# AI Assignment - 4

SHOBHIT RAJ
2022482

Q1.

Target (class) → 'Approve' or 'Reject'

$P(\text{approve}) = \frac{5}{10}$

$P(\text{reject}) = \frac{5}{10}$

Entropy $= -\sum_i p_i \log_2(p_i)$

$\therefore$ Entropy of dataset $= H = \left(\frac{-5}{10}\log_2\frac{5}{10} - \frac{5}{10}\log_2\frac{5}{10}\right)$

$= \frac{-1}{2}(-1) - \frac{1}{2}(-1) = \frac{1}{2} + \frac{1}{2} = 1.$

Calculating Information Gain for each feature:-

$$\text{Gain}(S, \text{feature}) = \text{Entropy}(S) - \underbrace{\text{Entropy}(S|\text{feature})}$$

$$\sum_{v \in \text{values}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

For Long Term Debt:

Weighted Entropy or Entropy(S|LongTermDebt) =

$$P(\text{LTD}=\text{Yes})\left(-\sum_{i=\text{approve or reject (given LTD=Yes)}} p_i \log_2 p_i\right) + P(\text{LTD}=\text{No})\left(-\sum_{i=\text{approve or reject (given LTD=No)}} p_i \log_2 p_i\right)$$

$$= \frac{5}{10}\left(-\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\frac{4}{5}\right) + \frac{5}{10}\left(-\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5}\right)$$

$\approx 0.721.$

For Unemployed:

$$\text{Entropy}(S|\text{Unemployed}) = \frac{2}{10}\left(\frac{-2}{2}\log_2\frac{2}{2} - 0\right) + \frac{8}{10}\left(\frac{-5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8}\right)$$

$\approx 0.763$

For Credit Rating:

$$\text{Entropy}(S|\text{CreditRating}) = \frac{3}{10}\left(\frac{-2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right) + \frac{7}{10}\left(\frac{-3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7}\right)$$

$\approx 0.965$

For Downpayment <20%:

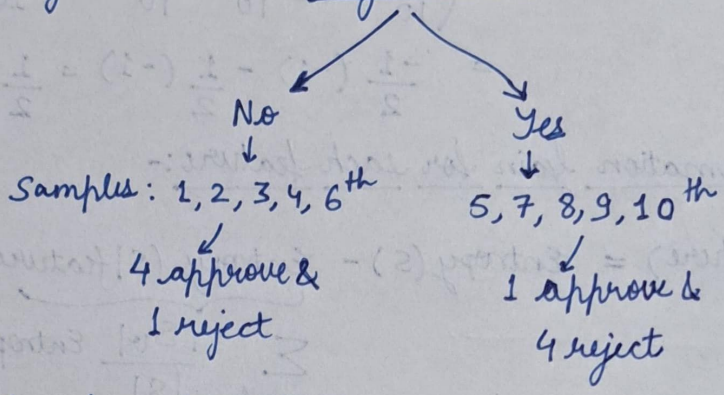
$$\text{Entropy}(S|\text{DP}<20\%) = \frac{5}{10}\left(\frac{-2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{5}{10}\left(\frac{-3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$$

$\approx 0.971$

∴ Information gain →

Gain (S, LTD) = 1 − 0.721 = 0.279 → highest IG.
Gain (S, Unemployed) = 1 − 0.763 = 0.237
Gain (S, CR) = 1 − 0.965 = 0.035
Gain (S, DP<20%) = 1 − 0.971 = 0.029

Splitting based on LongTermDebt :-




2nd split when LTD = "No" :-

$$\text{Entropy}(S) = \frac{-4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5} = 0.721$$

For Unemployed :

$$\text{Entropy}(S|UE) = \frac{4}{5}\left(\frac{-4}{4}\log_2\frac{4}{4} - 0\right) + \frac{1}{5}\left(-0 - \frac{1}{1}\log_2\frac{1}{1}\right) = 0$$

For Credit Range :

$$\text{Entropy}(S|CR) = \frac{2}{5}\left(\frac{-1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) + \frac{3}{5}\left(\frac{-3}{3}\log_2\frac{3}{3} - 0\right) = \frac{2}{5} = 0.4$$

For DP<20% :

$$\text{Entropy}(S|DP<20\%) = \frac{3}{5}\left(\frac{-2}{3}\log\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right) + \frac{2}{5}\left(\frac{-2}{2}\log\frac{2}{2} - 0\right) \approx 0.55$$

IG (S, LTD = "No") →

IG (S, Unemployed) = 0.721 − 0 = 0.721 → highest IG.
IG (S, CR) = 0.721 − 0.4 = 0.321
IG (S, DP<20%) = 0.721 − 0.55 = 0.171

Splitting on left side (LTD = No) based on Unemployed.

