**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 19,991 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.

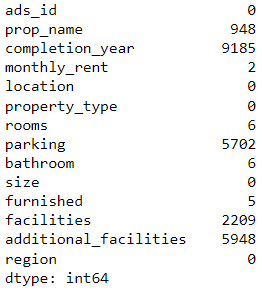


Diagram 1: Dataset contents

1. **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.

1. **df.drop()**

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

1. **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 |

Table 1: Income groups

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 |

Table 2: Rent category calculation

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 the numbers will be extracted using the function **df.str.extract.** This will create a new column (monthly\_rent\_price).Then the values were sorted out from lowest to highest using the **df.sort\_values** function. The output showed that there are some rows with 6-digits figures which is inappropriate to be considered as rental. Therefore, the values in these cells will be replaced using **df.loc** and **np.random.randint** function. But before that, the **df.filter** function is used to take a deeper look at the column like how many rows have at least 6 digits or bigger values that may indicate sales price.

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

1. **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.

1. **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).

1. **rooms**

During the visualization of the data, there was a string ‘More than 10’. The value of this cell(s) was 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.

1. **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.

1. **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).

1. **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.**

At the end of the cleaning, there are 19,991 rows and 35 columns. Below is the column list;

1. completion\_year
2. monthly\_rent
3. location
4. property\_type
5. rooms
6. parking (\*number of parking)
7. bathroom
8. size
9. furnished
10. facilities
11. additional\_facilities
12. region
13. monthly\_rent\_price
14. rent\_category
15. Barbeque area
16. Club house
17. Gymnasium
18. Jogging Track
19. Lift
20. Minimart
21. Multipurpose hall
22. Parking (\*have or do not have parking)
23. Playground
24. Sauna
25. Security
26. Squash Court
27. Swimming Pool
28. Tennis Court
29. Air-Cond
30. Cooking Allowed
31. Internet
32. Near KTM/LRT
33. Washing Machine
34. size\_num\_sf
35. size\_bins
    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.

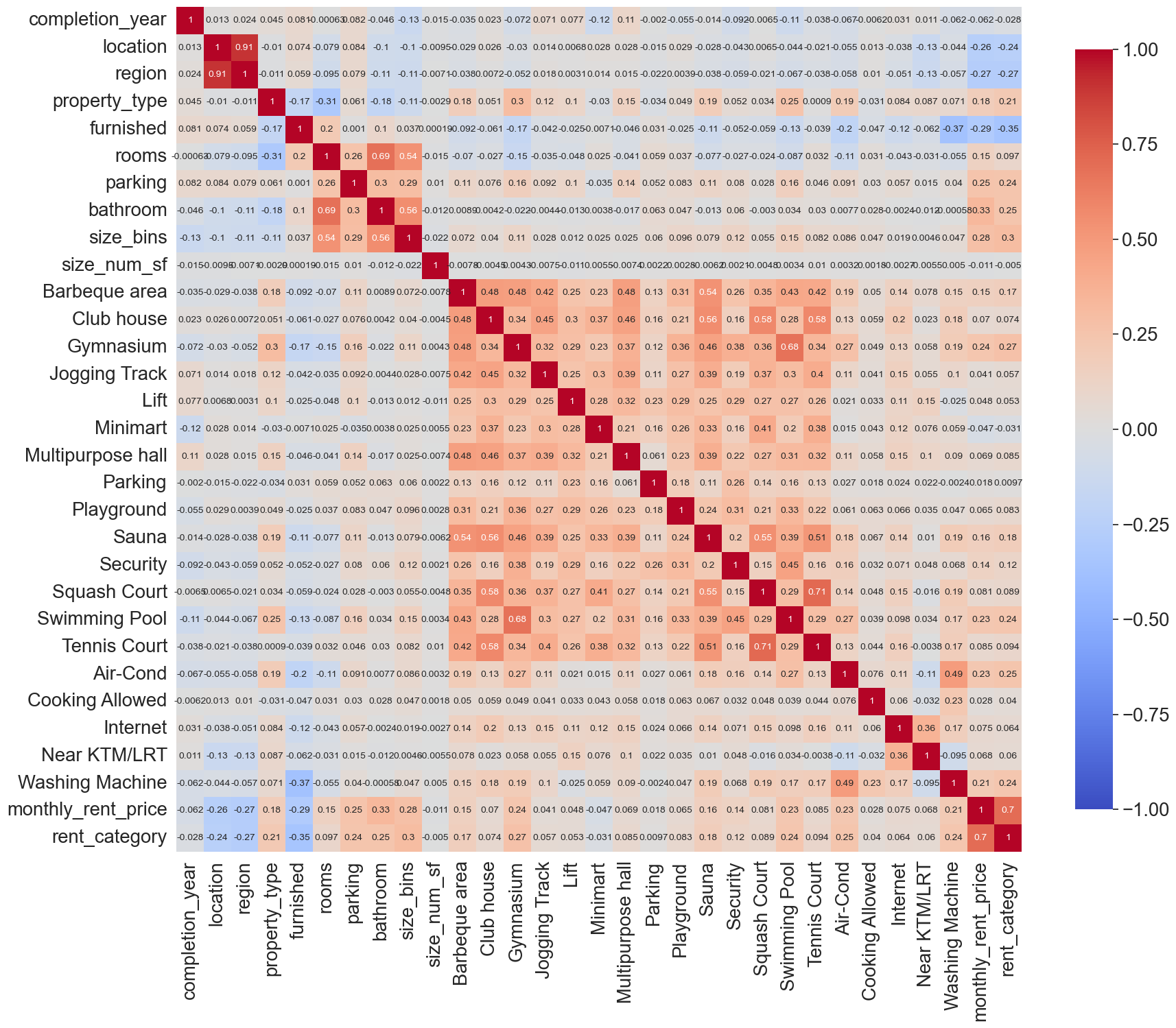
****

Diagram 2: 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.
* Completion\_year (-0.028): Weak negative correlation
* Region (-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**

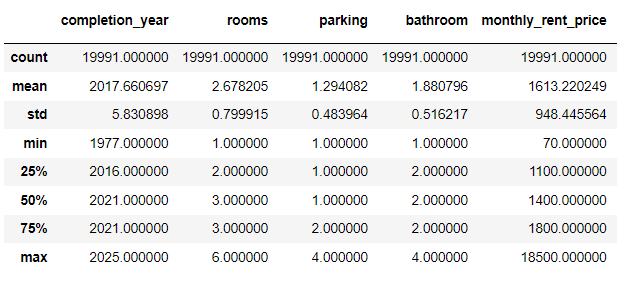
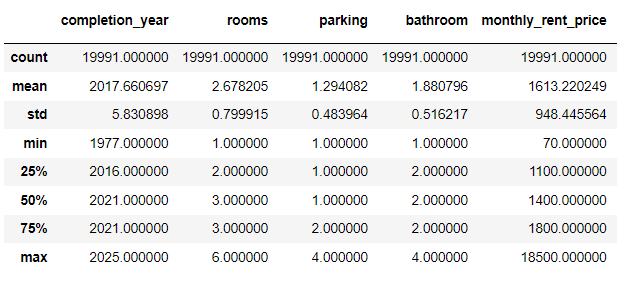
****

Table 3: Mathematical description of the dataset

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.

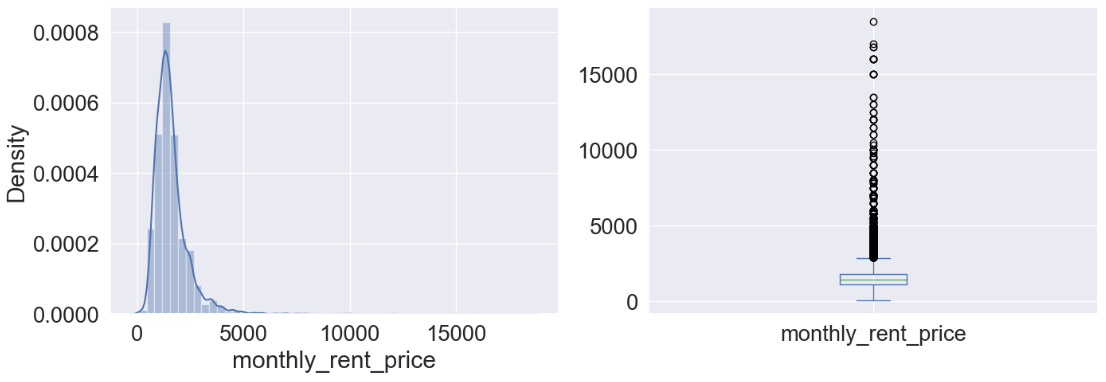
****

Diagram 3: Outlier projection

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:0.8055277638819409

Test: 0.80320080020005

CV score:0.8034014890128462

**LR 2 (with parameter)**

Train:0.7288644322161081

Test: 0.7171792948237059

CV score:0.7263005891655088

**LR 3 (with GridSearchCV)**

Train:0.8056528264132066

Test: 0.8024506126531633

CV score:0.8037141257987711

* 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:0.9890570285142571

Test: 0.8404601150287572

CV score:0.8528640223522007

* 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:0.7719484742371185

Test: 0.7694423605901476

CV score:0.7719485705449307

* 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:0.8192221110555278

Test: 0.8069517379344836

CV score:0.8108431320685151

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

* 1. **Random Forest**

**RF 1 (no parameter)**

Train:0.9889944972486243

Test: 0.8832208052013003

CV score:0.8825034636957569

* The train score is significantly higher than the test score, suggesting an overfitting.

