:** LR1 has a slightly higher F1-score (both training and testing) compared to LR2 and LR3. F1-score considers both precision and recall, providing a more balanced view of performance.
* **Higher MCC:** The MCC score in LR1 is slightly better than LR2 and LR3, indicating a stronger correlation between true and predicted classifications.

From the Decision Tree models, **DT 3 is preferred.**

* **Addresses Overfitting:** DT3 effectively controls overfitting as seen by the lower training score compared to DT1 and similar test/CV scores.
* **Reasonable Overall Accuracy:** It achieves a good overall accuracy (around 81%) on unseen data.
* **Improved Handling of Minority Class:** While not perfect, DT3 shows some improvement in classifying the "high" class compared to DT1 and DT2. It at least identifies some instances correctly.
* **Good Discrimination Ability:** The high ROC AUC score (0.89) suggests good overall ability to discriminate between rent categories.

From all the Random Forest models, **model 2 is preferred.**

* **Addresses Overfitting:** RF2 successfully reduces overfitting compared to RF1 (evident from the lower training score and similar test/CV scores).
* **Reasonable Overall Accuracy:** It achieves a good overall accuracy (around 83%) on unseen data.
* **Better Generalization:** While not perfect, RF2 shows some improvement in generalizability compared to RF1 (smaller gap between training and test scores).

**Drawbacks of RF1 and RF3:**

* **RF1:** While it performs well on the training data, it suffers from severe overfitting and doesn't generalize well to unseen data.
* **RF3:** Similar to RF2, it addresses overfitting but completely misses all instances of the "high" class. GridSearchCV in RF3 doesn't seem to have a significant advantage over the specific parameter choices in RF2.

**Important Note:**

It's important to acknowledge that even RF2 has limitations in handling the "high" class. It misses a significant portion of these instances.

From all the Gradient Boosting models, **GB3 is preferred.**

**Balance Between Overfitting and Generalizability:**

* GB2 achieves a good training score but not as high as GB3, suggesting less overfitting.
* It has a higher test score compared to GB1 (less-tuned model) and similar to GB3 (potentially overfitted model), indicating better generalizability on unseen data.

**Improved Handling of Minority Class ("High" Rent Category):**

* Similar to GB1, GB2 correctly classifies some instances of the "high" class that previous models missed entirely.
* It achieves a better F1-score on the test set compared to GB1, suggesting a more balanced performance across classes.

**Overall Performance:**

* GB2 offers a reasonable overall accuracy and a moderate MCC score.
* The ROC AUC score for GB2 is the highest among the three models analyzed, indicating strong overall discrimination ability between rent categories.