2nd split when LTD = "Yes" :-

$$\text{Entropy}(S) = \frac{-4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5} = 0.721$$

For Unemployed:

$$\text{Entropy}(S|UE) = \frac{1}{5}\left(-0 - \frac{1}{1}\log_2\frac{1}{1}\right) + \frac{4}{5}\left(\frac{-1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}\right)$$
$$\approx 0.649$$

For Credit Range:

$$\text{Entropy}(S|CR) = \frac{1}{5}\left(\frac{-1}{1}\log_2\frac{1}{1} - 0\right) + \frac{4}{5}\left(0 - \frac{4}{4}\log_2\frac{4}{4}\right) \approx 0$$

For DP < 20% :

$$\text{Entropy}(S|DP<20\%) = \frac{2}{5}\left(0 - \frac{2}{2}\log_2\frac{2}{2}\right) + \frac{3}{5}\left(\frac{-1}{3}\log\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right)$$
$$\approx 0.55$$

IG (S, LTD = "Yes") →

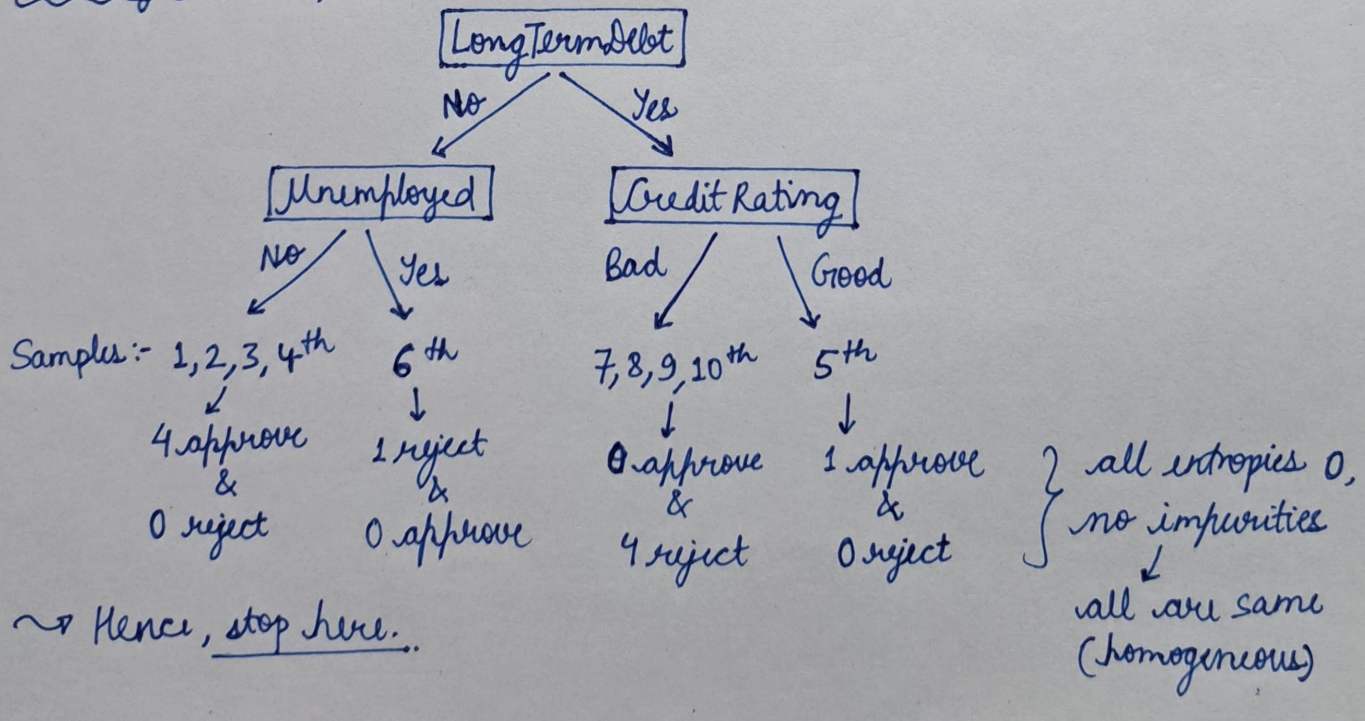
IG (S, Unemployed) = 0.721 − 0.649 = 0.072

IG (S, CR) = 0.721 − 0 = 0.721 → highest IG.

IG (S, DP < 20%) = 0.721 − 0.55 = 0.171

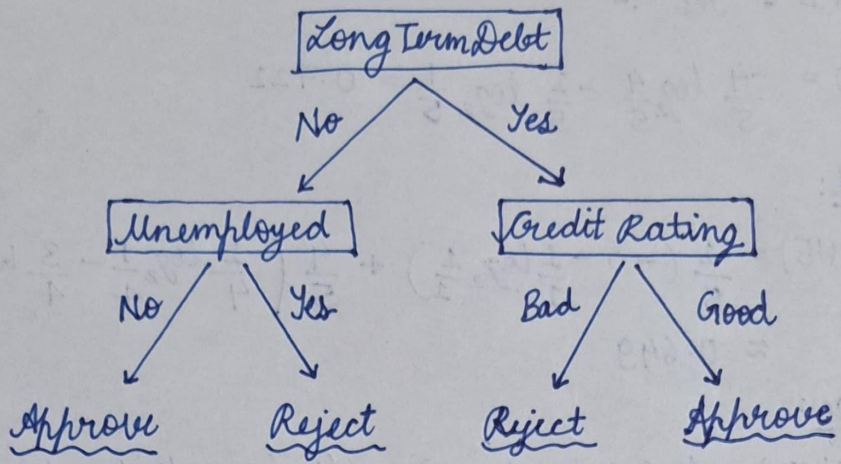
Splitting on right side (LTD = Yes) based on Credit Rating.

Making the 2nd split :-




↝ Hence, stop here.

