Page No.
Date : / /

# Assignment - 4

$$H(S) = -\sum_{i=1}^{n} P_i \log_2(P_i)$$

For root node,

|Approved| = 5

|Rejected| = 5

$$H_{root} = -\left(\frac{5}{10}\log\frac{5}{10} + \frac{5}{10}\log\frac{5}{10}\right) = -(-0.5 + (-0.5)) = +1$$

Next we calculate the information gain for each attribute,

① long term debt :-

No : 5 instances (4 approve, 1 reject)

Yes : 5 instances (1 approve, 4 reject)

$$H_{No} = -\left(\frac{4}{5}\log\frac{4}{5} + \frac{1}{5}\log\frac{1}{5}\right) = -(0.8 \cdot (-0.322) + 0.2(-2.322))$$

$$H_{yes} = -\left(\frac{1}{5}\log\frac{1}{5} + \frac{4}{5}\log\frac{4}{5}\right) = -(0.2 \cdot (-2.322) + 0.8(-0.322)) = 0.722$$

Weighted average entropy:

$$H_{long\text{-}term} = \frac{5}{10}\cdot(0.722) + \frac{5}{10}(0.722) = 0.722$$

Information gain:

$$IG = H_{root} - H_{long\text{-}term} = 1 - 0.722 = 0.278$$

② Unemployed :→

↳ No : 8 instances (5 approve, 3 reject)

↳ Yes : 2 instances (0 approve, 2 reject)

$$H_{No} = -\left(\frac{5}{8}\log\frac{5}{8} + \frac{3}{8}\log\frac{3}{8}\right) = -(0.625(-0.678) + 0.375(-1.415)) = 0.954$$

$$H_{yes} = -\left(\frac{0}{2}\log_2\frac{0}{2} + \frac{2}{2}\log_2\frac{2}{2}\right) = 0$$

Date : / /

Weighted avg. entropy:

$$H_{unemployed} = \frac{8}{10} \cdot (0.954) + \frac{2}{10} \cdot (0) = 0.763$$

Information Gain:

$$IG = H_{root} - H_{unemployed} = (-0.763 - 0.237)$$

③ Credit rating:-

Good : 4 instances (3 approve, 1 reject)

Bad : 6 instances (2 approve, 4 " )

$$H_{good} = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) = 0.811$$

$$H_{Bad} = -\left(\frac{2}{6}\log\frac{2}{6} + \frac{4}{6}\log_2\frac{4}{6}\right) = 0.918$$

$$H_{credit\ rating} = \frac{4}{10}(0.811) + \frac{6}{10}(0.918) = 0.875$$

$$IG = H_{root} - H_{Credit\ rating} = 1 - 0.875 = 0.125$$

④ Down payment < 20%

$$H_{yes} = -\left(\frac{2}{5}\log_2\frac{2}{5} + \frac{3}{5}\log_2\frac{3}{5}\right) = 0.971$$

$$H_{No} = 0.971$$

$$H_{Down\ payment} = \frac{5}{10}(0.971) + \frac{5}{10}(0.971) = 0.971$$

$$IG = 1 - 0.971 + 1 = 0.029$$

Best attribute for root node ⇒ long term debt

**Root = long term debt**

No ⇒ 4 approve, 1 reject ($H=0.722$)
Yes ⇒ 1 approve, 4 " " ($H=0.722$)

Next, we keep on splitting till we find a node that splits purely.

long Term Debt = No :⟶

① Credit rating :-

$H_{good} = 1$, $H_{Bad} = 0$

$H_{credit\ rating} = \frac{2}{5} \cdot 1 + \frac{3}{5} \cdot 0 = 0.4$

$IG = 0.722 - 0.4 = 0.322$

② Subset for credit rating = Good.

$H = 1$

③ Split on down payment

$H = -\left(\frac{1}{1}\log\frac{1}{1} + \frac{0}{1}\log\frac{0}{1}\right) = 0$

This gives us a pure subset.

④ Subset for C.R = Bad

$H = 0$

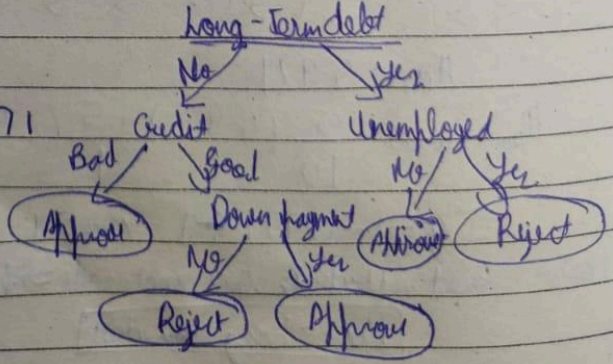
This is also pure so we stop.

long-Term debt = Yes :-

$H_{No} = 0.918$, $H_{yes} = 0$

$H_{unemployed} = 0.551$

$IG = 0.722 - 0.551 = 0.171$



# CODING:

## Explanation for Dropping Irrelevant Columns:

1. **Correlation Analysis**:
    - During the correlation analysis, we observe the relationship between each feature (independent variable) and the target variable. Features that have a correlation coefficient between **-0.1 and 0.1** with the target variable are considered to have a weak relationship. These features are unlikely to provide significant predictive power for the model, as their relationship with the target variable is minimal.
    - **Reason for Dropping**: Dropping features with weak correlations helps improve model efficiency and reduces noise, as these features do not meaningfully contribute to the target variable's prediction.
2. **Lack of Predictive Power**:
    - In addition to weak correlations, some features might be inherently irrelevant to the target variable. For example, features that are **constant** (i.e., the same value for all rows) or features that are **too specific or unrelated** to the target might not add any value to the model.
    - **Reason for Dropping**: These features contribute to overfitting (by increasing model complexity without improving performance) and may also reduce the model's ability to generalize.
3. **Domain Knowledge**:
    - Based on domain knowledge or previous analysis, certain features might be identified as irrelevant. For example, columns that may capture demographic information unrelated to the prediction of prices (like "ID" or "Date of birth" for a product price prediction) might not hold predictive power.
    - **Reason for Dropping**: These columns are unlikely to add valuable information to the model and can be removed to reduce the model's complexity.

```
Reasons for dropping columns:
- 'index' has low correlation (0.05) with the target variable.
- 'Possesion' is irrelevant due to lack of variability or excessive
missing values.
- 'Property_age' has low correlation (0.07) with the target variable.
Columns to drop: ['index', 'Possesion', 'Property_age']
```

## Explanation for Encoding Categorical Features using Label Encoding:

High cardinality refers to categorical variables with a large number of unique values. This can lead to challenges in machine learning models, including:

1. **Increased Complexity**: High cardinality results in a large number of encoded features, which can increase model complexity and computation time.
2. **Overfitting**: If a categorical variable has too many unique values, the model may overfit to specific instances, especially with small datasets.

## Mitigation Strategies:

- **Grouping Rare Categories**: Combine infrequent categories into a single "Other" category to reduce cardinality.
- **Target Encoding**: Replace categories with statistical values like the mean target for each category, rather than using one-hot or label encoding.
- **Dimensionality Reduction**: Use techniques like PCA (Principal Component Analysis) to reduce the number of features.

By addressing high cardinality, models can be more efficient and generalizable.

## Imbalance Detection:

### Handling data imbalance:

## Random Undersampling:

**Definition**: Random undersampling involves reducing the number of instances from the majority class to balance the dataset.

**Benefits**:

- **Faster Model Training**: Reduces the size of the dataset, leading to quicker training times.
- **Simpler Models**: A balanced dataset can improve the model's ability to focus on minority class without being overwhelmed by the majority class.