**Drawbacks of GB1 and GB3:**

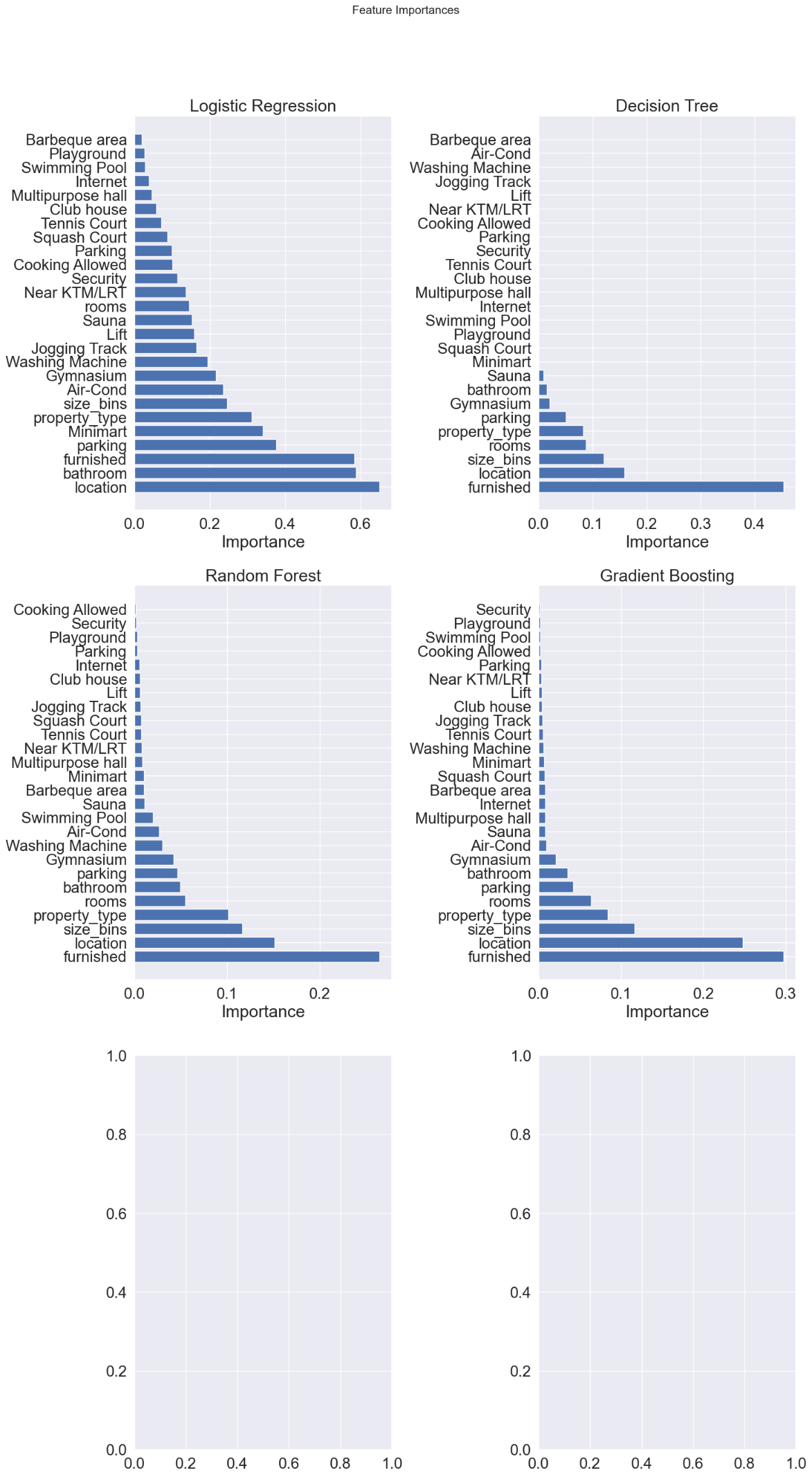
* **GB1:** While it performs well on the "high" class compared to untuned models, it suffers from slightly lower overall accuracy and generalizability.
* **GB3:** Although it achieves high overall accuracy and MCC, the very high training score suggests significant overfitting, which could negatively impact performance on unseen data.
  1. **Feature Importance**

Feature importance refers to techniques that calculate a score for all the input features for a given model. The scores represent the “importance” of each feature. A higher score means that the specific feature will have a larger effect on the model that is being used to predict a certain variable. The Feature Importance is only generated for the selected ones.