Final Decision Tree :-




$$\text{Training error} = \frac{\text{Misclassified samples}}{\text{Total samples}}$$

As all samples are correctly classified, training error = 0%.

—— X ——

## Coding Questions

Q2. Data Preprocessing and Exploratory Data Analysis

1. Task 1: Understanding the Dataset

   EDA is conducted on the training dataset to ensure that the test data remains unseen, maintaining the integrity of the evaluation process.

   Dataset Overview: The dataset contains **15 columns** and **6,256 rows** with no missing values across all columns. It has 3 object type/categorical (string) columns that are Address, Possession, Furnishing and 12 numerical columns (8 float and 4 integers). The dataset occupies around 733 KB in memory.

   Analysis of Unique Values:

- **index:** Every row has a unique value, making this a potential primary key.
- **Address:** Contains 3,223 unique values, representing detailed property locations.
- **Possession:** All entries are "Ready to move", this feature can be removed as it is always same.
- **Furnishing:** Three distinct categories - Semi-Furnished, Unfurnished, and Fully Furnished.
- **Numerical Columns:** Vary significantly in their ranges, such as Buildup_area and Price and require further analysis of their distribution which is done using describe() in the next part.

   Descriptive Statistics for the numerical columns:

| | index | Buildup_area | Carpet_area | Bathrooms | Property_age | Parking | Price | Brokerage | Floor | Per_sqft_price | BHK | Total_bedrooms |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 6256.000000 | 6256.000000 | 6256.000000 | 6256.000000 | 6256.000000 | 6256.000000 | 6.256000e+03 | 6.256000e+03 | 6256.000000 | 6256.000000 | 6256.000000 | 6256.000000 |
| mean | 4879.818894 | 1120.690537 | 864.869801 | 1.968057 | 7.519661 | 1.298593 | 3.057852e+07 | 1.148133e+07 | 19.885595 | 23415.351551 | 2.159527 | 2.206878 |
| std | 2770.439333 | 735.147038 | 583.283918 | 0.911779 | 7.374092 | 0.797501 | 3.790301e+07 | 3.164281e+07 | 13.951480 | 13067.308580 | 1.002020 | 0.985628 |
| min | 1.000000 | 180.000000 | 150.000000 | 1.000000 | 1.000000 | 0.000000 | 7.800000e+05 | 0.000000e+00 | 2.000000 | 1440.000000 | 1.000000 | 1.000000 |
| 25% | 2494.750000 | 650.000000 | 475.000000 | 1.000000 | 2.000000 | 1.000000 | 1.050000e+07 | 1.000000e+05 | 10.000000 | 15657.500000 | 1.000000 | 2.000000 |
| 50% | 4920.500000 | 950.000000 | 708.315583 | 2.000000 | 5.000000 | 1.000000 | 1.920000e+07 | 2.500000e+05 | 16.000000 | 21355.000000 | 2.000000 | 2.000000 |
| 75% | 7276.250000 | 1325.000000 | 1050.000000 | 2.000000 | 10.000000 | 2.000000 | 3.500000e+07 | 1.100000e+07 | 23.000000 | 28792.500000 | 3.000000 | 3.000000 |
| max | 9546.000000 | 15000.000000 | 14000.000000 | 10.000000 | 99.000000 | 9.000000 | 5.000000e+08 | 5.000000e+08 | 99.000000 | 100000.000000 | 10.000000 | 10.000000 |

2. Task 2: Drop Irrelevant Columns
   In this task, columns that were either irrelevant or lacked predictive power were removed based on their correlation with the target variable, Price, and other considerations. Correlation analysis was conducted for numerical columns, revealing that index and Property_age had very low correlation coefficients (0.0536 and 0.0696, respectively) with Price, falling within the specified range of -0.1 to 0.1. These columns were deemed insignificant and were removed.

   Additionally, the address column was excluded because it contained textual data with many unique values, making it unsuitable for predictive modeling. As it represents descriptive information, it is unlikely to contribute meaningfully to price prediction. The possession column was also removed as it contained only one unique value across all rows, offering no variability or predictive value.

3. Task 3: Encoding Categorical Features
   In this task, the categorical column Furnishing, which had three unique values, was label-encoded successfully. Label encoding was suitable for this feature due to its low cardinality, ensuring efficient transformation without introducing unnecessary complexity.

   High cardinality, where a categorical variable contains many unique values, poses challenges in data modeling. It can increase model complexity, may introduce noise or lead to overfitting, and require significant memory and computational resources. For example, the column Address had high cardinality with numerous unique entries, making it unsuitable for direct encoding. Mitigation strategies include grouping rare categories into an "Other" class, using frequency or target encoding, or removing such columns if they do not contribute significantly to the target variable.

   In this case, the Address column was removed in the previous task as it not only had high cardinality but also lacked a meaningful contribution to predicting the target variable.

4. Task 4: Feature Scaling

   In this task, numerical features will be scaled using the Standard Scaler to ensure they have a mean of 0 and a standard deviation of 1. Standardization helps in normalizing data distribution and ensures all features contribute equally to the model, avoiding dominance of features with larger scales.