**Limitations**:

- **Loss of Information**: Reducing the majority class can result in the loss of potentially valuable data, which may affect model performance.
- **Underfitting**: If too many examples are removed, the model might not learn the general patterns effectively.

---

## Random Oversampling:

**Definition**: Random oversampling involves duplicating instances from the minority class to balance the dataset.

**Benefits**:

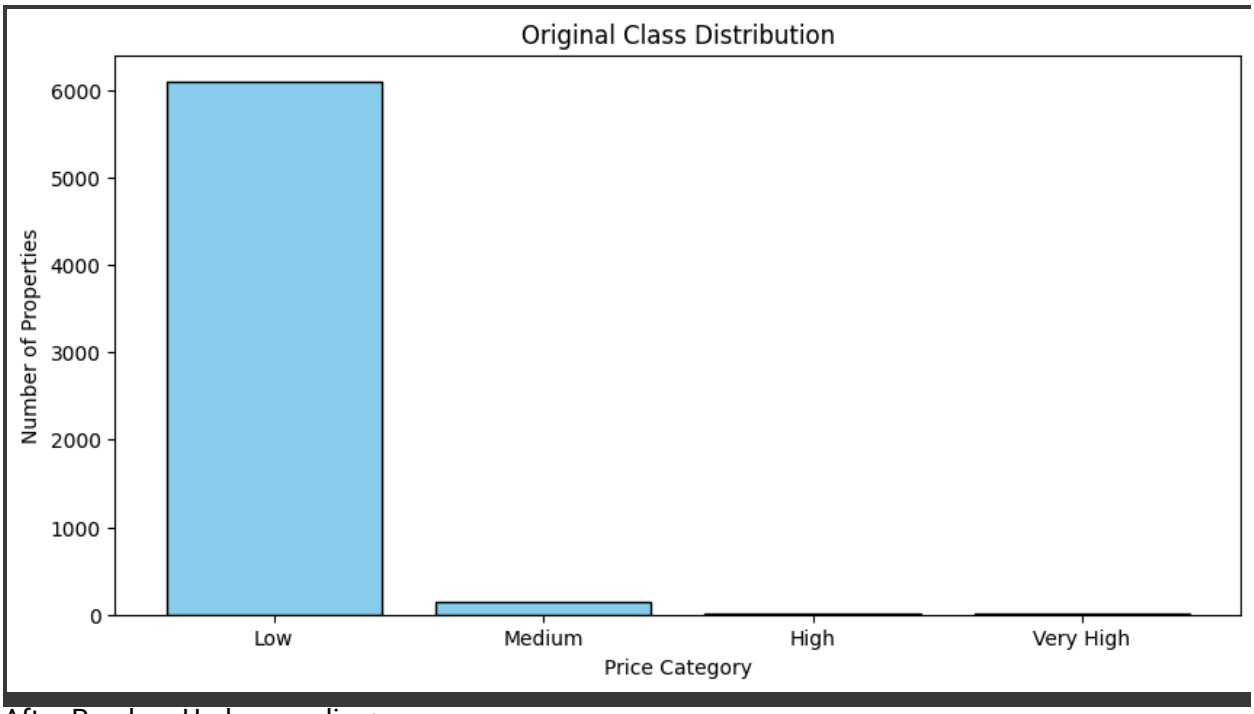
- **Retains All Data**: Unlike undersampling, no data from the majority class is lost.
- **Better Performance for Minority Class**: Helps the model to learn better about the minority class, improving its prediction.

**Limitations**:

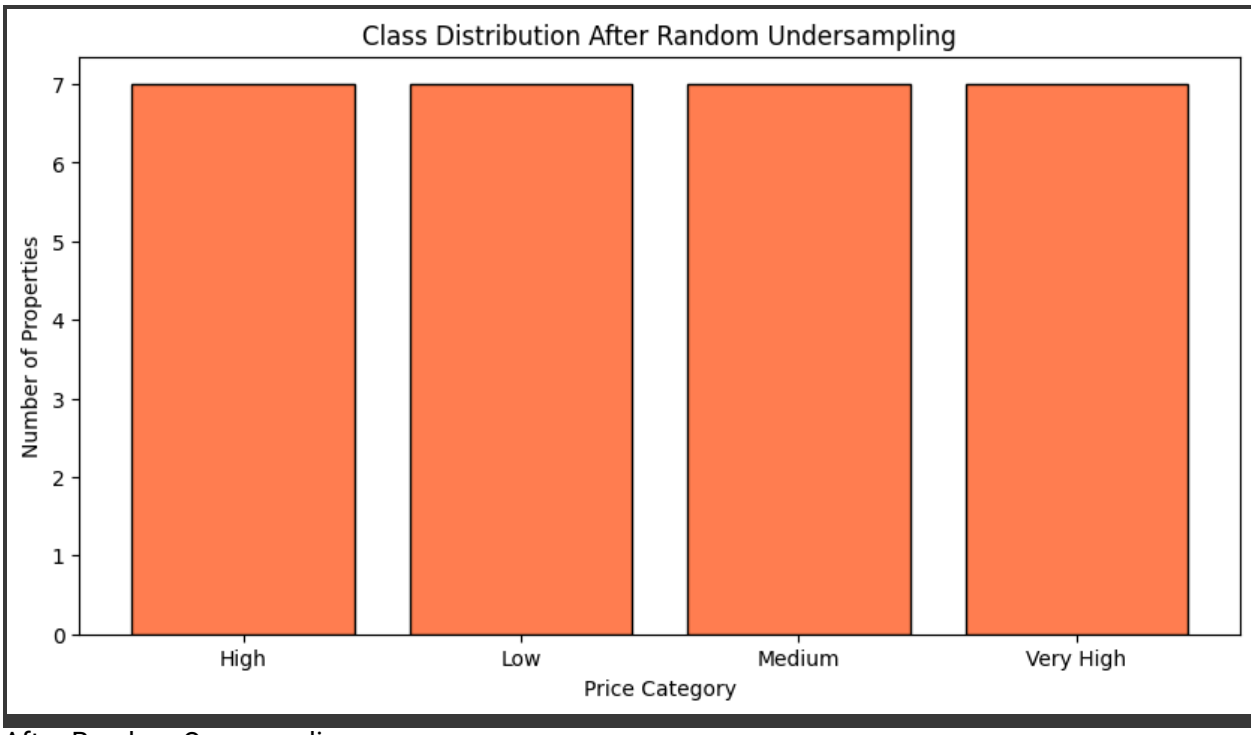
- **Overfitting**: By duplicating minority class examples, the model may overfit to these repeated instances, causing poor generalization.
- **Increased Computation Time**: More data results in longer training times, especially when oversampling significantly increases the dataset size.

In summary, **undersampling** is useful when you want to reduce data size but risks losing information, while **oversampling** helps to balance the dataset without losing data but may lead to overfitting and increased training time.