**\*\*parking – number of parking**

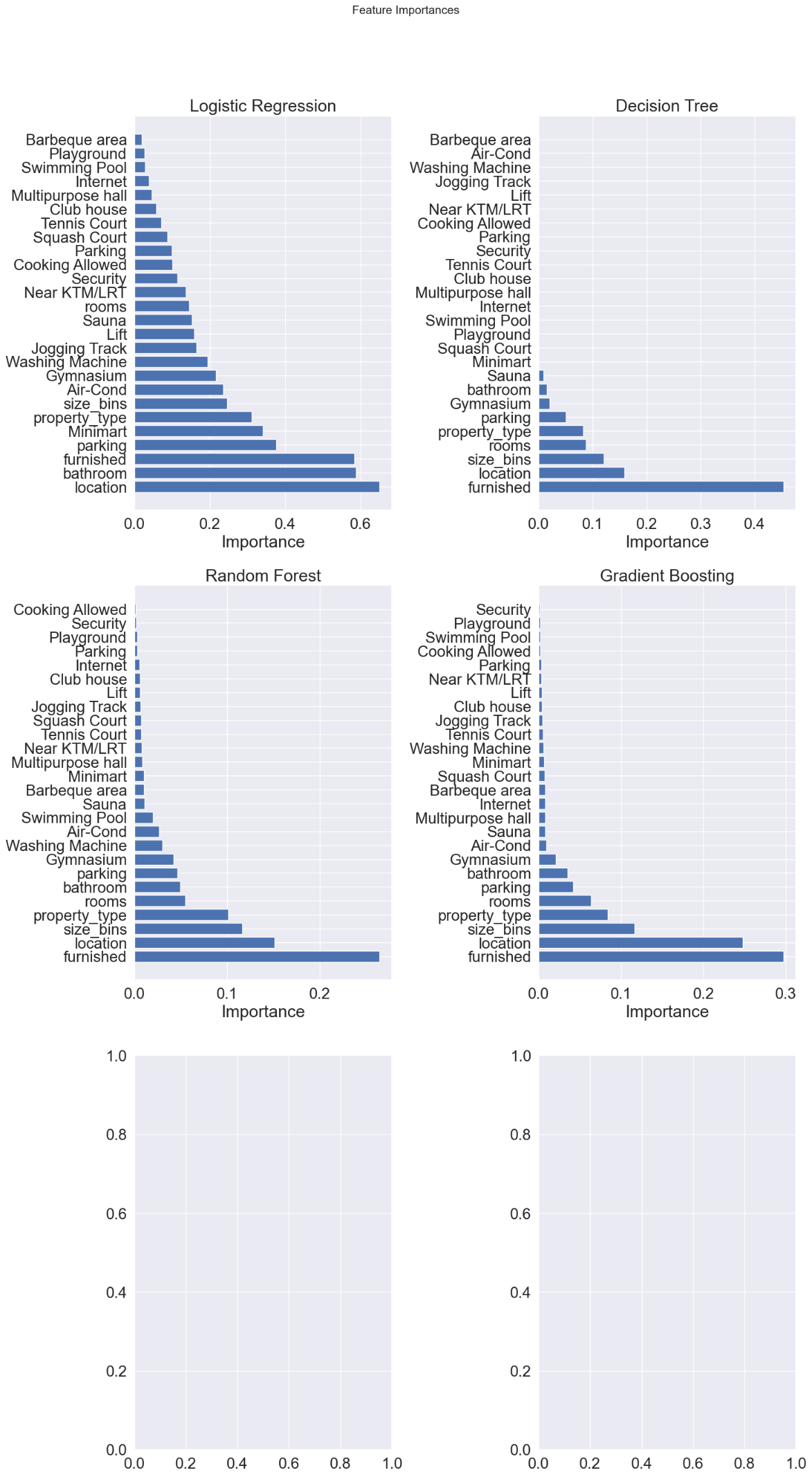
**\*\*Parking – with or without parking**