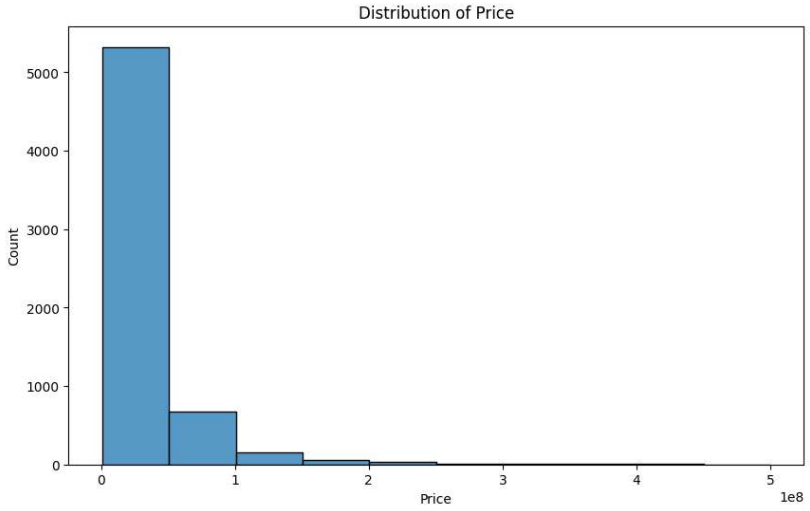
   *The effects of scaling (in code, this is done at last just before the bonus section) :-

   Training and testing a Decision Tree Regressor on scaled data resulted in an RMSE of approximately **2.74M** and an $R^2$ score of **0.9936** on the test set. Comparing this to the unscaled data performance, which achieved an RMSE of **2.67M** and an $R^2$ of **0.9941**, the difference in performance is minimal. Decision Trees are inherently insensitive to feature scaling due to their hierarchical splitting based on feature values rather than distances or magnitudes.

   Scaling numerical features does not significantly improve or harm the Decision Tree's performance, confirming that scaling is generally unnecessary for such models.

5. Task 5: Target Variable Imbalance Detection

   Distribution of target variable Price :-




   The distribution of the target variable, Price, was analyzed by categorizing it into four fixed price brackets: **Low** (below ₹1 crore), **Medium** (₹1 crore to ₹10 crore), **High** (₹10 crore to ₹20 crore), and **Very High** (above ₹20 crore). The categorized distribution revealed significant imbalance across the defined price categories.

Distribution of Price Categories :-


The Medium price category overwhelmingly dominates the dataset, comprising approximately 73.18% of the properties, followed by the Low category at 22.23%. In contrast, the High and Very High price categories constitute only 3.63% and 0.96% of the dataset, respectively. This stark imbalance highlights the limited representation of higher-value properties in the dataset.
Such skewed distribution can pose challenges, as the model may underperform in predicting the rarer **High** and **Very High** categories. Strategies like stratified sampling, data augmentation for rare categories, or weighted loss functions in model training can be employed to mitigate these effects.

6. Task 6: Handling Imbalanced Data

Original Class Distribution:

Medium - 4578
Low - 1391
High - 227
Very High - 60

To address the imbalance in price categories, random undersampling reduced all categories to 60 samples each, ensuring equal representation but with potential information loss. Conversely, random oversampling increased all categories to 4,578 samples, balancing the data while preserving the original distribution but risking overfitting.

**Random Undersampling**

**Benefits:** Efficient for large datasets as it reduces the dataset size, leading to faster training and processing times & improves model performance as more balanced class distributions.

**Limitations:** Risks losing important information from the majority class, which can lead to biased or inaccurate models & can result in overfitting.

**Random Oversampling**

**Benefits:** Ensures no data loss by replicating minority class examples, preserving all original data points. & improves model performance as more balanced class distributions.

**Limitations:** Increases the dataset size, leading to longer training times and higher memory requirements & risks overfitting as it duplicates existing minority class examples, which might not add meaningful diversity.

*The analysis of sampling techniques on model performance by evaluating the accuracy of predicted price categories using Decision Tree Regressor across default, undersampled, and oversampled datasets (in code, this is done at last just before the bonus section) :-

The analysis of sampling techniques revealed notable differences in model performance. Without sampling, the model achieved excellent results with an RMSE of **2,669,927** and $R^2$ of **0.9939** on the test data, and a high accuracy of **98.72%** for price category predictions.

For the undersampled model, while training performance was perfect, test performance dropped significantly with an RMSE of **13,797,027**, $R^2$ of **0.8371**, and accuracy of **90.60%**, indicating underfitting due to reduced data diversity.

The oversampled model maintained a balance, achieving an RMSE of **3,237,508**, $R^2$ of **0.9910**, and test accuracy of **98.59%**, closely matching the default performance but slightly lower than the no-sampling case.