**RF 2 (with parameter)**

Train:0.8683716858429215

Test: 0.8467116779194799

CV score:0.8491747636114397

* The chosen hyperparameters for this model might be too restrictive as the scores are lower than RF 1.
* The gap between training and test scores is smaller than RF 1.

**RF 3 (with GridSearchCV)**

Train:0.8495497748874438

Test: 0.8304576144036009

CV score:0.8303528443945799

* This model achieves a balance between training and test scores (around 0.83-0.85) 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:0.8654327163581791

Test score:0.8522130532633159

CV score: 0.854301913062654

* 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:0.9129564782391195

Test score:0.8779694923730933

CV score: 0.8825661005305558

* The high train score than the test suggests the model has an overfitting.

**GB 3 (with GridSearchCV)**

Train score:0.9365932966483241

Test score:0.8907226806701676

CV score: 0.8921335447033265

* The training score (0.93) is significantly higher than previous GB models and higher than the test score(0.89) , suggesting an 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:** GB 2 and GB 3 have scores between 0.87-0.93, while GB 1 has a minimal gap between train and test scores. This might be the best-performing model overall for now.
* **Logistic Regression:** All the models have scores between 0.80, it might be the most stable models for now.
* **Random Forest:** RF 1 has a significant overfitting issue while RF 2 and RF 3 seem to have slightly better scores and stability.
* **Decision Tree:** DT 2 has the lowest scores, DT 3 may seems as the stable model and DT 1 has a significant overfitting issue.

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**

**LR 1**

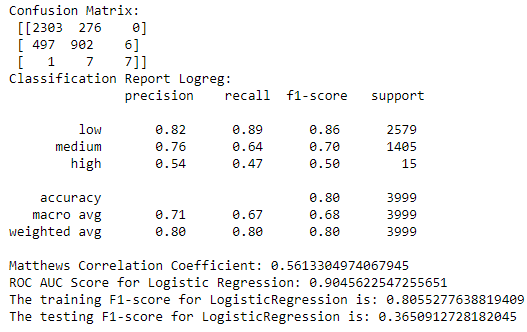
****

Table 4: Logistic Regression without parameter (1) Evaluation Result

**Confusion Matrix:**

* **Overall Accuracy:** The model seems to have an overall accuracy of 80% (based on the reported accuracy in the classification report), which means it correctly classified 80% of the rent categories in the test set.
* **Class-wise Performance:** However, the confusion matrix reveals an imbalance in performance across categories:
  + **Low Rent:** The model performs well for the "low" rent category with a high precision (0.82) and recall (0.89). This means the model accurately predicts most instances in this category.
  + **Medium Rent:** Performance drops for the "medium" category with lower precision (0.76) and recall (0.64). There are more false positives (incorrectly predicted as medium) and false negatives (missed medium rent units).
  + **High Rent:** Performance is significantly lower for the "high" rent category with low precision (0.54) and recall (0.47). This indicates the model struggles to accurately identify high rent units, often misclassifying them as other categories.

**Classification Report LR 1:**

This report confirms the observations from the confusion matrix. While the overall accuracy is decent, the model struggles with the "high" rent category.

**Other Metrics:**

* **F1-Score:** The reported F1-score for the test set (0.365) is significantly lower than the training F1-score (0.805), suggesting potential overfitting. The model might be performing well on the training data but failing to generalize well to unseen data.
* **Matthews Correlation Coefficient (MCC):** The MCC of 0.56 indicates a moderate correlation between predicted and actual rent categories.
* **ROC AUC Score:** The ROC AUC score of 0.9 suggests a good ability to distinguish between rent categories, but this metric can be misleading for imbalanced data.

**Overall Observations:**

* The model shows promise for predicting low and medium rent categories but needs improvement for high rent units.
* The F1-score and MCC suggest potential overfitting.

**LR 2**

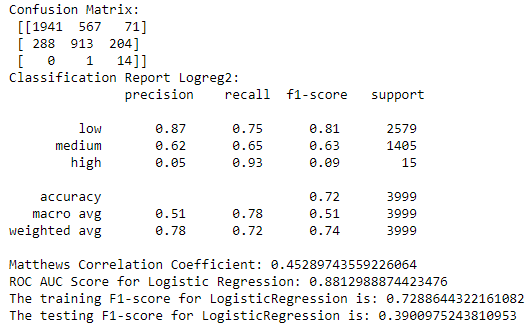
****

Table 5: Logistic Regression -with parameters (2) Evaluation Result

**Confusion Matrix:**

* **Overall Accuracy:** The model's accuracy dropped slightly to 72%.
* **Low Rent:** Performance remains good for the "low" rent category with high precision (0.87) but lower recall (0.75) compared to the previous model. This suggests the model might be misclassifying some low rent units as other categories.
* **Medium Rent:** There's a slight improvement in precision (0.62) for the "medium" category, but recall (0.65) remains similar. The model still struggles to identify some medium rent units.
* **High Rent:** This is where the significant improvement lies. The model's recall for the "high" rent category jumped to 0.93, indicating it's now very good at identifying most high rent units. However, the precision dropped significantly (0.05). This means many instances predicted as high rent actually belong to other categories (high false positives).

**Classification Report Logreg2:**

This report confirms the observations from the confusion matrix. The high recall for "high" rent comes at the cost of very low precision. The overall F1-score (0.39) suggests the model might be overfitting to the high rent category.

**Other Metrics:**

* **F1-Score:** The testing F1-score improved slightly compared to the previous model but still indicates potential overfitting.
* **Matthews Correlation Coefficient (MCC):** The MCC of 0.45 suggests a moderate correlation between predicted and actual rent categories.
* **ROC AUC Score:** The ROC AUC score of 0.88 is still good but might be misleading due to the imbalanced data.

**Effect of Class Weight:**

The introduction of class\_weight='balanced' in LogisticRegression2 seems to have addressed the issue of underperforming for the "high" rent category. However, it might have come at the expense of performance for other categories.

**Overall Observations:**

* The model significantly improved at identifying high rent units (high recall).
* However, the very low precision for "high" rent indicates excessive false positives.
* The F1-score and MCC suggest potential overfitting.

**LR 3**

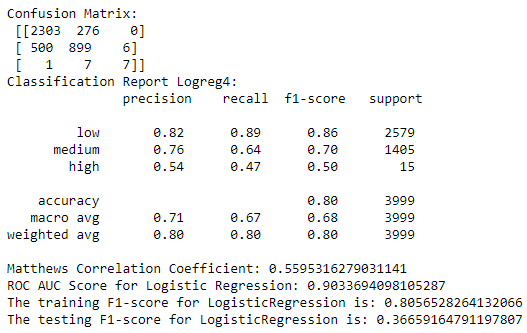
****

Table 6: Logistic Regression – with GridSearchCV (3) Evaluation Result

**Similarities to Original Model:**

* The confusion matrix, classification report, and other metrics are almost identical to the results for LogisticRegression (without parameter).
* This suggests that the default hyperparameters for logistic regression might already be suitable.
  1. **Decision Tree**

**DT1**

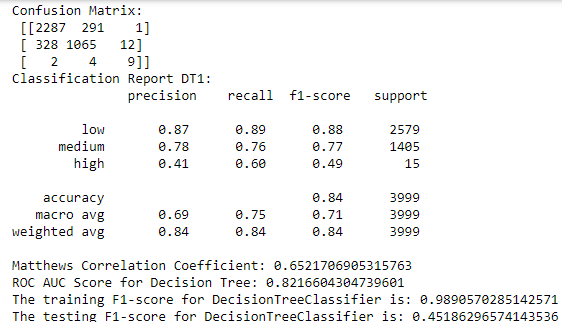
****

Table 7: Decision Tree - without parameter (1) Evaluation Result

**Overall Performance:**

* DT1 achieves a higher overall accuracy (84%) compared to the Logistic Regression models (around 80%). This suggests the Decision Tree might be better at capturing the relationships between features and rent categories.

**Class Imbalance:**

* Similar to the Logistic Regression models, DT1 still struggles with the "high" rent category, evident in the lower precision (0.41) and higher recall (0.60). It identifies many high rent units correctly but also generates some false positives.

**Other Metrics:**

* **FI Score:** The testing F1-score (0.452) for DT1 is still lower than the training F1-score (0.989), suggesting potential overfitting. However, it's significantly higher compared to the F1-scores of the Logistic Regression models (around 0.36), indicating better overall performance on the testing set.
* **The Matthews Correlation Coefficient (MCC):** MCC of 0.65 suggests a moderate correlation between predicted and actual rent categories, which is higher than the MCC for Logistic Regression models (around 0.5).
* **ROC AUC Score:** The ROC AUC score of 0.82 for DT1 is lower than some Logistic Regression models (around 0.9). However, ROC AUC can be misleading for imbalanced data, so F1-score is a more reliable metric in this case.