* **Logistics Regression**

****

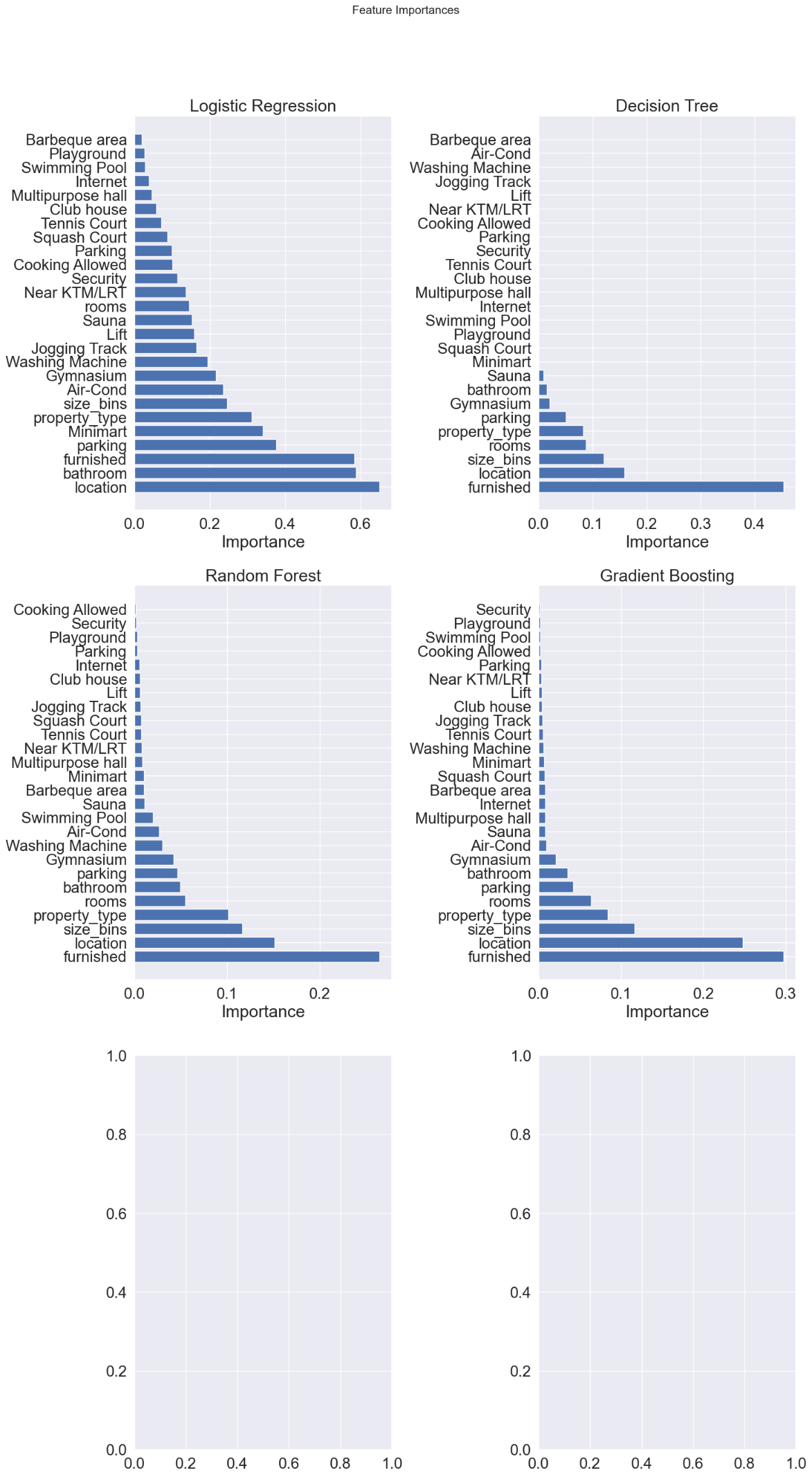
From the graph above, the most significant feature is location, followed by bathroom, furnished, parking (number), property\_type and size\_bins. Top facilities that influence the prediction are Minimart, Air-Cond and Gymnasium.

* **Decision Tree**

****

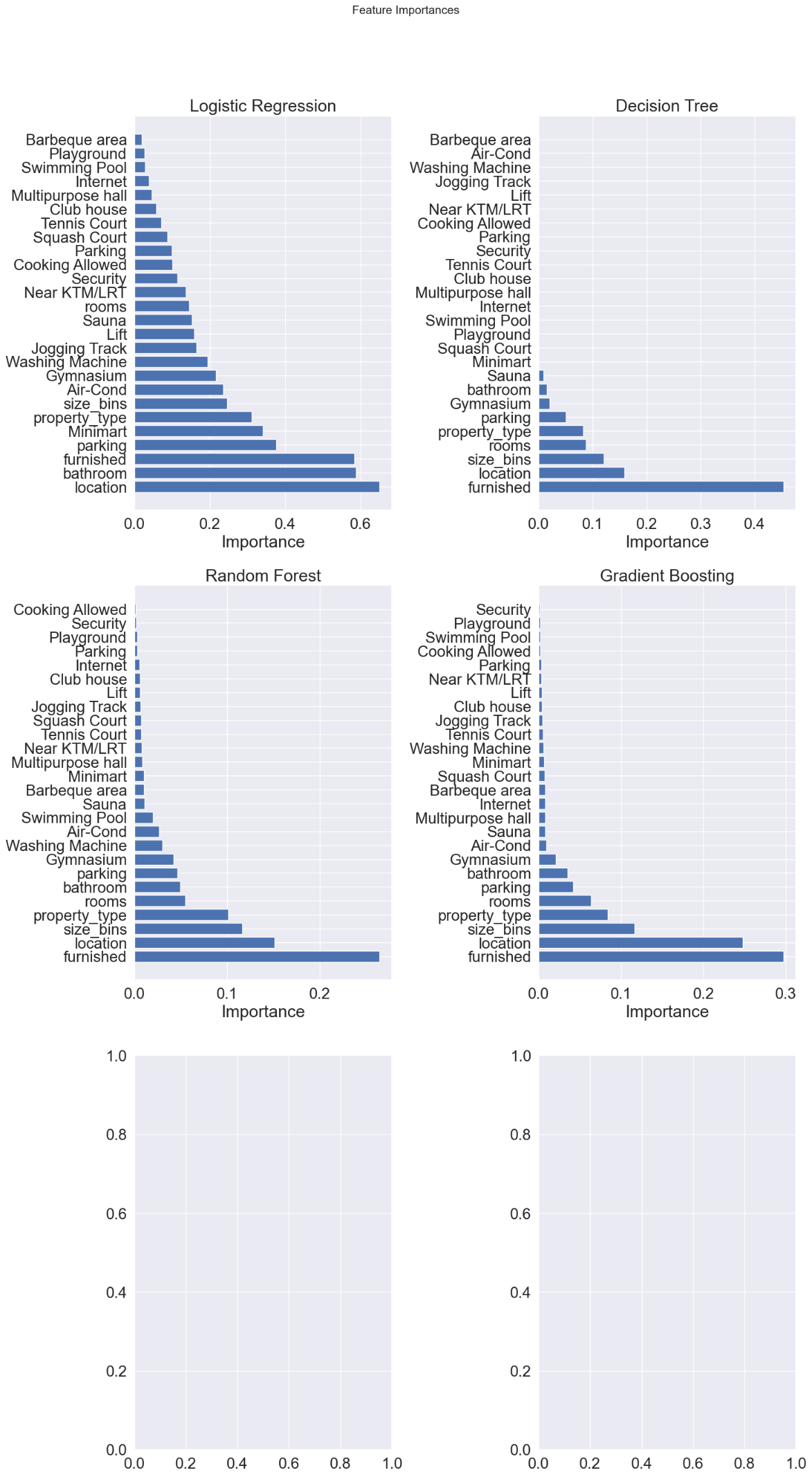
From the graph above, the most significant feature is furnished, followed by location, size\_bins, rooms, property\_type, parking, Gymnasium and bathroom.