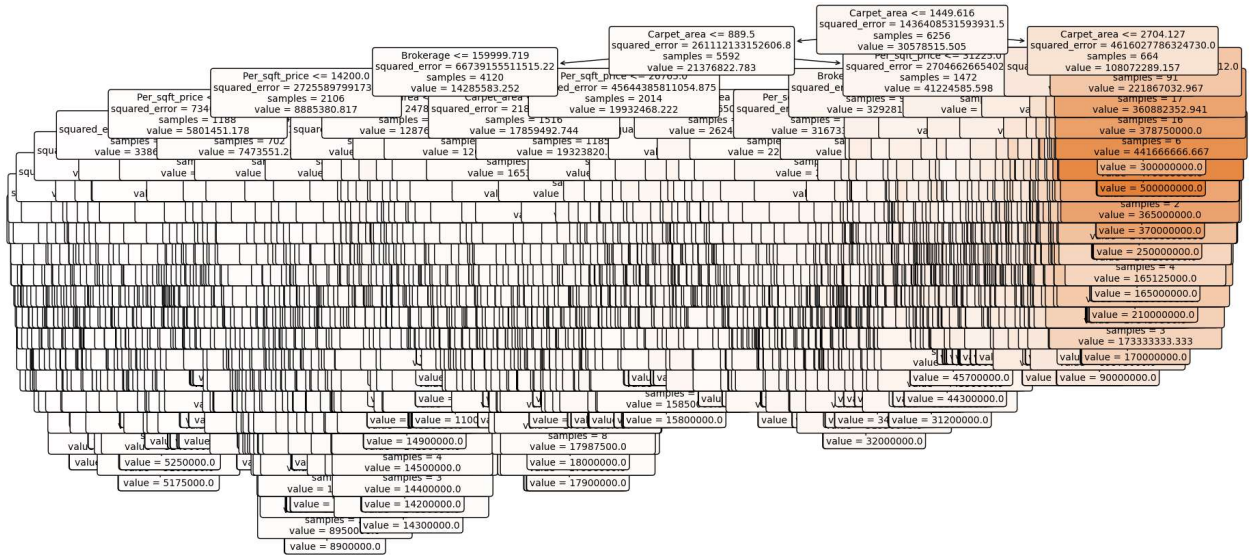
This suggests that oversampling better preserves data diversity, minimizing performance loss, whereas undersampling sacrifices generalization to achieve balance, which can hinder model accuracy.

Q3. Building Decision Tree Model

1. Task 1: Model Training

   (a) Decision Tree Regressor trained using training data.

   (b) Decision Tree Structure Visualization:

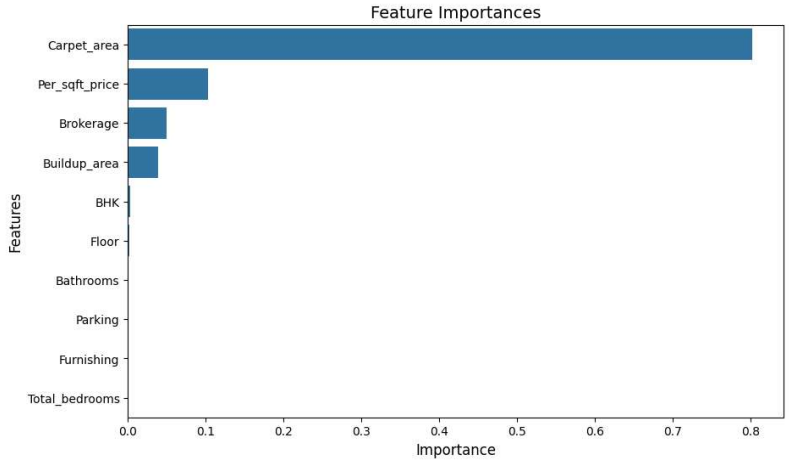



The depth of the tree is 24 & number of leaves are 3548.

The Decision Tree Regressor splits the data at each node based on feature thresholds to minimize variance in the target variable within each split. The depth of the tree reflects its complexity, with deeper trees capturing more detailed patterns but potentially risking overfitting.

2. Task 2: Feature Importance and Hyperparameter Tuning

   (a) Feature Importances :-

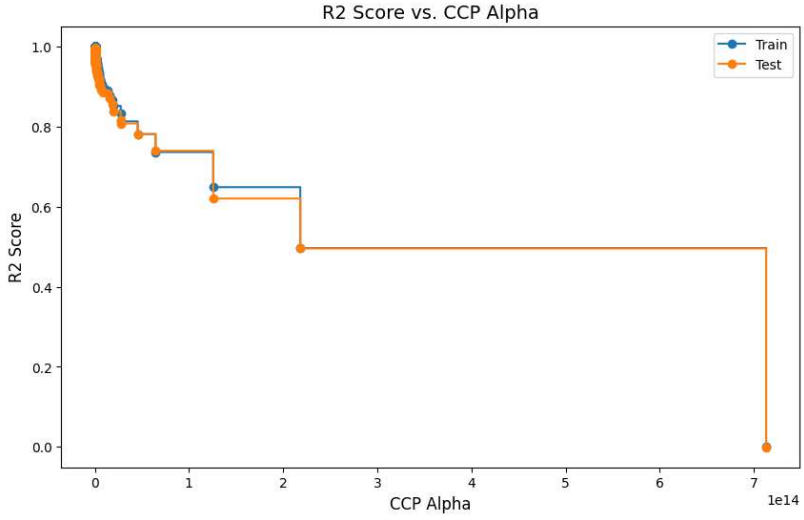


(b) The feature importance analysis reveals that “Carpet Area” and “Per Sqft Price “are the most influential factors in predicting property prices, which aligns with expectations as these features directly relate to a property's valuation. Lesser importance of features like “Total Bedrooms” and “Furnishing” is reasonable since they contribute indirectly or have weaker correlations with the target variable. This distribution supports the intuition that square footage and pricing metrics are primary drivers in real estate pricing models.