**Decision Tree (DT1) vs. Logistic Regression:**

* DT1 seems to be a better choice compared to the Logistic Regression models you explored previously. It achieves higher overall accuracy and F1-score on the testing set.

**DT 2**

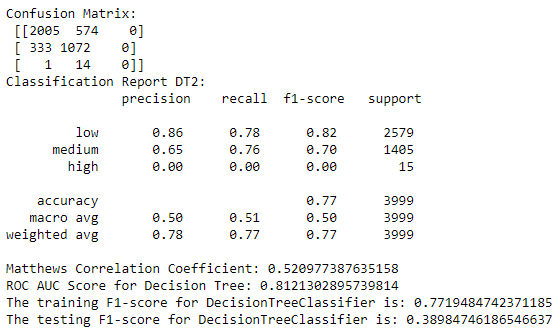
****

Table 8: Decision Tree -with parameters (2) Evaluation Result

**Class Imbalance:**

* **High Rent Unit Misclassification:** DT2 completely fails to identify any high rent units (precision and recall of 0.00).
* This is a significant issue as identifying high rent units is important.

**Performance Decline:**

* DT2 has a lower overall accuracy (77%) and testing F1-score (0.390) compared to DT1 (84% and 0.452). This suggests the chosen hyperparameter might have led to overfitting or a model less suitable for the data.

**Metrics:**

* **FI Score:** The testing F1-score (0.389) for DT1 is still lower than the training F1-score (0.771), suggesting potential overfitting. Lower than DT 1.
* **Matthews Correlation Coefficient (MCC)**: MCC of 0.52 is lower than DT1 (0.65) and suggests a weaker correlation between predicted and actual rent categories.
* **ROC AUC:** The score (0.81) is similar to DT1 and not very informative due to the imbalanced data.

**Overall:**

* While DT2 performs worse than DT1, it still achieves a similar overall accuracy to the Logistic Regression models (around 80%).
* However, DT2 completely misses all high rent units, making it a less suitable choice if identifying high rent units is crucial.

**DT 3**

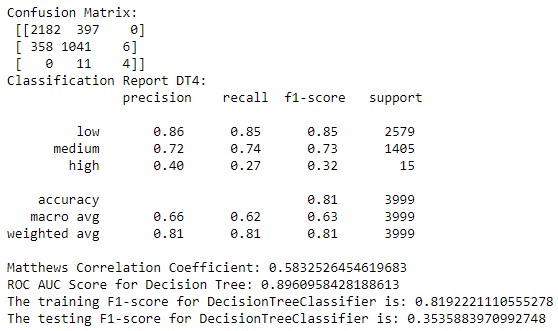
****

Table 9: Decision Tree – with GridSearchCV (3) Evaluation Result

**Class-Level Performance:**

* **Performance Class:** Precision and recall for DT 1 serve better than DT and DT.
* **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.

**Other Metrics:**

* **F1 Score**: Lower Overall F1 (train:0.81 and test:0.35) than DT 1 (train:0.98 and tets:0.45) but better than to DT2.
* **MCC score:** The MCC (0.58) is slightly better than DT2 but lower than DT1 (0.65), suggesting a slightly lower correlation between true and predicted classifications than DT 1 but better than DT 2.
* **ROC AUC score:** A score of 0.89 is the highest compared to other DT models, indicating good overall discrimination between rent categories.

**Overall Observations:**

* **Overall Accuracy:** DT3 achieves a decent overall accuracy (81%) similar to DT1 (84%) and slightly better than DT2 (77%).
* **High Rent Unit Identification:** DT3 performs better than DT2 at identifying high rent units, with a precision of 0.40 and recall of 0.27. This suggests GridSearchCV might have found a configuration that balances performance across categories.

**Drawbacks:**

* **Testing F1-score:** Similar to previous models, DT4 still suffers from a significant gap between training and testing F1-score (0.82 vs 0.35), suggesting potential overfitting.
  1. **Random Forest**

**RF 1**

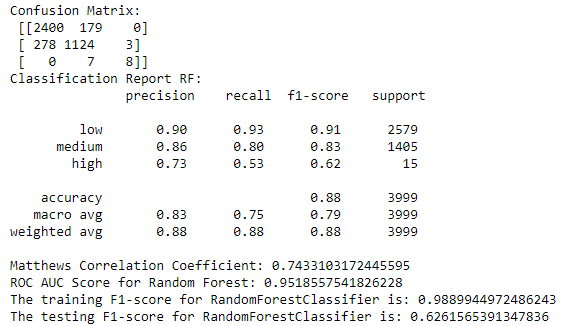
****

Table 10: Random Forest - without parameter (1) Evaluation Result

**Overall Performance:**

* RF achieves a significantly higher overall accuracy (88%) compared to the Decision Tree models (DT1: 84%, DT2: 77%, DT3: 81%). This suggests Random Forest is better at capturing the complex relationships between features and rent categories.

**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 0.90 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.

**Other Metrics:**

* **F1 Score:** The testing F1-score (0.63) for RF is lower than the training F1-score (0.99), suggesting some overfitting, but the gap is much smaller compared to the Decision Tree models. This indicates better generalization to unseen data.
* **MCC score:** MCC score of 0.74 is significantly higher than all Decision Tree models, suggesting a strong correlation between predicted and actual rent categories.
* **ROC AUC score:** The ROC AUC score of 0.95 for RF is also higher than the Decision Tree models, indicating good performance even with imbalanced data.

**Random Forest vs. Decision Trees:**

* Overall, Random Forest outperforms the Decision Tree models (DT1, DT2, DT3) in terms of accuracy, high rent unit identification, and generalizability (lower overfitting).

**RF 2**

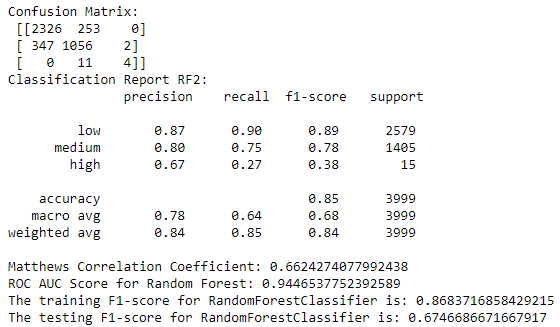
****

Table 11: Random Forest -with parameters (2) Evaluation Result

**Class-Level Performance:**

* **Overall Accuracy:** RF2 has a slightly lower overall accuracy (85%) compared to RF (88%).
* **Consistent Performance on Majority Class:** Precision and recall for the "low" and ‘medium’ class are similar between RF1 and RF2.
* **Performance on high-rent class:** RF2 has a lower precision and recall (0.67 and 0.27) for high rent units compared to RF (precision: 0.73, recall: 0.53). This suggests RF2 might be identifying slightly fewer high rent units.

**Other Metrics:**

* **Lower Overall F1:** The training and testing F1-scores (around 0.86 and 0.67, respectively) are lower than RF1. The testing F1-score is closer to the training F1-score compared to RF, suggesting that parameter tuning might have helped reduce overfitting
* **MCC:** The MCC (0.66) is significantly lower than RF 1, but can be considered a moderate correlation between predicted and actual rent categories.
* **ROC AUC score**: A score (0.94) is similar to RF1 (0.95).

**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**

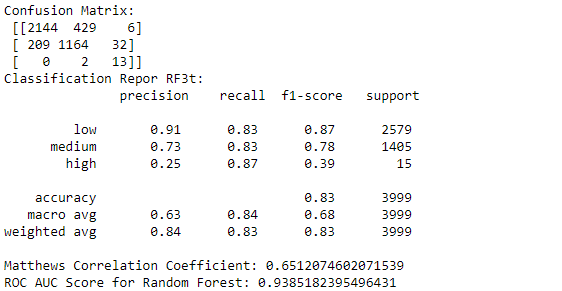
**** ****

Table 12: Random Forest – with GridSearchCV (3) Evaluation Result

**Class-Level Performance:**

* **Overall Accuracy:** The overall accuracy (83%) lower than previous models.
* **High Rent Unit Identification:** This is the most significant improvement. RF3 now has a recall of 0.87 for high-rent units, meaning it captures most of them. While the precision (0.25) suggests it might classify some non-high-rent units as high rent.

**Other Metrics:**

* **F1 Score**: The testing F1-score (0.64) for RF3 is still lower than the training F1-score (0.85), suggesting some overfitting
* **MCC score:** A score of 0.65 is similar to previous models, indicating a moderate correlation between predicted and actual rent categories.
* **ROC AUC score:** A score of 0.94 is also similar, indicating good overall performance despite the focus on the minority classes.
  1. **Gradient Boosting**

**GB 1**

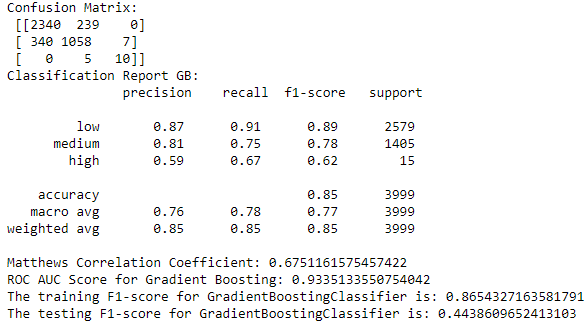
****

Table 13: Gradient Boosting - without parameter (1) Evaluation Result

**Class Level Performance:**

* **Overall Accuracy:** The overall accuracy (85%) is similar to the previous Random Forest models.
* **High Rent Unit Identification:** Lower precision score than RF 1 and RF2 but better than RF 3 and better recall than RF 1 and RF 2 but lower than RF 3.