* **Random Forest**

****

From the graph above, the most significant feature is furnished, followed by location, size\_bins, property\_type, rooms, bathroom and parking. Top facilities that influence the prediction are Gymnasium, Washing Machine, and Air-Cond.

* **Gradient Boosting**

****

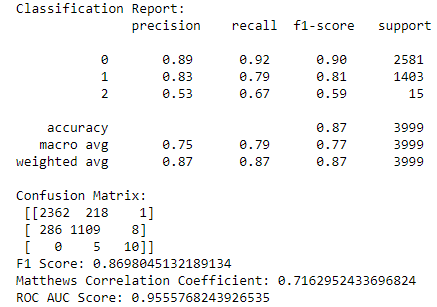
From the graph above, the most significant feature is furnished, followed by location, size\_bins, property\_type, rooms, parking and bathroom. Top facilities that influence the prediction are Gymnasium, Air-Cond and Sauna.

**6 Answer the Problem**

**6.1 The Best Model**

From the 4 selected models, Gradient Boosting model is preferred. GB2 seems to strike a good balance between overfitting control, generalizability, and the ability to handle the critical "high" rent category. It performs well in all classes while maintaining a reasonable overall accuracy.

The model was trained and tested on the entire dataset by default.



Best Model Result

**Classification Report:**

* This table summarizes the performance of the model on a test dataset, broken down by each class.
  + **Precision:** Out of all the units the model predicted as low rent, 89% were truly low rent, 83% for medium and 53% for high rent.
  + **Recall:** Indicates that the model identified 92% of the actual low rent units in the test data, 79% for medium and 67% for high category.
  + **F1-score:** low and medium categories seem to have better overall performance.

**Accuracy:**

* Overall accuracy of the model is 0.87, which means the model predicted the correct class for 87% of the data points in the test set.

**Macro Average and Weighted Average:**

* These are two ways to summarize the overall performance across all classes.
  + **Macro Average:** In this case, the macro average values are lower than the weighted average, likely due to the class imbalance (more weight is given to the majority classes).
  + **Weighted Average:** The weighted average is the same as the overall accuracy (0.87) because precision, recall, and F1-score are all very similar for the majority classes (0 and 1).