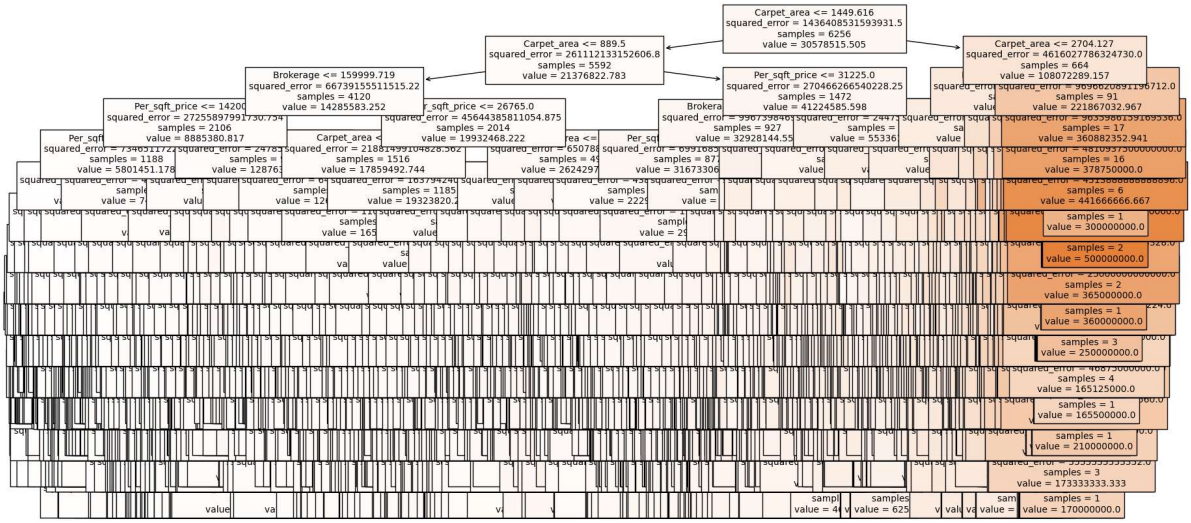
(c) The hyperparameter tuning process yielded the best parameters: **max_depth**=15, **max_features**=None, **min_samples_leaf**=1, and **min_samples_split**=2. The tuned model slightly improved performance on the test data compared to the default model. The test RMSE decreased from 2,669,927 to 2,621,767, and the $R^2$ score improved from 0.9939 to 0.9941. This indicates that hyperparameter tuning helped enhance the model's generalization by reducing overfitting, as seen in the more balanced performance between the training and testing datasets. However, the improvement was marginal, suggesting the default parameters were already near optimal for this dataset.

3. Task 3: Pruning Decision Tree

(a) Using minimal cost-complexity pruning with the optimal alpha value of **26,641,091.22**, the pruned decision tree achieved a more compact structure while maintaining strong predictive performance. The pruned model had an **RMSE of 137,451.17** and an **$R^2$ score of 0.9999868** on the training data, and an **RMSE of 2,618,867.31** with an **$R^2$ score of 0.994129** on the test data. These metrics are comparable to the unpruned tree but indicate slightly better generalization on the test set.



(b) Pruned Decision Tree Structure Visualization:



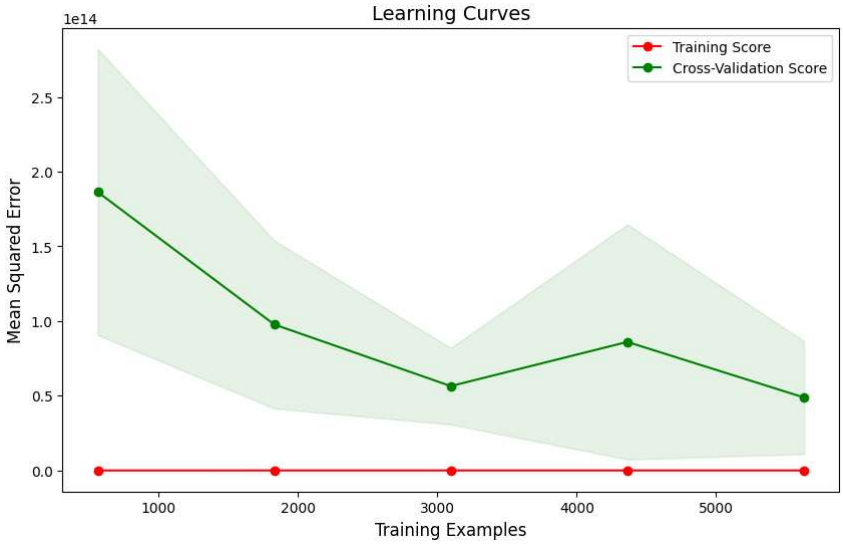

The pruned tree is simpler and shallower than the unpruned one, reducing its depth and complexity while retaining the same critical feature importances. This improved interpretability and reduced overfitting without compromising predictive accuracy, emphasizing the effectiveness of pruning in enhancing the tree's robustness and usability.

4. <u>Task 4: Handling Overfitting</u>

(a) Cross-Validation to Assess Model Generalization - The cross-validation $R^2$ scores ranged from 0.924 to 0.993, with a mean score of 0.9663 and a standard deviation of 0.0241. This indicates that the model has strong predictive power across different folds, with minor variability in performance, suggesting good generalization and minimal overfitting.

P.T.O.

(c) Learning Curves :-




The learning curves revealed that the training error quickly reduced to zero, indicating the model overfits the training data. The validation error initially decreased, then increased slightly before stabilizing at its lowest value. This fluctuation suggests a tendency to overfit during intermediate stages but demonstrates overall good performance on unseen data.