**Other Metrics:**

* The testing F1-score (0.44) is significantly lower than the training F1-score (0.87), suggesting severe overfitting. This is a major concern and indicates the model is not performing well on unseen data.
* The Matthews Correlation Coefficient (MCC) of 0.67 is similar to the previous models.
* The ROC AUC score of 0.93 is also similar, indicating good overall performance on the majority classes (low and medium rent) despite the high rent unit issue.

**GB 2**

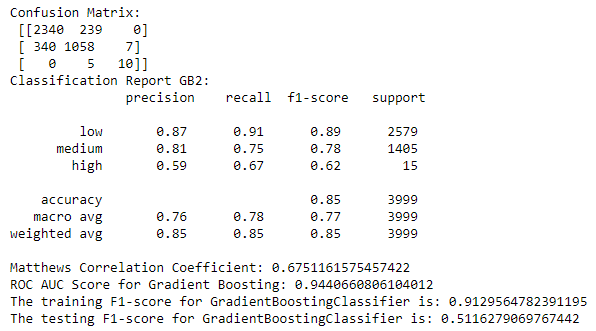
****

Table 14: Gradient Boosting -with parameters (2) Evaluation Result

**Overall Performance:** The overall accuracy (85%). GB 2 scores are similar to GB 1. It seems the parameter tuning implemented didn't significantly impact identifying high-rent units (recall: 0.67, precision: 0.59).

**Other Metrics:**

* **F1 Score:** The testing F1-score (0.51) is still lower than the training F1-score (0.91), but the overfitting gap GB 2 (0.4) is slightly smaller compared to GB 1 (0.42). This suggests parameter tuning helped reduce overfitting even though it is not much.
* **MCC score:** It has the same MCC score (67%) as GB 1.
* **ROC AUC score**: A score of 0.94 and most metrics remain similar to GB, indicating good performance on the majority classes (low and medium rent)

**GB 3**

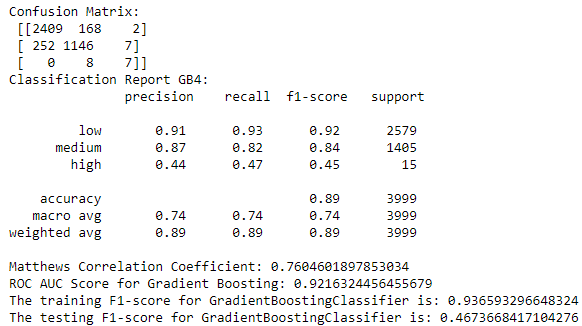
****

Table 15: Gradient Boosting – with GridSearchCV (3) Evaluation Result

**Class-Level Performance:**

* **Overall Accuracy:** The overall accuracy (89%) is the highest among all models tested.
* **Consistent Performance on Majority Class:** Precision and recall for the "low" and ‘medium’ classes are higher than GB 1 and GB 2.
* **Performance on high-rent class:** GB 3 has a lower precision and recall (0.44 and 0.47) for high-rent units compared to GB 1 and GB 2. This suggests the model might be more cautious in identifying high-rent units to avoid overfitting, potentially missing some.

**Other Metrics:**

* **F1 Score:** The test score (0.46) is still lower than the training score (0.93). The overfitting gap (0.47) between previous models did not improve.
* **MCC Score:** A score of 0.76 is the highest among all models, indicating a stronger correlation between predicted and actual rent categories.
* **ROC AUC Score:** A score of 0.92 is slightly lower than previous versions but still suggests good overall performance despite the focus on imbalanced data.

**5.3 Model Selection**

From the evaluation of models, in Logistic Regression, the **LR1 is preferred** for now.

* All models struggle with the imbalanced data, particularly for the "high" rent category.
* High accuracy of 80% and strong performance on 'low' and 'medium' classes.
* High MCC (0.56) and ROC AUC (0.90) scores.
* Balanced performance across classes without significant overfitting.
* F1-score for the test set (0.365) is significantly lower than the training F1-score (0.805), suggesting potential overfitting. The model might be performing well on the training data but failing to generalize well to unseen data.
* GridSearchCV didn't improve performance significantly in this case.

From the Decision Tree models, **DT 1 is preferred.**

* **Overall Accuracy:** DT1 achieves a good overall accuracy (84%), suggesting it can correctly classify most rent categories.
* **High Rent Unit Identification:** While not perfect, DT1 offers a balance between identifying high rent units (precision: 0.41, recall: 0.60) and overall performance.
* **DT2 (with parameter):** This model suffers from significantly worse performance, particularly in identifying high-rent units (precision and recall of 0.00).
* **DT3 (with GridSearchCV):** While DT3 shows some improvement in identifying high rent units compared to DT2, it still has a significant overfitting issue (gap between training and testing F1-score) that needs to be addressed.

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

* **Overall Accuracy:** RF 2 has slightly lower accuracy at 85% than RF 1 (88%), but the overfitting gap (0.19) is also lesser than RF 1(0.36) and RF 3 (0.21).
* **High Rent Unit Identification:**
* RF1: Decent recall (0.53) but higher precision (0.73) for high rent units.
* RF2: Lower recall (0.27) but higher precision (0.67) for high rent units, indicating a more cautious approach.
* RF3: Significantly improved recall (0.87) for high rent units, however, the precision (0.25) suggests it might classify some non-high rent units as high rent (false positives).
* **MCC Score**: All models have similar Matthews Correlation Coefficient (MCC) values (around 0.65-0.74), indicating a moderate correlation between predicted and actual rent categories.
* **ROC AUC Score:** All models have high ROC AUC scores (around 0.94-0.95), indicating good performance even with imbalanced data.

From all the Gradient Boosting models, **GB 3 is preferred.**

* **Overall Accuracy**: GB 3 has the highest accuracy at 0.89, indicating better generalization on the test set.
* **Rent category**: GB 3 has higher precision and recall for ‘low’ and ‘medium’ classes than GB 1 and GB 2 models. Slightly lower for ‘high’ class units, but this might indicate a more cautious approach.
* **F1-Score**: the F1 test score is 0.46, indicates a slightly better performance in terms of the harmonic mean of precision and recall than GB 1, even though slightly lower than in GB 2.
* **MCC**: GB 3 has the highest MCC (0.760), suggesting a better correlation between predicted and actual classes.
* **ROC AUC Score:** All models have high ROC AUC scores (around 0.92-0.94), indicating good performance even with imbalanced data
  1. **Gradient Boosting with SMOTEENN**

Of the 4 selected models, the Gradient Boosting model (GB 3) is preferred due to higher accuracy, MCC, and ROC AUC score, though it suffers from overfitting and a slightly lower F1 test score.

To improve the model further, it was trained with a resampling technique, Synthetic Minority Oversampling Technique + Edited Nearest Neighbors (SMOTEENN). A SMOT will generate synthetic samples for the minority class without affecting the actual dataset. ENN is used to clean the data by removing any samples that are misclassified by their nearest neighbors (GeekforGeeks, 2024). This combination helps in cleaning up the synthetic samples, improving the overall quality of the dataset (GeekforGeeks, 2024).

For extra work, I did try out other GB models with SMOTEENN.

* + 1. **GBR 1**

**Gradient Boosting with SMOTEENN**

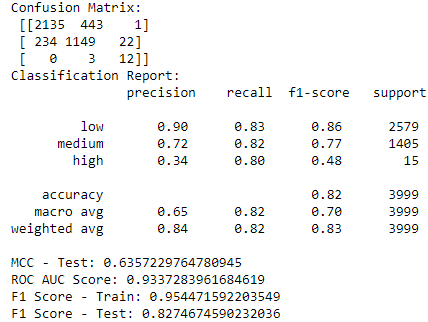
****

Table 16:Gradient Boosting – with SMOTEEN Evaluation Result

**Performance:**

* **Overall Accuracy:** The overall accuracy (82%) is lower compared to previous models without SMOTEENN (around 85-89%). However, considering the class imbalance, this might be a reasonable trade-off.
* **High Rent Unit Identification:** This model seems to prioritize recall (0.80) for high rent units, suggesting it captures most of them, even if it generates some false positives (classifies non-high rent units as high rent). This is an improvement over previous models without SMOTEENN that struggled with recall for high rent units.
* **Other Metrics:** The testing F1-score (0.83) is still lower than the training F1-score (0.95), suggesting some overfitting might be present even with SMOTEENN. The MCC (0.64) and ROC AUC score (0.93) remain decent, indicating a moderate correlation between predicted and actual rent categories and good overall performance despite the imbalanced data.

