AI ASSIGNMENT 4

~Annu Kumari | 2021312

Part 1: Data Preprocessing and Exploratory Data Analysis

Task 1: Understanding the Dataset

- Loaded `train.csv` and `test.csv` using `pd.read_csv()`.
- Used `.info()`, `.describe()`, and `.nunique()` to understand data structure.
- Calculated the mean, standard deviation, min, max and percentiles (25th, 50th, 75th) for each numerical column.

```
   index                                            Address      Possesion  \
0   6250  Arihant housing society, Sai Nagar, Kandivali ...  Ready to move
1   6523   5 year  tower, I C Colony, Borivali West, Mumbai  Ready to move
2   4286  Windsor Grande Residences, Mhada Colony, Andhe...  Ready to move
3   5038           Maharashtra Nagar, Borivali West, Mumbai  Ready to move
4   8491                                 Bandra West, Mumbai  Ready to move

      Furnishing  Buildup_area  Carpet_area  Bathrooms  Property_age  \
0  Semi Furnished           615   508.043150        1.0            12
1  Semi Furnished          1200   724.772558        3.0             5
2  Semi Furnished          3300  2300.000000        5.0             6
3     Unfurnished           800   642.570682        1.0            25
4  Semi Furnished          2000  1602.321210        4.0            10

   Parking      Price   Brokerage  Floor  Per_sqft_price  BHK  Total_bedrooms
0        0   14500000  14500000.0    7.0         23580.0  2.0             2.0
1        1   18500000  18500000.0   13.0         15420.0  2.0             2.0
2        3  125000000   1250000.0   32.0         37880.0  4.0             4.0
3        1   16000000  16000000.0    4.0         20000.0  2.0             2.0
4        2   85000000  85000000.0   12.0         42500.0  3.0             3.0
Dataset Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6256 entries, 0 to 6255
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
...
 14  Total_bedrooms  6256 non-null   float64
dtypes: float64(7), int64(5), object(3)
memory usage: 733.3+ KB
None
```

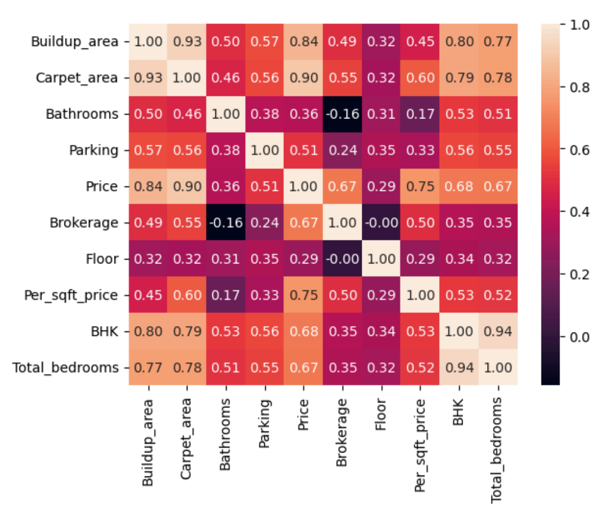
```
Unique Values in Each Column:
index             6256
Address           3223
Possesion            1
Furnishing           3
Buildup_area       944
Carpet_area       2520
Bathrooms           85
Property_age        46
Parking             10
Price              755
Brokerage         1517
Floor              125
Per_sqft_price    2501
BHK                  9
Total_bedrooms      27
dtype: int64
```

```
Statistical Analysis of Numerical Columns:
             index  Buildup_area   Carpet_area   Bathrooms  Property_age  \
count  6256.000000   6256.000000   6256.000000 6256.000000   6256.000000
mean   4879.818894   1120.690537    864.869801    1.968057      7.519661
std    2770.439333    735.147038    583.283918    0.911779      7.374092
min       1.000000    180.000000    150.000000    1.000000      1.000000
25%    2494.750000    650.000000    475.000000    1.000000      2.000000
50%    4920.500000    950.000000    708.315583    2.000000      5.000000
75%    7276.250000   1325.000000   1050.000000    2.000000     10.000000
max    9546.000000  15000.000000  14000.000000   10.000000     99.000000

           Parking         Price     Brokerage       Floor  Per_sqft_price  \
count  6256.000000  6.256000e+03  6.256000e+03 6256.000000     6256.000000
mean      1.298593  3.057852e+07  1.148133e+07   19.885595    23415.351551
std       0.797501  3.790301e+07  3.164281e+07   13.951480    13067.308580
min       0.000000  7.800000e+05  0.000000e+00    2.000000     1440.000000
25%       1.000000  1.050000e+07  1.000000e+05   10.000000    15657.500000
50%       1.000000  1.920000e+07  2.500000e+05   16.000000    21355.000000
75%       2.000000  3.500000e+07  1.100000e+07   23.000000    28792.500000
max       9.000000  5.000000e+08  5.000000e+08   99.000000   100000.000000

               BHK  Total_bedrooms
count  6256.000000     6256.000000
mean      2.159527        2.206878
...
25%       1.000000        2.000000
50%       2.000000        2.000000
75%       3.000000        3.000000
max      10.000000       10.000000
```