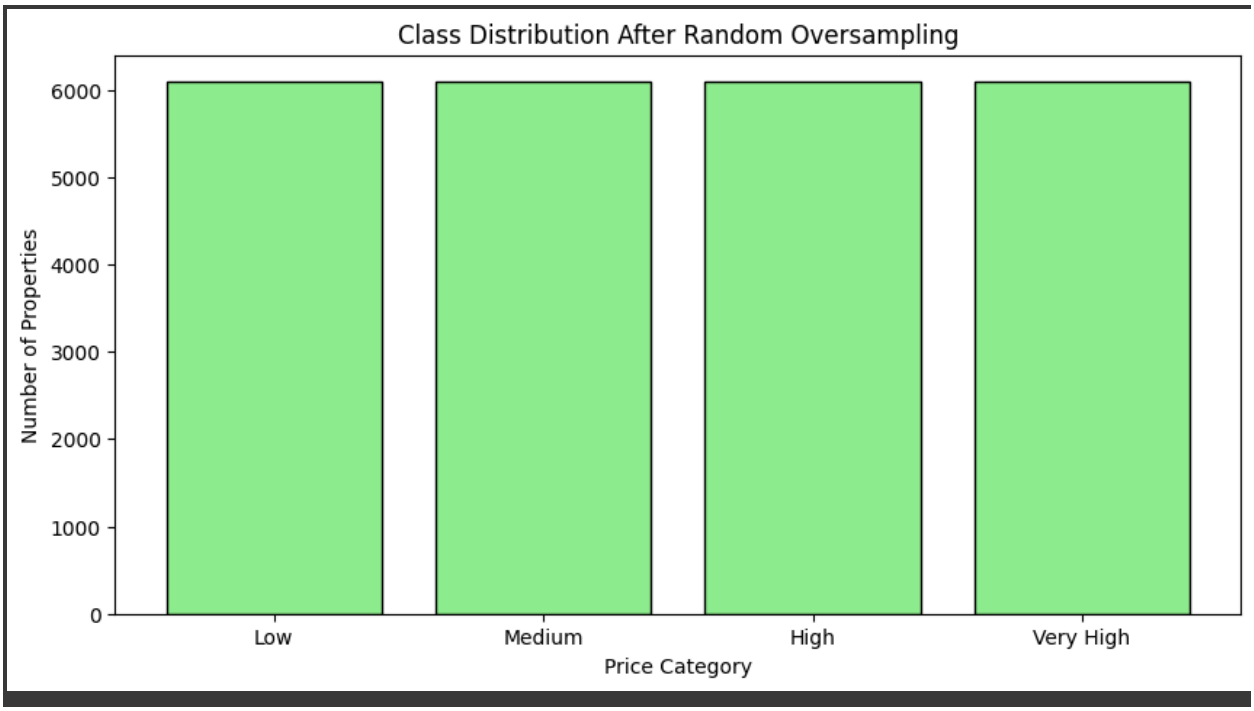
Original class distribution: Counter({'Low': 6092, 'Medium': 139, 'High': 18, 'Very High': 7})



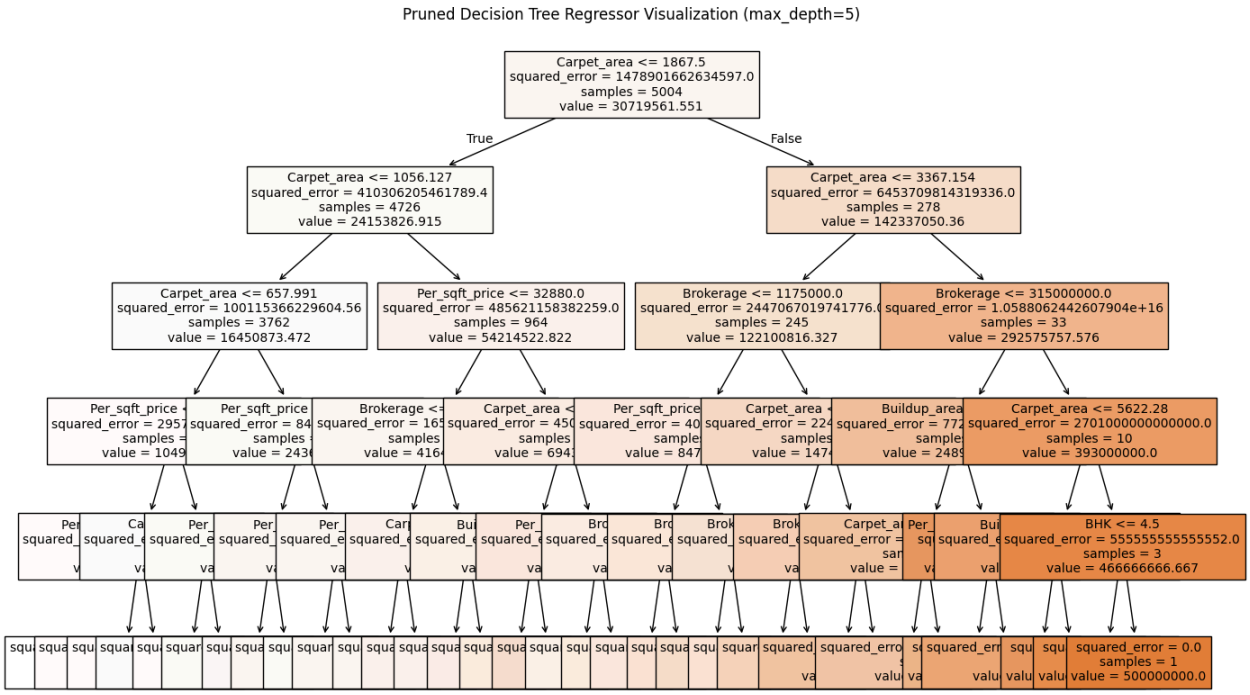

After Random Undersampling:
Class distribution: Counter({'High': 7, 'Low': 7, 'Medium': 7, 'Very High': 7})




After Random Oversampling:
Class distribution: Counter({'Low': 6092, 'Medium': 6092, 'High': 6092, 'Very High': 6092})

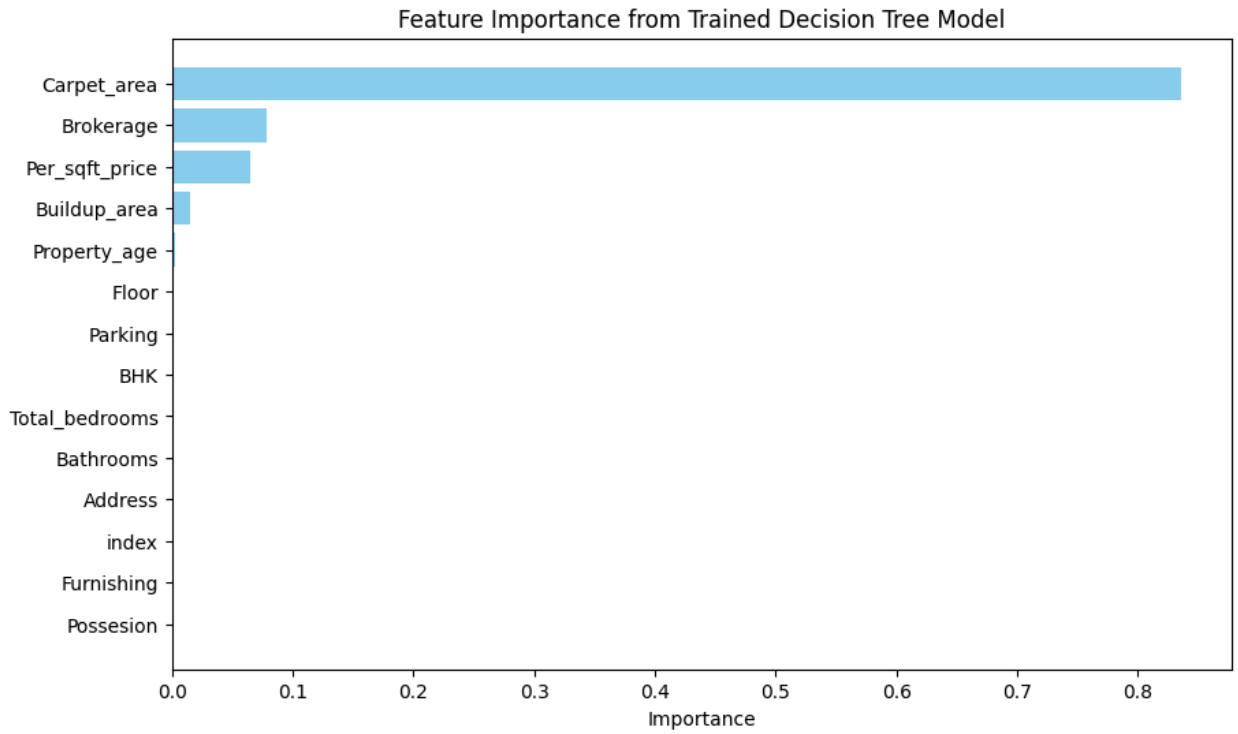



## DECISION TREE VISUALIZATION




The decision tree had 22 levels and it could not be visualized clearly in one frame so I have

reduced the depth to 5. But, in code I have added the the full tree with 22 levels and 5 levels.So please refer to code if 22 levels needed for grading.