**Impact of SMOTEENN:**

* While overall accuracy is lower, SMOTEENN seems to have addressed the issue of identifying high rent units effectively. This suggests SMOTEENN successfully increased the representation of the minority class (high rent) in the training data, allowing the model to learn from them better.
  + 1. **GBR 2**

**Gradient Boosting with SMOTEENN (with weight)**

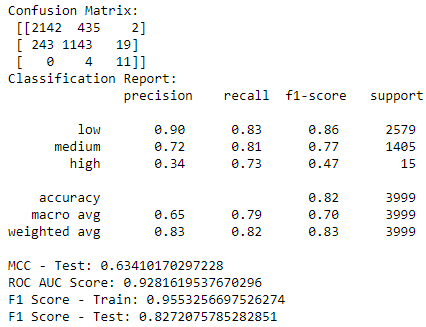
****

Table 17:Gradient Boosting – with SMOTEEN (with weight) Evaluation Result

**Performance:**

* **Overall Accuracy and Metrics:** The overall accuracy (82%), MCC (0.63), ROC AUC score (0.93), and F1-score (0.83) are very similar to the previous model with SMOTEENN (without weight resampling). This suggests weight resampling might not have had a significant impact on overall performance.
* **High Rent Unit Identification:** The recall for high rent units (0.73) is slightly higher compared to the previous model (0.80). The precision (0.34) remains similar. This suggests a small improvement in identifying some true high rent units while maintaining the same level of false positives.
* **Overfitting:** The gap between training F1-score (0.96) and testing F1-score (0.83) is still present, indicating some overfitting. However, it's slightly smaller compared to the previous model (0.95 vs 0.83). Weight resampling might have contributed to a marginally reduced overfitting effect.

**Resampling with Weight Impact:**

* The impact of weight resampling in this case seems to be subtle. It might have slightly improved high rent unit recall and reduced overfitting to a small degree. However, the overall performance remains very similar to the previous model with SMOTEENN.
  + 1. **GBR 3**

**Gradient Boosting with SMOTEENN (with weight and parameter from GB 2)**

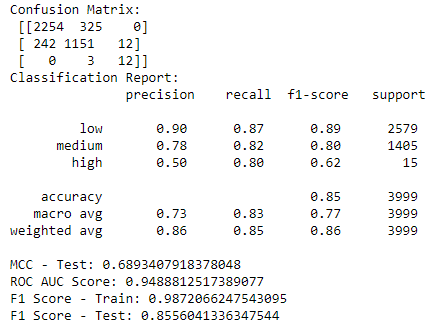


Table 18:Gradient Boosting – with SMOTEEN (with weight and parameter from GB 2) Evaluation Result

**Performance:**

* **Overall Accuracy:** The overall accuracy (85%) is slightly higher than previous models with SMOTEENN (around 82%).
* **High Rent Unit Identification:** This model shows a good balance:
  + Recall for high rent units (0.80) remains high, indicating successful identification of most true positives.
  + Precision (0.50) has improved compared to previous models (around 0.34), suggesting a reduction in false positives (classifying non-high rent units as high rent).
* **Overfitting:** The gap between training F1-score (0.99) and testing F1-score (0.86) is significantly smaller compared to previous models. GridSearchCV seems to have effectively reduced overfitting while maintaining good performance.
* **Other Metrics:** The MCC (0.69) and ROC AUC score (0.95) are also improved, indicating a stronger correlation between predicted and actual rent categories and good overall performance despite the imbalanced data.
  + 1. **GBR 4**

**Gradient Boosting with SMOTEENN (with weight and parameter from GB 3)**

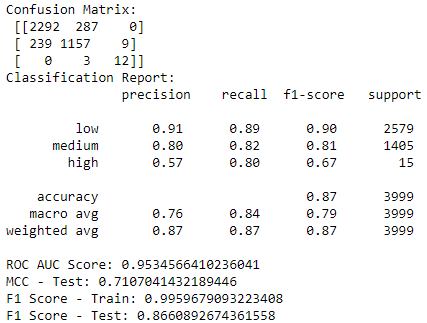
****

Table 19:Gradient Boosting – with SMOTEEN (with weight and parameter from GB 3) Evaluation Result

**Performance:**

* **Overall Accuracy:** The overall accuracy (87%) is the highest among all models trained.
* **Low and Medium Rent class:** Has higher precision and recall than previous models.
* **High Rent Unit Identification:** This model maintains a good balance:
  + Recall for high rent units (0.80) remains high, indicating successful identification of most true positives.
  + Precision (0.57) has improved again compared to GBR3 (0.50), suggesting a further reduction in false positives.
* **Overfitting:** The gap between training F1-score (0.99) and testing F1-score (0.87) is minimal, indicating excellent performance with minimal overfitting. GridSearchCV seems to have effectively addressed this issue.
* **Other Metrics:** The MCC (0.71) and ROC AUC score (0.95) are also the highest, suggesting a very strong correlation between predicted and actual rent categories and exceptional overall performance despite the imbalanced data.

**Impact of Tuning and Weight Resampling:**

* GridSearchCV likely played a crucial role in achieving the best overall performance and minimal overfitting.
* Weight resampling might have further contributed to the improvement in high rent unit precision (reducing false positives) compared to GBR 3.

**Comparison with Other Models:**

GBR4 surpasses all previous models in terms of:

* Overall accuracy
* High rent unit identification (both recall and precision)
* Reduced overfitting

**Conclusion:**

**GBR 4** with SMOTEENN, weight resampling, and parameter tuning from GridSearchCV appears to be the **best-performing model** overall. It offers exceptional performance on all metrics, effectively addresses the imbalanced data issue, and excels at identifying high rent units with good precision.

* 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 5 selected models discussed previously, to see how each model uses the features from the dataset or to see the most correlated features.

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

* **Logistics Regression**

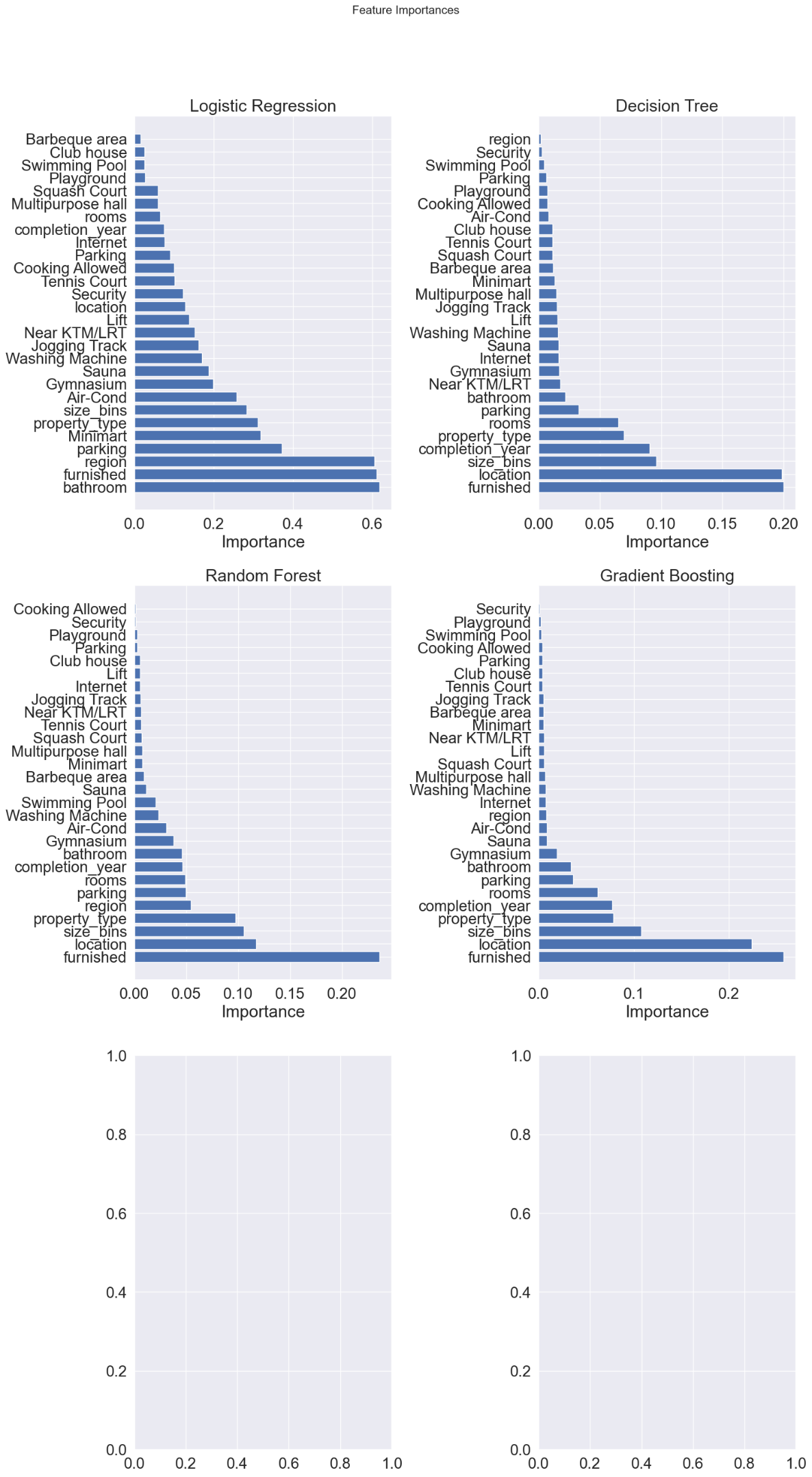
****

Diagram 4: Logistic Regression (LR 1) Feature Importance

From the graph above, the most significant feature is bathroom, furnished, region, parking, property\_type, size\_bins. Top facilities/amenities that influence the prediction are Minimart, Gymnasium and Sauna.