**Confusion Matrix:**

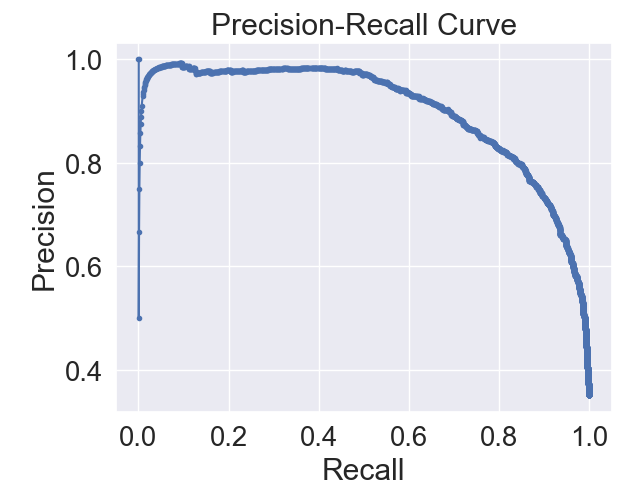
* This table shows how many data points from each actual class (rows) were predicted into each class (columns) by the model.
  + High values along the diagonal, where the model correctly predicted the class.
  + In this case, the confusion matrix confirms the observations from the classification report:
    - The model performs well on class 0 (low rent) with a high number of correct predictions (2362) and relatively few misclassified points.
    - Class 1 (medium rent) has a decent number of correct predictions (1109) but also some misclassifications (286 to low rent and 8 to high rent).
    - Class 2 (high rent) seems to be the most challenging, with very few correct predictions (10) and many misclassifications (5 to medium rent and 0 to low rent). This aligns with the class imbalance where there might be fewer examples of high rent units to learn from.

**F1 Score, Matthews Correlation Coefficient, ROC AUC Score:**

* + **F1 Score (0.8698):** This is consistent with the F1-score reported in the classification report, indicating a reasonable overall performance.
  + **Matthews Correlation Coefficient (0.7163):** This is a measure that considers both correct and incorrect classifications. A value closer to 1 indicates better performance. In this case, 0.7163 suggests a good performance but with room for improvement.
  + **ROC AUC Score (0.9556):** This metric looks at the Receiver Operating Characteristic (ROC) curve and provides a measure of how well the model can distinguish between positive and negative classes. A score closer to 1 indicates better performance at distinguishing the classes. In this case, 0.9556 suggests good overall ability to distinguish between the classes, but the confusion matrix highlights the challenge with the minority class (high rent).

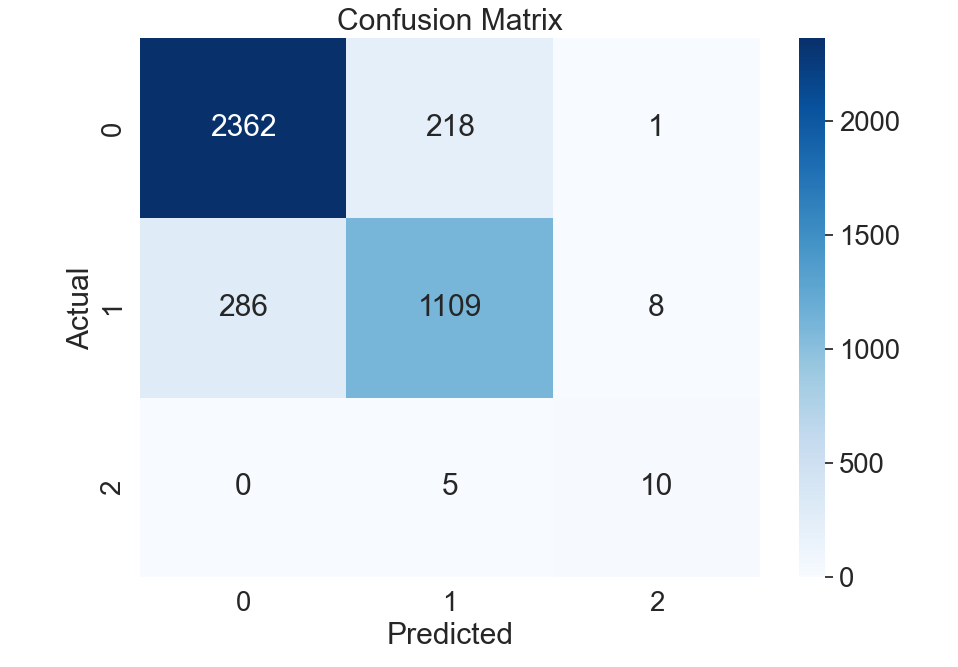
**Overall Interpretation:**

* The model seems to perform well on the majority classes (low and medium rent) with high precision, recall, and F1-score.
* However, the model struggles with the minority class (high rent), likely due to the class imbalance. There are very few high rent units in the test data, making it difficult for the model to predict the particular class.