(d) Cross-validation ensures that the model is evaluated on multiple dataset splits, minimizing the risk of overfitting to a single train-test split. It helps identify hyperparameter settings that balance training performance with generalization, thus improving the robustness of the Decision Tree model.

P.T.O.

Q4. Model Evaluation and Error Analysis

1. Task 1: Model Evaluation
   (a) The model used for evaluation was the final Decision Tree Regressor, which was both hyperparameter-tuned and pruned for optimal performance.
   (b)

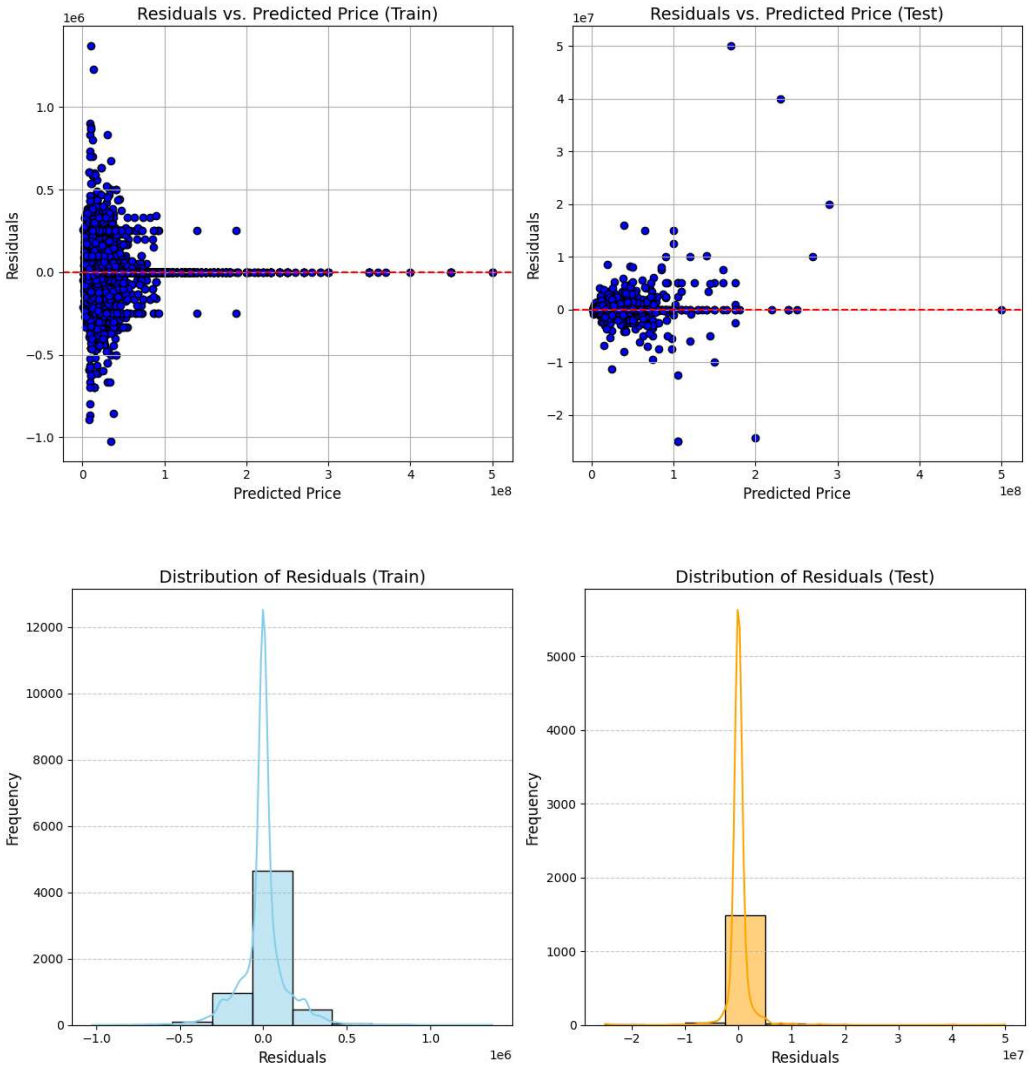
   Performance Metrics of the Best Model on Train Data:
   Mean Squared Error: 18892825564.041683
   Mean Absolute Error: 77126.71971487999
   R2 Score: 0.9999868471781193

   Performance Metrics of the Best Model on Test Data:
   Mean Squared Error: 6858466025332.059
   Mean Absolute Error: 823675.7155864021
   R2 Score: 0.994129379568809

   The low Mean Squared Error (MSE) and Mean Absolute Error (MAE) on both datasets indicate that the model is accurately predicting house prices, with significantly smaller errors on the training set. The difference in performance between the training and test data is expected, and the relatively small decrease in $R^2$ on the test set suggests that the model has good generalization capabilities, despite minor overfitting during training.

2. Task 2: Residual and Error Analysis
   (a) On the training data, the residuals are centered around zero, with a mean close to zero and a standard deviation of 137,462. The residuals range from -1,025,000 to 1,370,588, and the median residual is 0, indicating that the model predictions are balanced around the central tendency. However, for the test data, the residuals show a larger mean (163,717.9) and a much higher standard deviation (2,614,581). This points to some bias and significant errors when predicting prices on unseen data. Extreme residuals in the test set range from -25,000,000 to 50,000,000, highlighting the model's challenges in handling outliers or very high prices.