* **Decision Tree**

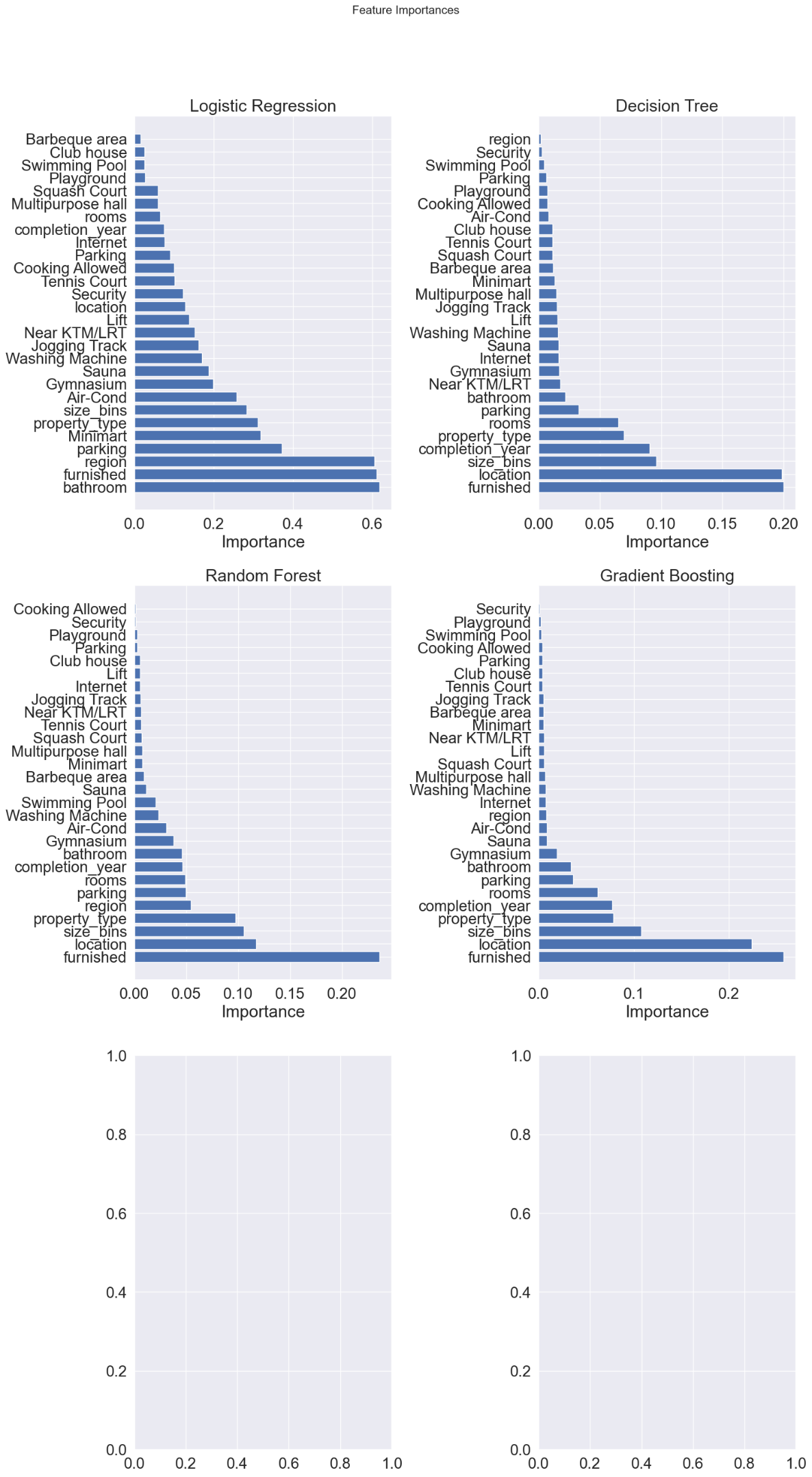
****

Diagram 5: Decision Tree (DT 1) Feature Importance

From the graph above, the most significant feature is furnished, followed by location, size\_bins, completion\_year, property\_type, rooms, parking, and bathroom. Top amenities that are influencing the rental prediction are units that are near to KTM/LRT, the gym and having Internet connection.