Precision-Recall Curve for the Best Model

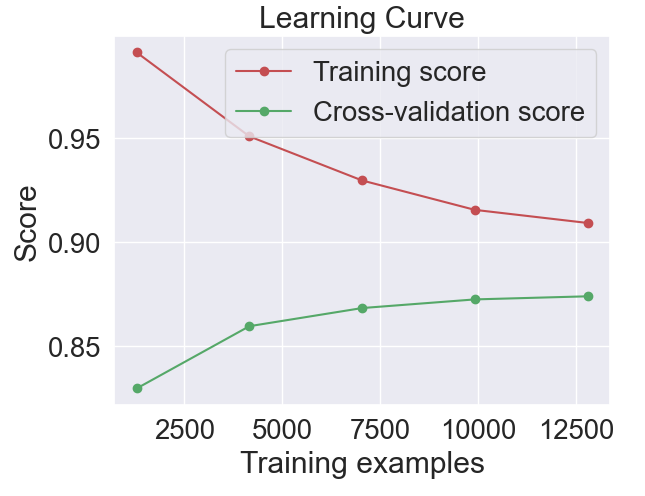
* **Higher precision at lower recall:** The initial part of the curve shows a high precision (correctly predicted high rent). This is likely because the model prioritizes avoiding false positives (predicting high rent for non-high rent units).
* **Precision drop at higher recall:** As the model tries to capture more true positives (correctly identify high rent units), it starts to include some false positives as well. This drives the precision down.



Actual vs Predicted Heatmap

**Overall:**

The heatmap suggests that the model performs well for identifying low rent units but might have difficulty with medium and high rent categories, particularly high rent which seems to be the most crucial class.



A learning curve is a plot that shows how the training and cross-validation scores of a model change with the number of training examples. Here's how to interpret the provided learning curve:

1. **Training Score**:
   * The red line represents the training score.
   * The training score starts high (near 1.0) and decreases as the number of training examples increases.
   * A high initial training score suggests that the model fits the training data very well when the training set is small.
   * The decrease in training score indicates that as the model is trained on more data, it becomes slightly less overfitted to the training data.
2. **Cross-Validation Score**:
   * The green line represents the cross-validation score.
   * The cross-validation score starts lower and gradually increases as the number of training examples increases.
   * This suggests that with more training data, the model generalizes better to unseen data.
   * The cross-validation score seems to plateau as the number of training examples increases, indicating that additional data may not significantly improve performance beyond this point.
3. **Gap Between Training and Cross-Validation Scores**:
   * A large gap between the training and cross-validation scores indicates overfitting, where the model performs well on the training data but poorly on unseen data.
   * As the number of training examples increases, the gap between the training and cross-validation scores narrows, suggesting that the model is becoming less overfitted and more generalized.

**6.2 Risk/Assumption/Limitation:**

**Model Bias:** Machine learning models can be biased towards the majority class during training. In this case, a model trained on this data might perform well for "low" rentals but perform poorly for "medium" and "high" rentals due to a lack of sufficient data for these categories. This can lead to inaccurate predictions and unreliable risk assessments for these rental categories.

**Class Imbalance:** The extreme imbalance between "low" and other categories suggests potential class imbalance issues. Techniques like oversampling, undersampling, or cost-sensitive learning might be necessary to address this imbalance and improve model performance for minority classes in the future.

**Rental Market Representation:** The data is assumed to be representative of the actual rental market distribution. If the data comes from a specific source or region with a unique rental landscape, it might not generalize well to other markets.

Limited Generalizability: The model might not perform well on unseen data with a different rental category distribution. For example, if the model is deployed in a market with a higher proportion of "medium" or "high" rentals, its predictions might be unreliable.

**Data Quality:** Data quality can affect the model's performance. Errors, missing info, or inconsistencies in the data can lead to unreliable model predictions.

**Data Understanding:** Building a good model requires a deep understanding of the data and its industry. Failing to grasp the data and its context can lead to flawed models. Data analysis is essential for translating raw data into insights that guide effective model building.