(b)




The residuals vs. predicted price plots reveal that errors increase with higher prices, showing the model's difficulty in maintaining consistent accuracy across the price range. This issue is more evident in the test data, where underpredictions for high-priced properties are prominent. The residual distribution plots further highlight these challenges, with a more skewed and dispersed pattern for test data residuals compared to training data.

The model underperforms for high-priced properties and outliers, likely due to their underrepresentation in the training data. To improve, applying a log transformation to the target variable could stabilize variance, and better handling of outliers, such as removal or adjustment, may help. Adding more relevant features and trying advanced models like Gradient Boosting could also enhance predictions. These steps can address the model’s limitations and improve its performance on unseen data.

3. Task 3: Feature Importance based analysis

   I examined the top 3 important features (Carpet_area, Per_sqft_price, and Brokerage) and trained separate models using each feature individually. I calculated the RMSE for both the training and test datasets.

   The analysis of the top three important features, Carpet_area, Per_sqft_price, and Brokerage, revealed varying predictive power for the target variable Price. Among them, Brokerage had the lowest RMSE on the training dataset (6.31M), indicating its strong fit on training data. However, Carpet_area exhibited better generalization with the lowest RMSE on the test dataset (12.16M), highlighting its critical role in predicting Price. Per_sqft_price had the highest RMSE on both datasets, especially on the test set (28.50M), suggesting its predictive power is weaker when used independently.

   These results suggest that Carpet_area plays the most critical role in predicting Price, while the other two features are more complementary in their impact.

P.T.O.

Q5. Bonus Challenge

1. Task 1: Advanced Imbalance Handling
   The original dataset showed significant class imbalance, particularly in the "High" and "Very High" price categories. Both SMOTE and ADASYN techniques were employed to address this issue, balancing all categories to approximately 4578 samples.

- **SMOTE:** Achieved perfect fit on the training data with RMSE 0.0 and $R^2$ 1.0. Test data performance showed RMSE of ~4.07M and $R^2$ of 0.9858. Classification accuracy was 100% on training data and 98.40% on test data.
- **ADASYN:** Like SMOTE, ADASYN achieved a perfect fit on training data with RMSE 0.048 and $R^2$ 1.0. Test data performance improved slightly with RMSE ~3.44M and $R^2$ 0.9899. Classification accuracy was 100% on training data and 98.59% on test data.

  Both methods successfully balanced the data and improved the model's ability to predict minority categories. ADASYN demonstrated slightly better test performance, likely due to its adaptive nature, generating synthetic samples that focus on harder-to-learn areas. This can lead to better generalization for borderline and minority cases. Overall, ADASYN marginally outperformed SMOTE in handling the dataset's imbalance.

2. Task 2: Ensemble Learning: Random Forest
   The Decision Tree Regressor outperformed the Random Forest Regressor in terms of model performance. On the test data, the Decision Tree achieved an RMSE of 2,669,718 and an $R^2$ score of 0.9939, while the Random Forest had a higher RMSE of 3,251,998 and a lower $R^2$ of 0.9909. Although Random Forest improved generalization by averaging over multiple trees, it resulted in slightly worse performance compared to the Decision Tree on this dataset. Feature importance analysis indicated that Carpet_area, Per_sqft_price, and Brokerage were again the most important features.

   The single Decision Tree, while prone to overfitting, provided better performance in this case due to its simpler structure and lower computational cost. Random Forest, though more robust and less prone to overfitting, has the trade-off of increased complexity and a slight reduction in accuracy in this scenario. The decision between using a single tree or an ensemble depends on the dataset and the need for interpretability vs. accuracy.

Overall Results :-

For Price as Target (Regression Task) :-

| Model Type | Train RMSE | Train R2 Score | Test RMSE | Test R2 Score |
|---|---|---|---|---|
| Default Model | 0.0 | 1.0 | 2,669,927.07 | 0.9939 |
| Best Model (Hyperparameter Tuning) | 105,971.34 | 0.99999 | 2,621,767.30 | 0.9941 |
| Pruned Model | 137,451.18 | 0.99999 | 2,618,867.32 | 0.9941 |
| Scaled Data Model | 8.94 | 1.0 | 2,743,451.30 | 0.9936 |
| Undersampled Model | 0.0 | 1.0 | 13,797,027.04 | 0.8371 |
| Oversampled Model | 0.0 | 1.0 | 3,237,507.89 | 0.9910 |
| SMOTE Model | 0.0 | 1.0 | 4,074,013.85 | 0.9858 |
| ADASYN Model | 0.05 | 1.0 | 3,440,247.13 | 0.9899 |
| Random Forest Model | 2,249,864.57 | 0.99648 | 3,251,998.82 | 0.9909 |

For Price_Category as Target (Classification Task) :-

| Model Type | Train Accuracy | Test Accuracy |
|---|---|---|
| Without Sampling | 1.0 | 0.9872 |
| Undersampled Model | 1.0 | 0.9060 |
| Oversampled Model | 1.0 | 0.9859 |
| ADASYN Model | 1.0 | 0.9859 |
| SMOTE Model | 1.0 | 0.9840 |
| Random Forest Model | 0.9915 | 0.9904 |