* **Random Forest**

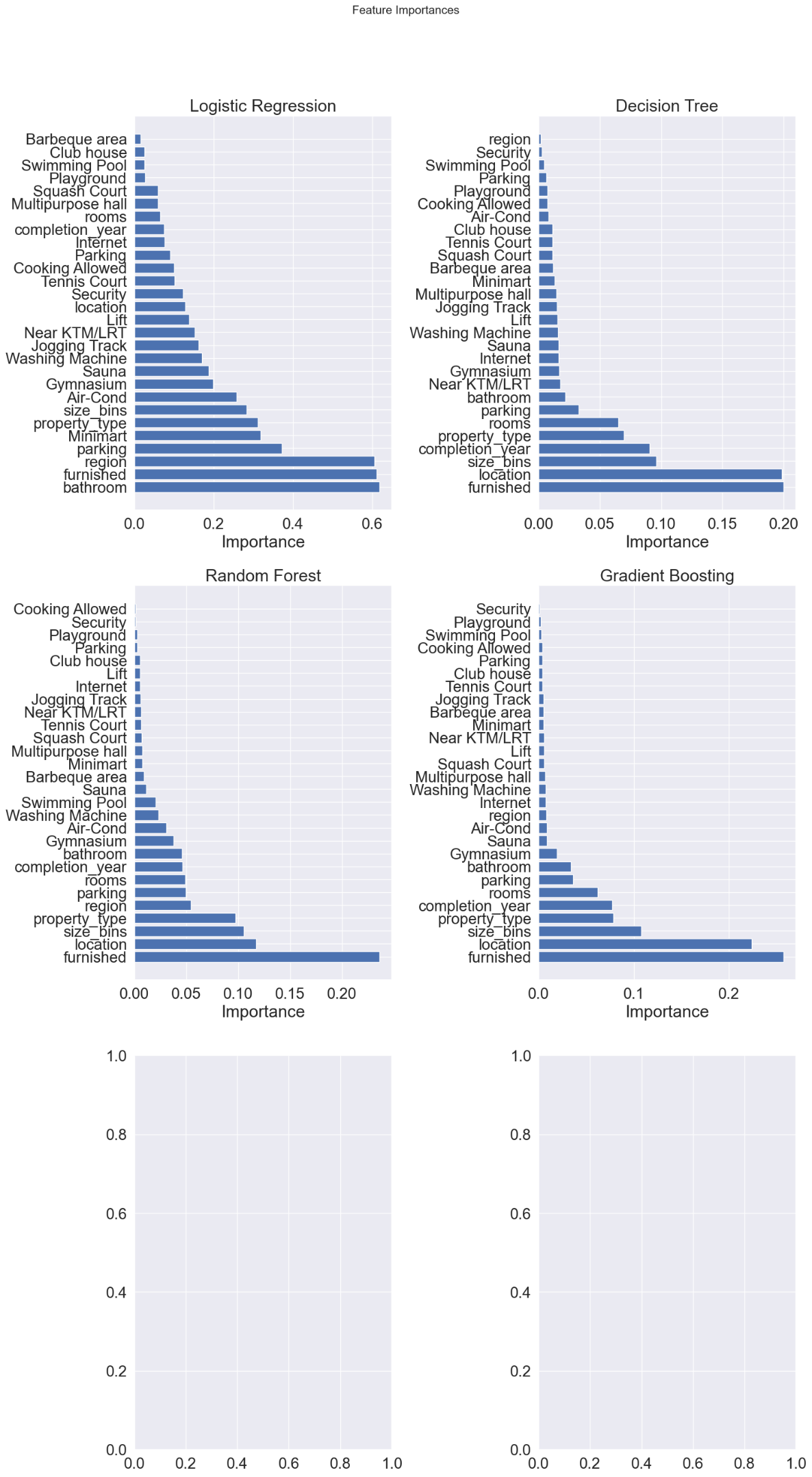
****

Diagram 6: Random Forest (RF 2) Feature Importance

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

* **Gradient Boosting**

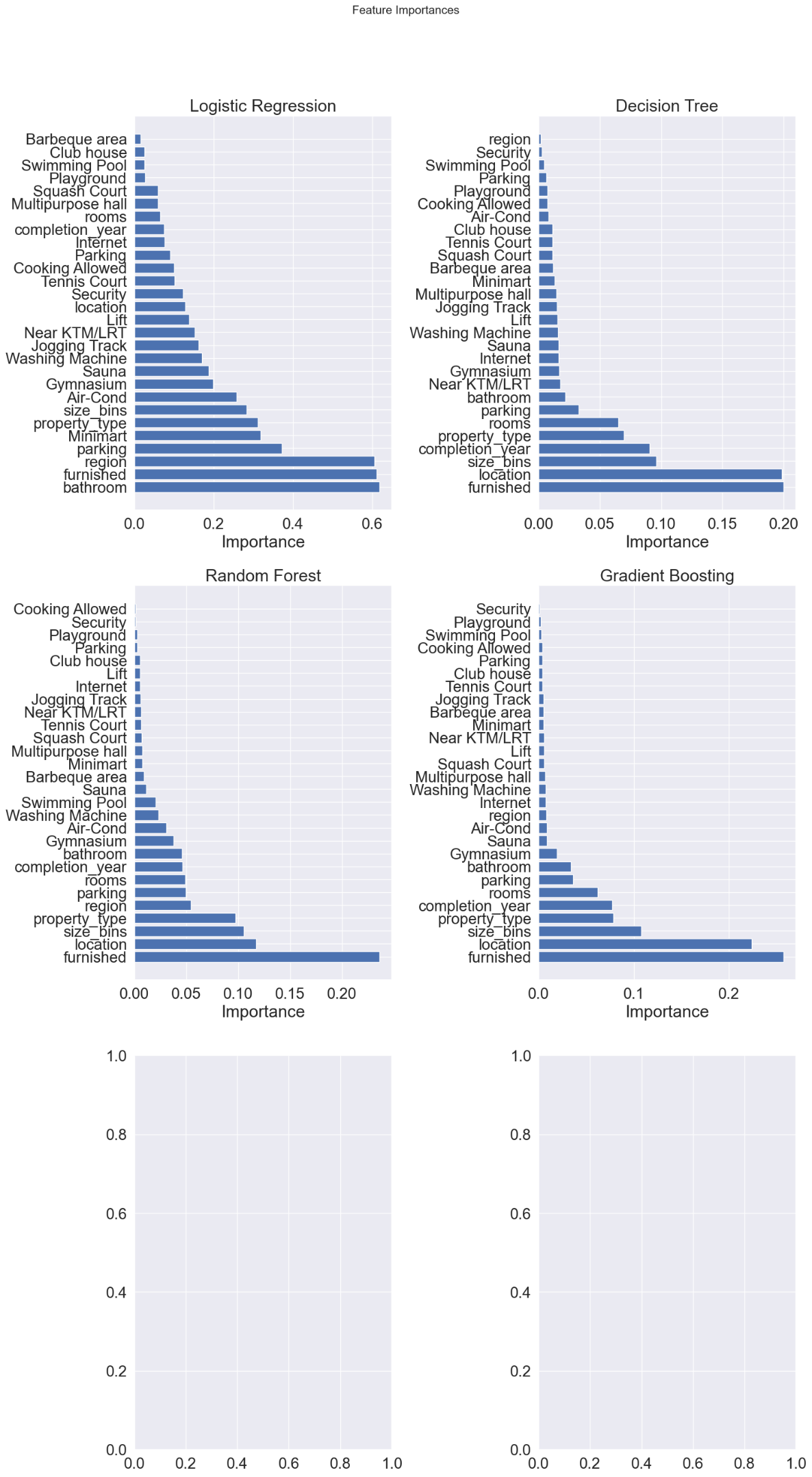
****

Diagram 7: Gradient Boosting (GB 3) Feature Importance

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

* **Gradient Boosting Resampling (GBR 4)**

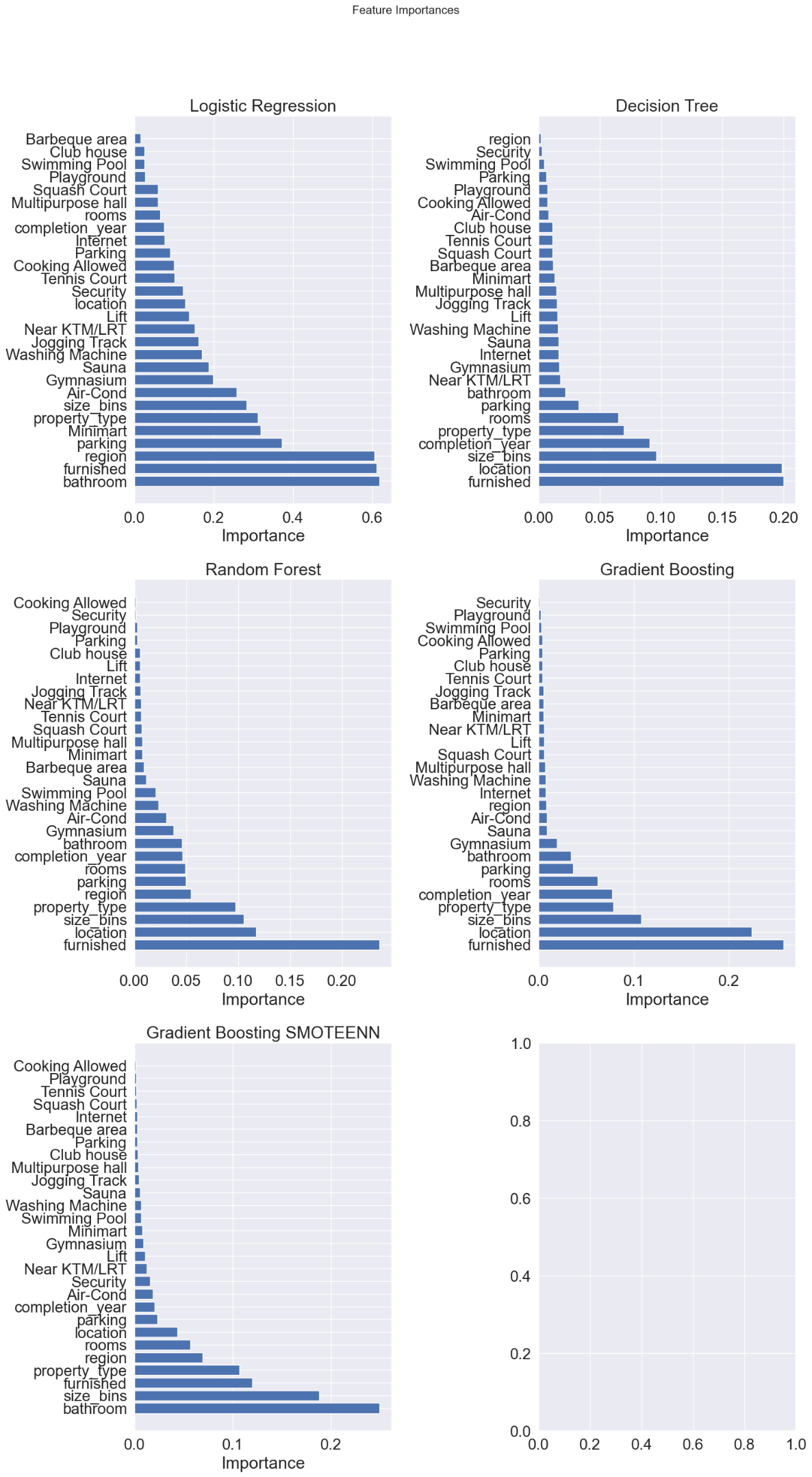
****

Diagram 8: Gradient Boosting Resampling (GBR 4) Feature Importance

From the graph above, the most significant feature is bathroom, size\_bins, furnished, property\_type, region, rooms, location, parking and completion\_year. The top facilities/amenities that influence the prediction are the Air-Cond, security, and nearby KTM/LRT.

**6. Answer the Problem**

**6.1 The Best Model**

From all the models, the Gradient Boosting with resampling model (GBR 4) is the best-performing model for this project due to higher overall accuracy, MCC, and ROC AUC score, higher precision and recall across categories and reduced overfitting.

1. **Train and Test Scores**:
   * **GBR 4** has the highest test score (0.8655) and train score (0.9960).
2. **Cross-validation Score**:
   * **GBR 4** also has the highest cross-validation score (0.9833), indicating good generalization.
3. **MCC (Matthews Correlation Coefficient) - Test**:
   * **GBR 4** has the highest MCC (0.7107), indicating a better balance between precision and recall for different classes.
4. **ROC AUC Score**:
   * **GBR 4** has the highest ROC AUC score (0.9535), indicating better overall performance across all thresholds.
5. **F1 Scores**:
   * **GBR 4** has the highest F1 Score on both the train (0.9960) and test (0.8661) sets.

**Conclusion**

**GB 4 (Gradient Boosting with SMOTEENN, Weight, and Parameters from GridSearchCV)** is the best model based on the evaluation metrics. It has the highest scores across most performance indicators, suggesting it is the most effective and well-balanced model for this project.