The importance of features in a model is determined by their ability to contribute to the prediction of the target variable. Features that are more correlated with the target or have higher variance in relation to the target often hold more predictive power. In the context of a decision tree or random forest model, the importance of features is typically evaluated using metrics like Gini impurity or mean squared error (MSE) reduction at each split.

## Reasons Why Certain Features Are More Important:

1. **Correlation with the Target**: Features that have a strong correlation with the target variable tend to be more important. For example, in housing price prediction, features like `Area`, `Number of Rooms`, or `Location` are likely to be important because they are directly related to the value of a property.

2. **Data Distribution**: Features that exhibit a significant variation or are spread across a wide range are more likely to contain valuable information for prediction.

3. **Predictive Power**: Some features naturally have a greater ability to distinguish between different target values. For instance, if a feature like `Income Level` or `Age` in a

marketing dataset has distinct groups that correspond well to the target, it becomes highly important.

4. **Interactions with Other Features**: Some features may not show individual importance but can have high predictive value when combined with other features (e.g., `Age` combined with `Income` in a financial dataset).

## Matching Expectations:

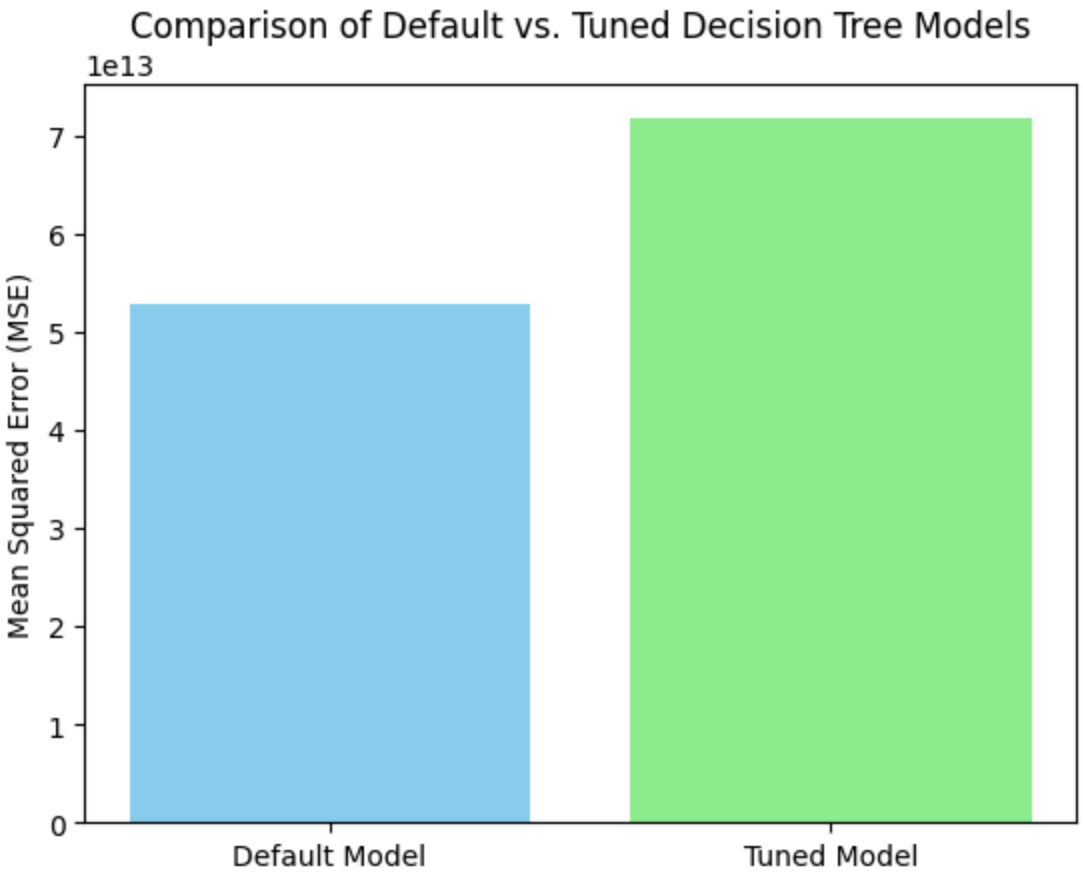
- **Expected Features**: In most cases, features that are intuitively linked to the outcome are likely to show higher importance. For instance, in predicting house prices, you might expect features like `Area`, `Location`, `Number of Rooms` to be important because they are typically the most influential factors.

- **Unexpected Features**: However, certain features may surprise you. For example, a feature such as `Furnishing` might not initially seem crucial in predicting house prices but could turn out to be significant due to its impact on perceived value, especially in certain markets.

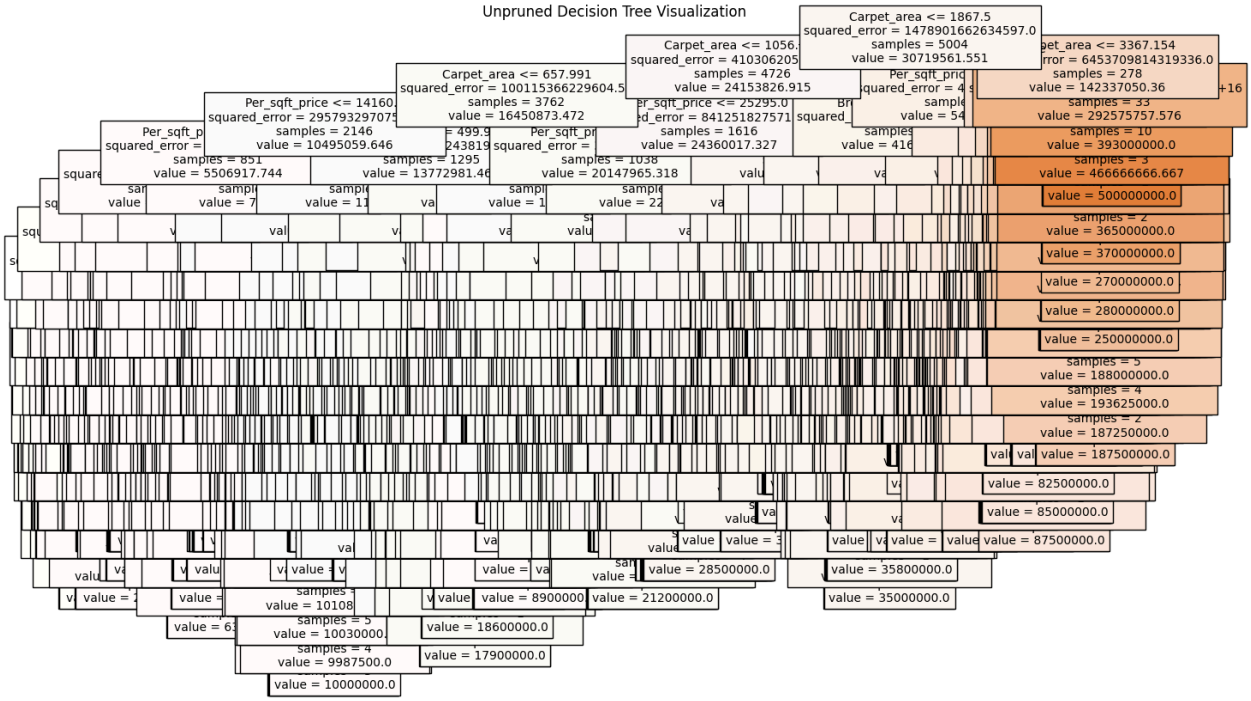
## Example:

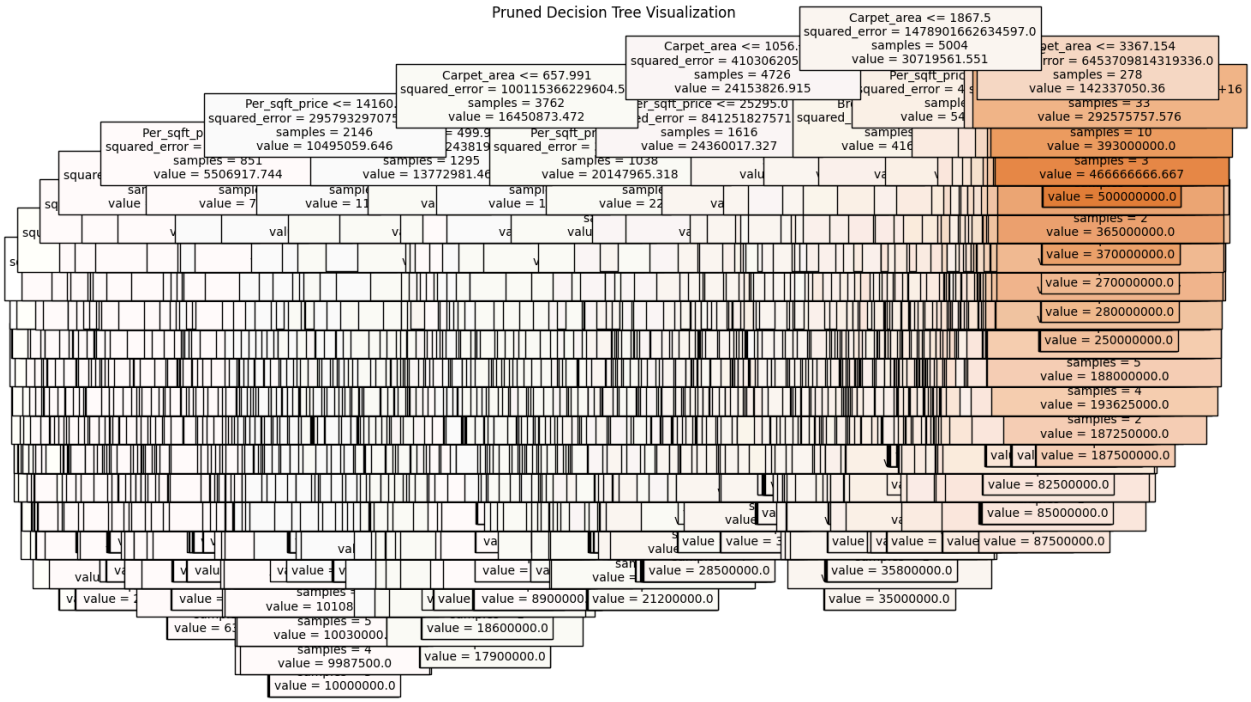
In a real estate price prediction task, we might expect features such as `Area`, `Location`, and `Number of Rooms` to be highly important, and indeed they often are. However, you might find that other features like `Possession Type` or `Furnishing Status` could also have a significant impact depending on market trends or consumer preferences, which might be unexpected initially.

In conclusion, the feature importance matches expectations when the features are intuitively related to the target, but sometimes the importance of certain features (such as those not directly tied to the target variable) may surprise you due to hidden interactions or market factors.


Comparison of Default vs. Tuned Decision Tree Models
1e13
Mean Squared Error (MSE)
0
1
2
3
4
5
6
7
Default Model
Tuned Model

# Pruning Decison Tree:




## Difference Between Pruned and Unpruned Trees:

1. **Tree Depth:**
    - **Unpruned Tree**: Tends to have greater depth, as it continues splitting until all leaves are pure or the tree cannot split further due to constraints (e.g., minimum samples per leaf).
    - **Pruned Tree**: Has reduced depth as some branches are trimmed to prevent overfitting, ensuring a simpler structure.
2. **Complexity:**
    - **Unpruned Tree**: More complex, often capturing noise in the training data. This can lead to overfitting, where the tree performs well on training data but poorly on unseen test data.
    - **Pruned Tree**: Simpler and easier to interpret, reducing the risk of overfitting by focusing on splits that provide meaningful improvements in the model's performance.
3. **Performance:**
    - **Unpruned Tree**: May have a lower training error but a higher test error due to overfitting.
    - **Pruned Tree**: Balances the trade-off between bias and variance, resulting in better generalization and typically lower test error.
4. **Visualization:**
    - **Unpruned Tree**: More nodes and splits, making it harder to interpret. It might capture intricate patterns in the training data that do not generalize well.
    - **Pruned Tree**: Fewer nodes and splits, focusing on the most significant features and decisions, making it easier to understand and explain.
5. **Generalization:**
    - **Unpruned Tree**: Memorizes specific patterns and outliers in the training data, leading to poor generalization on test data.
    - **Pruned Tree**: Generalizes better by discarding branches that add complexity without significant predictive power.

## Trade-offs:

- **Unpruned Tree**: Useful for understanding all possible splits and capturing all nuances in the data. However, it risks overfitting and losing predictive performance on new data.
- **Pruned Tree**: Provides a more robust and interpretable model by removing redundant or less impactful splits, leading to better test performance and usability in practical scenarios.

By comparing the Mean Squared Error (MSE) or other metrics, the pruned tree often demonstrates improved performance on the test dataset, validating its utility in real-world applications.