**6.2 Deployment**

I will set up a server on Linode, which is a cloud hosting provider. I will deploy the app using Nginx (a web server) and Gunicorn (a Python WSGI HTTP server). This combination allows the app to handle incoming requests efficiently.

For the domain, I will need to purchase one from a domain provider (such as Namecheap). Then, configure the domain’s nameservers to point to the Linode server. DNS records will be set up to connect the domain to the server’s IP address. You ensured that both the “www” subdomain and the root domain (without “www”) redirect to your server.

To obtain the SSL certificates, I will get them from Let’s Encrypt, a free certificate provider. SSL certificates will secure communication between clients (browsers) and the server. This ensures that data transmitted over HTTPS is encrypted and trustworthy.

* 1. **Usage**
     1. **Utilization**

This model will benefit people involved in the real estate field such as the real estate agent/negotiator, valuers to give simple indications to their customers and homeowners who wish to rent out their properties for a passive income but do not want to engage with the professionals, they may head to the app to check how much their properties’ worth monthly (rental) based on features of their properties.

On the website, the user can select the pre-defined type of property (for now the model is only limited to high-rise type of properties) (such as Serviced Residence, Condominium, Flat, Studio, etc)), size, type of furnishing (not furnished, partially furnished or fully furnished), number of bedrooms bathroom and parking and the location (within KL and Selangor only for now). A list of features (swimming pool, gymnasium, sauna, etc) is provided so that a user can tick any features that exist within their property. After filling in all the required info, a user can proceed to the “Calculate” button and the model will give its output in terms of a category and range such as Low: RM 1,800 to RM 2,500. The user can choose any price from the range.

This will simplify the work and be hassle-free and time-saving with a more accurate figure.

**6.4 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.

* 1. **Future Work/Improvement**

**Hyperparameter Tuning:** Further explore hyperparameter tuning for the Gradient Boosting model. Utilize techniques like GridSearchCV or RandomizedSearchCV to systematically evaluate different combinations of hyperparameters and identify the best configuration for the specific dataset.

**Ensemble Methods:** Consider using ensemble methods that combine multiple Gradient Boosting models. Techniques like bagging or boosting can lead to more robust models, potentially improving performance on imbalanced data.

**Regularization:** To overcome overfitting issue, explore various techniques including proper selection of evaluation metrics, resampling methods like oversampling and undersampling, employing ensemble methods like BalancedBaggingClassifier, and adjusting threshold values for optimal classification (Mazumder S, 2024).

**Feature Importance:** Train and evaluate the model based on the model’s feature importance result.

**Feature Engineering:** Creating new features from existing ones that might be more informative for the model to be trained. For example, creating a new feature from the existing features such as "building age". This might capture the combined effect of these factors on rent prices, where older buildings might have lower rents compared to newer ones.

**7. References**

‌Badra. (2024, June 19). *Maksud B40, M40, T20, miskin tegar – kategori pendapatan terkini*. eCentral. https://ecentral.my/maksud-miskin-tegar-b40-m40-t20/

Chee, J. (2024, January 24). *Median household income by state: How much do Malaysian households earn*. DollarsAndSense.my. https://dollarsandsense.my/median-household-income-by-state-how-much-do-malaysian-households-earn/

*Classification: ROC Curve and AUC*. (n.d.). Google for Developers. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Daython. (2022, December 8). 7 ESSENTIAL PLOTS FOR CATEGORICAL AND NUMERICAL DATA. *Medium*. https://medium.com/@daython3/7-essential-plots-for-categorical-and-numerical-data-4b96c0278106

‌Digifolio. (2023, August 13). Using Machine Learning in Real Estate: Predicting Rental demand. *Medium*. https://medium.com/@k.abouelnaga/using-machine-learning-in-real-estate-predicting-rental-demand-98c0e99e5430

GeeksforGeeks. (2021, January 22). *Hyperparameters of random forest classifier*. GeeksforGeeks. https://www.geeksforgeeks.org/hyperparameters-of-random-forest-classifier/

GeeksforGeeks. (2024, April 16). *How to tune a Decision Tree in Hyperparameter tuning*. GeeksforGeeks. https://www.geeksforgeeks.org/how-to-tune-a-decision-tree-in-hyperparameter-tuning/

GeeksforGeeks. (2024, June 20). *Logistic regression in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/understanding-logistic-regression/

GeeksforGeeks. (2024, March 11). *Bagging and random forest for imbalanced classification*. GeeksforGeeks. https://www.geeksforgeeks.org/bagging-and-random-forest-for-imbalanced-classification/

GeeksforGeeks. (2024, March 11). *Weighted logistic regression for imbalanced dataset*. GeeksforGeeks. https://www.geeksforgeeks.org/weighted-logistic-regression-for-imbalanced-dataset/

GeeksforGeeks. (2024, May 17). *Decision tree*. GeeksforGeeks. https://www.geeksforgeeks.org/decision-tree/

GeeksforGeeks. (2024, May 3). SMOTE for Imbalanced Classification with Python. GeeksforGeeks. https://www.geeksforgeeks.org/smote-for-imbalanced-classification-with-python/

Goyal, C. (2024, June 6). *Outlier Detection & Removal | How to detect & remove outliers (Updated 2024)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/05/feature-engineering-how-to-detect-and-remove-outliers-with-python-code/

*GradientBoostingClassifier*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier

Gustavo. (2021, December 7). Understanding Logistic Regression step by step - Towards Data Science. *Medium*. https://towardsdatascience.com/understanding-logistic-regression-step-by-step-704a78be7e0a

*How to explain the ROC AUC score and ROC curve?* (n.d.). https://www.evidentlyai.com/classification-metrics/explain-roc-curve

Izzuddin, M. (2018, October 14). Machine learning ensemble models for high rise residential in Kuala Lumpur. *Medium*. https://izzuddin8803.medium.com/machine-learning-ensemble-models-for-kuala-lumpurs-high-rise-residentials-2a9d91e0985e

*JABATAN PERANGKAAN MALAYSIA DEPARTMENT OF STATISTICS MALAYSIA*. (n.d.). https://www.dosm.gov.my/uploads/release-content/file\_20231118095531.pdf

Jain, A. (2022, October 10). End to End Machine Learning Project - Analytics Vidhya - Medium. *Medium*. https://medium.com/analytics-vidhya/end-to-end-machine-learning-project-a3543c89f6eb

Jayawardena, N. (2022, March 30). How to Deal with Imbalanced Data - Towards Data Science. *Medium*. https://towardsdatascience.com/how-to-deal-with-imbalanced-data-34ab7db9b100

Li, K., Tunku, U., & Rahman, A. (2020). *A DASHBOARD TO MONITOR HOUSING MARKET A REPORT SUBMITTED TO*. http://eprints.utar.edu.my/3866/1/17ACB04196\_FYP.pdf

‌Malaysia, J. (n.d.). *LAPORAN SURVEI PERBELANJAAN ISI RUMAH HOUSEHOLD EXPENDITURE SURVEY REPORT*. https://storage.googleapis.com/dosm-public-publications/multiyear\_2022\_hies.pdf

*matthews\_corrcoef*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews\_corrcoef.html

Mazumder, S. (2024, May 27). *5 Techniques to handle imbalanced data for a classification problem*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/

*MEF | News*. (n.d.). https://www.mef.org.my/news/general\_article.aspx?@ID=59

Misun\_Song. (2023, September 23). Understanding the ROC-AUC Curve - misun\_song - Medium. *Medium*. https://medium.com/@msong507/understanding-the-roc-auc-curve-cc204f0b3441

Rędzia, M. (2023, March 28). I built my own housing dataset and trained a ML model to predict rental prices. *Medium*. https://python.plainenglish.io/i-built-my-own-housing-dataset-and-trained-a-ml-model-to-predict-rental-prices-945b7d274ec6

*roc\_auc\_score*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc\_auc\_score.html

‌*RPubs - prediction of rental price\_2*. (n.d.). Rpubs.com. https://rpubs.com/DONGXUEYING/1046846

Satpathy, S. (2024, June 6). *SMOTE for Imbalanced Classification with Python*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/

Science, B. O. C., & Science, B. O. C. (2024, March 18). *F-1 score for Multi-Class Classification | Baeldung on Computer Science*. Baeldung on Computer Science. https://www.baeldung.com/cs/multi-class-f1-score

*sklearn.metrics*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/api/sklearn.metrics.html#module-sklearn.metrics

Smolic, H. (2024, January 9). Understanding the importance of F1 score in machine learning. *Graphite Note*. https://graphite-note.com/understanding-the-importance-of-f1-score-in-machine-learning-3/

Trevisan, V. (2022, March 24). Multiclass classification evaluation with ROC Curves and ROC AUC. *Medium*. https://towardsdatascience.com/multiclass-classification-evaluation-with-roc-curves-and-roc-auc-294fd4617e3a

Vaish, P. (2021, June 16). *Visualise categorical variables in Python - a data analyst*. A Data Analyst. https://adataanalyst.com/data-analysis-resources/visualise-categorical-variables-in-python/

*What is Matthews Correlation Coefficient (MCC)*. (n.d.). https://www.activeloop.ai/resources/glossary/matthews-correlation-coefficient-mcc/