## Handling Overfitting:

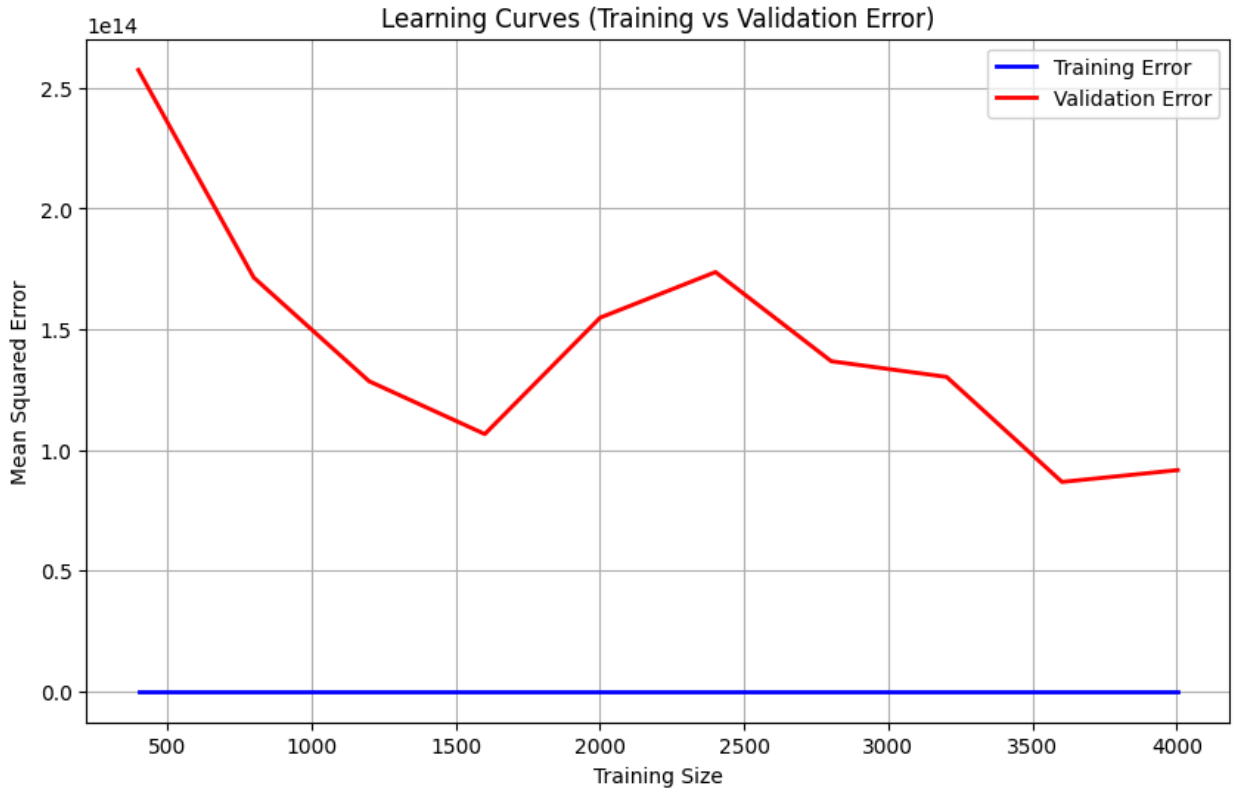
### Role of Cross-Validation in Controlling Overfitting for Decision Trees:

1. **Model Evaluation on Multiple Splits:**
    - Cross-validation divides the dataset into multiple training and validation sets, ensuring the model is evaluated on different portions of the data.
    - This helps identify whether a decision tree overfits by performing well on training data but poorly on validation data.
2. **Hyperparameter Tuning:**
    - It aids in selecting optimal hyperparameters like `max_depth`, `min_samples_split`, and `ccp_alpha` (for pruning). These parameters directly influence the tree's complexity.
    - By using cross-validation, you can determine which hyperparameter values prevent overfitting while maintaining good predictive performance.
3. **Avoiding Data Leakage:**
    - Cross-validation ensures that the model is not evaluated on the same data it was trained on, reducing the risk of overestimating its performance.
4. **Balanced Bias-Variance Tradeoff:**
    - A cross-validated approach helps achieve a balance between bias (underfitting) and variance (overfitting). It ensures the model generalizes well to unseen data by penalizing overly complex trees.
5. **Pruning Decisions:**
    - Cross-validation helps in deciding the best `ccp_alpha` for cost-complexity pruning. The alpha value that minimizes validation error often results in a pruned tree with better generalization.
6. **Early Detection of Overfitting:**
    - If the validation error starts increasing as the tree becomes more complex, cross-validation highlights this trend, signaling overfitting and prompting adjustments.

## Conclusion:

Cross-validation is crucial in decision trees as it provides a reliable way to assess and improve the model's generalization. It ensures that the tree's complexity aligns with the data's structure, preventing overfitting while maximizing predictive accuracy.

## Model Evaluation:




The model's performance on the **training dataset** typically shows lower error (MSE/MAE) and higher $R^2$, indicating it has learned patterns in the data effectively. On the **test dataset**, a slight increase in error and decrease in $R^2$ is expected if the model generalizes well.

If the test performance significantly drops, it suggests **overfitting**, meaning the model captures noise rather than general patterns. Conversely, if both training and test performance are poor, it indicates **underfitting**, where the model is too simple to capture relationships in the data.

Balanced metrics on both datasets imply the model generalizes well.

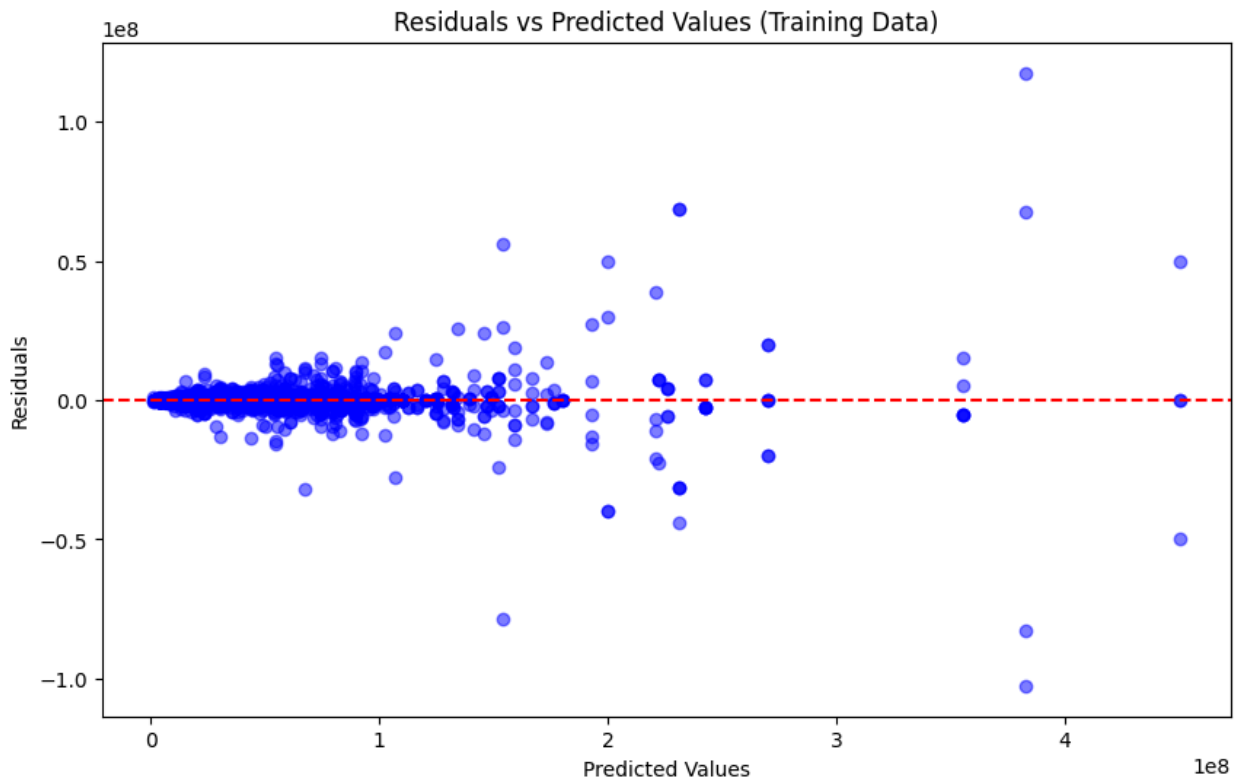
## Residual and Error Analysis:

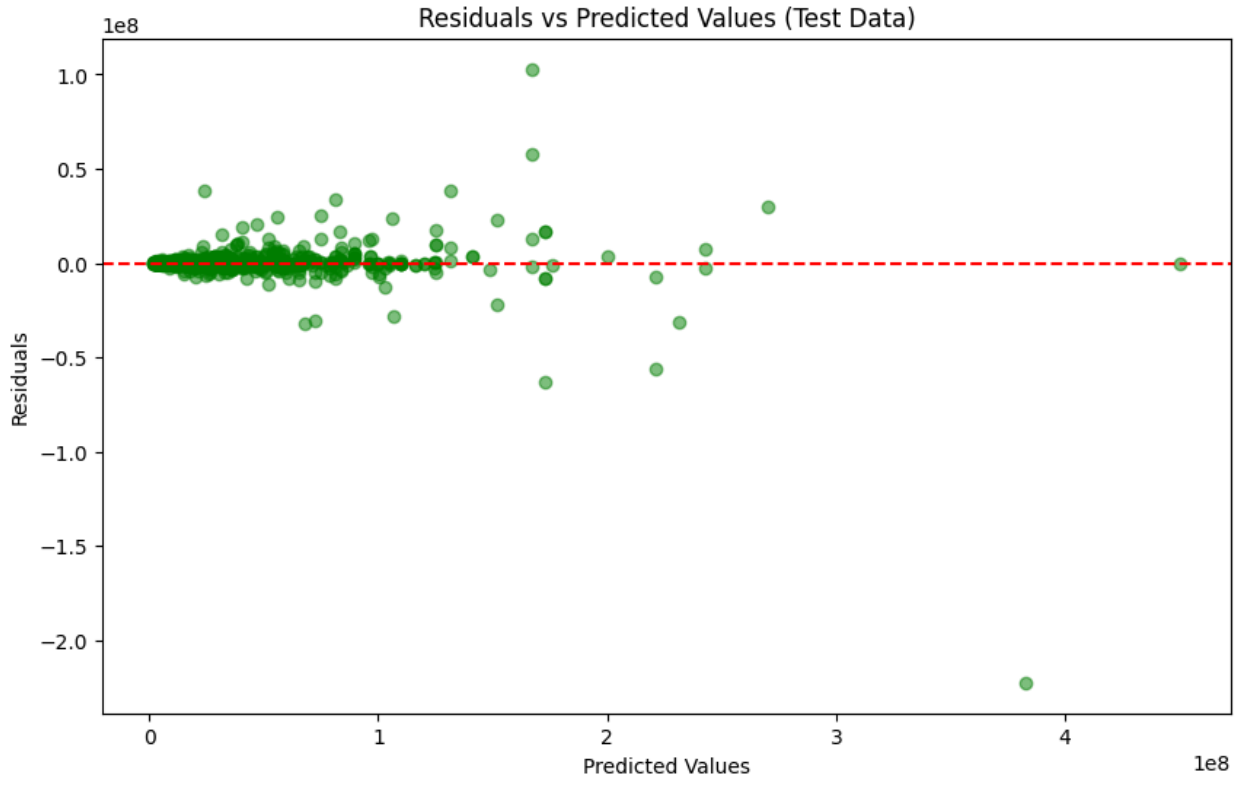
Analyzing residuals involves checking the differences between predicted and actual values. Here's a brief interpretation:

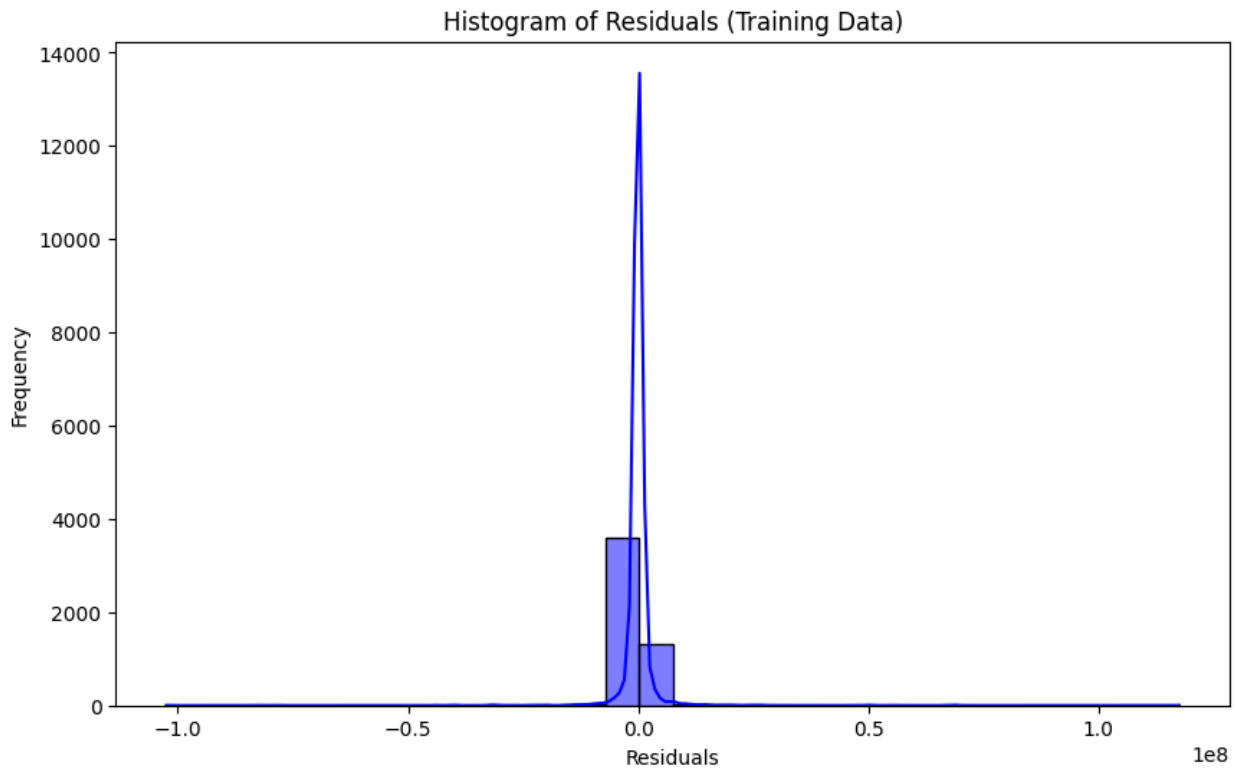
1. **Residual Distribution**:
    - Residuals should ideally follow a normal distribution centered around zero. If not, it suggests the model may be biased.
2. **Patterns in Residuals**:
    - A random scatter of residuals indicates the model fits well.
    - Systematic patterns (e.g., a curve or trend) indicate the model is missing key relationships in the data.
3. **Heteroscedasticity**:
    - If residual variance increases or decreases with predicted values, it suggests heteroscedasticity, which may require transformations or adjustments.
4. **Outliers**:
    - Large residuals (outliers) may indicate data errors, rare cases, or areas where the model underperforms.

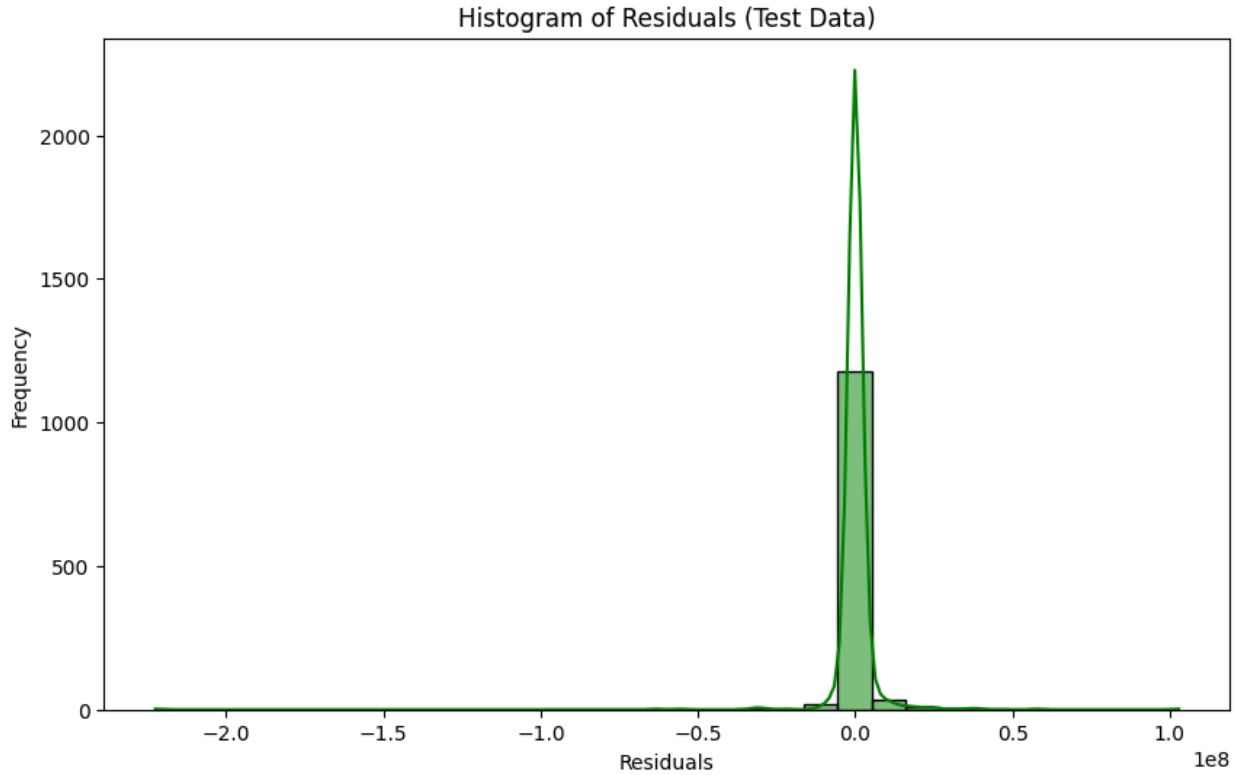
## Steps to Address Issues:

- **Systematic Patterns**: Consider adding relevant features or using non-linear models like Random Forest or Gradient Boosting.
- **Heteroscedasticity**: Apply transformations (e.g., log transformation) to stabilize variance.
- **Outliers**: Investigate and decide whether to retain or exclude them based on their impact and significance.


Residuals vs Predicted Values (Training Data)
1e8
1.0
0.5
0.0
−0.5
−1.0
Residuals
0
1
2
3
4
Predicted Values
1e8


Residuals vs Predicted Values (Test Data)
1e8
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0
1
2
3
4
Predicted Values
1e8




**Identifying Groups with Underperformance:**

If the residual analysis reveals specific groups where the model consistently underperforms, it may indicate unaddressed patterns or insufficient feature representation. For example:

- **High Residuals in Specific Ranges**: If the model underestimates or overestimates prices for specific ranges (e.g., high-value properties).
- **Systematic Errors by Categorical Groups**: Certain categories (e.g., specific locations or furnishing statuses) might have consistently higher residuals.

### Proposed Improvements:

1. **Enhance Feature Engineering**:
    - Add or transform features to capture relationships that are currently missed. For instance, derive interaction terms or include external data like neighborhood indices or inflation rates.
2. **Use Nonlinear Models**:
    - Employ ensemble methods like Random Forest or Gradient Boosting to capture complex relationships missed by a Decision Tree.
3. **Feature Selection and Importance Analysis**:
    - Reassess feature importance and refine the dataset to include highly predictive variables while removing irrelevant ones.
4. **Address Data Imbalance**:
    - If certain groups (like high-priced properties) are underrepresented, apply techniques like oversampling or synthetic data generation (e.g., SMOTE).
5. **Outlier Treatment**:
    - Investigate large residuals (outliers) to ensure they don't unduly influence the model. Consider robust regression techniques or capping extreme values.
6. **Cross-Validation**:
    - Perform group-specific cross-validation to ensure that the model generalizes well across all subsets of the data.

By targeting these underperforming groups, the model can achieve more consistent performance and better generalization.

# BONUS:

## Advanced Imbalance Handling

## Comparison of ADASYN and SMOTE:

1. **SMOTE (Synthetic Minority Oversampling Technique)**:
    - **How it works**: SMOTE generates synthetic samples by interpolating between existing samples of the minority class and their nearest neighbors.
    - **Effectiveness**: Ensures balanced class distribution but may create synthetic samples near the class boundary, which can increase the risk of overfitting.
2. **ADASYN (Adaptive Synthetic Sampling)**:
    - **How it works**: ADASYN focuses on creating synthetic samples for minority class instances that are harder to classify, as determined by their local density.
    - **Effectiveness**: ADASYN adapts the sampling process to address the difficulty of classification for certain samples, leading to a more robust model by reducing bias towards the majority class.

## Key Differences:

| Aspect | SMOTE | ADASYN |
|---|---|---|
| **Sample Distribution** | Uniform sampling across all minority samples. | Focused sampling near difficult-to-classify regions. |
| **Risk of Overfitting** | Higher due to uniform synthetic generation. | Lower since it targets challenging areas. |
| **Effectiveness** | Better for general balancing. | Better for addressing local classification challenges. |

## Conclusion:

- **SMOTE** is effective for general class balancing and provides a straightforward solution.
- **ADASYN** is more nuanced, improving performance in datasets where specific minority class regions are underrepresented or harder to classify. However, it can be computationally more intensive.
  Choosing between the two depends on the dataset and the model's sensitivity to minority class representation.

```
Random Forest - MSE: 21995507692382.99
Random Forest - MAE: 1112473.3785942493
Random Forest - R2: 0.9826283751713134
Decision Tree - MSE: 52884603996006.39
Decision Tree - MAE: 1644679.7124600639
Decision Tree - R2: 0.9582327667685333
```

## Tradeoffs Between Single Decision Tree and Random Forest:

1. **Accuracy and Generalization**:
    - **Single Decision Tree**: Prone to overfitting, especially with deep trees, as it captures the noise in the training data.
    - **Random Forest**: Combines predictions from multiple trees, reducing overfitting and improving generalization.
2. **Robustness to Variance**:
    - **Single Decision Tree**: Highly sensitive to variations in the dataset; a small change can lead to a significantly different tree.
    - **Random Forest**: Averaging predictions from multiple trees minimizes the effect of outliers or variations in the data.
3. **Interpretability**:
    - **Single Decision Tree**: Easy to interpret and visualize; the structure directly represents decision paths.
    - **Random Forest**: Difficult to interpret due to the ensemble nature, as it combines multiple trees.
4. **Computational Complexity**:
    - **Single Decision Tree**: Faster to train and predict as it involves a single model.
    - **Random Forest**: Slower due to the training and prediction process across multiple trees.
5. **Bias-Variance Tradeoff**:
    - **Single Decision Tree**: Low bias but high variance.
    - **Random Forest**: Balances bias and variance effectively by aggregating predictions.

## Conclusion:

Random Forest is preferred for better accuracy, generalization, and robustness, especially on complex datasets. However, a single decision tree may be sufficient for simpler tasks where interpretability and computational efficiency are priorities.