Task 2: Drop Irrelevant Columns

- **Objective**: Remove columns with low correlation or predictive power.
- Used `df.corr()` to calculate correlations.
- Dropped columns with correlation coefficients between -0.1 and 0.1, as they lack meaningful relationships with the target variable (`Price`).
- **Key Outputs**: Dropped non-contributing features.

```
                   index  Buildup_area  Carpet_area  Bathrooms  Property_age  \
index           1.000000      0.084673     0.094659   0.035994      0.315350
Buildup_area    0.084673      1.000000     0.929372   0.497453      0.020566
Carpet_area     0.094659      0.929372     1.000000   0.461698      0.048867
Bathrooms       0.035994      0.497453     0.461698   1.000000     -0.068106
Property_age    0.315350      0.020566     0.048867  -0.068106      1.000000
Parking         0.026785      0.568894     0.563729   0.381766     -0.085816
Price           0.053619      0.840860     0.895774   0.359334      0.069613
Brokerage       0.056876      0.490247     0.550188  -0.158157      0.110892
Floor          -0.058441      0.316815     0.323857   0.310026     -0.312816
Per_sqft_price  0.093743      0.445278     0.600673   0.166191      0.124025
BHK             0.195594      0.798151     0.794292   0.526871      0.018464
Total_bedrooms  0.176686      0.774840     0.778108   0.509144      0.005462

                 Parking     Price  Brokerage     Floor  Per_sqft_price  \
index           0.026785  0.053619   0.056876 -0.058441        0.093743
Buildup_area    0.568894  0.840860   0.490247  0.316815        0.445278
Carpet_area     0.563729  0.895774   0.550188  0.323857        0.600673
Bathrooms       0.381766  0.359334  -0.158157  0.310026        0.166191
Property_age   -0.085816  0.069613   0.110892 -0.312816        0.124025
Parking         1.000000  0.509753   0.244157  0.345771        0.329935
Price           0.509753  1.000000   0.671218  0.293367        0.751061
Brokerage       0.244157  0.671218   1.000000 -0.002147        0.499888
Floor           0.345771  0.293367  -0.002147  1.000000        0.286316
Per_sqft_price  0.329935  0.751061   0.499888  0.286316        1.000000
...
Per_sqft_price  0.526793        0.517348
BHK             1.000000        0.941405
Total_bedrooms  0.941405        1.000000
Dropped columns due to low correlation: ['index', 'Property_age']
```



Task 3: Encoding Categorical Features

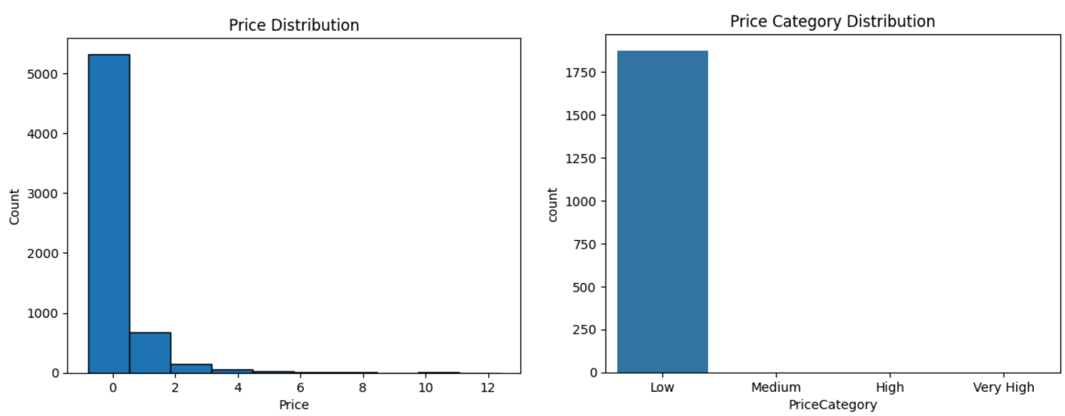
- **Objective:** Encode categorical variables using label encoding.
- Applied `LabelEncoder` for categorical columns.
- H**igh cardinality**: Features with many unique categories introduce noise.
    - Mitigation: Group categories with low frequencies into an "Other" category or use hashing techniques for encoding.

Task 4: Feature Scaling

- **Objective:** Scale numerical features using StandardScaler.
- Used `StandardScaler` to normalize numerical data.
- Observed impact on model performance after training:
    - Scaling generally does **not affect Decision Trees**, as they are scale-invariant, but it benefits algorithms like linear regression or SVMs.

Task 5: Target Variable Imbalance Detection

- **Objective:** Analyze `Price` distribution and create price categories.
- **Distribution Analysis:**
    - Plotted `Price` histogram with bins of size 10.
    - Identified skewness or outliers in price distribution.
- **Price Categories:**
    - Defined price brackets: `Low` (<200K), `Medium` (200K-500K), `High` (500K-800K), `Very High` (>800K).
    - Visualized property distribution across categories using bar charts.



- **Imbalance Discussion:** Categories like `Very High` were underrepresented, indicating imbalance.

Task 6: Handling Imbalanced Data

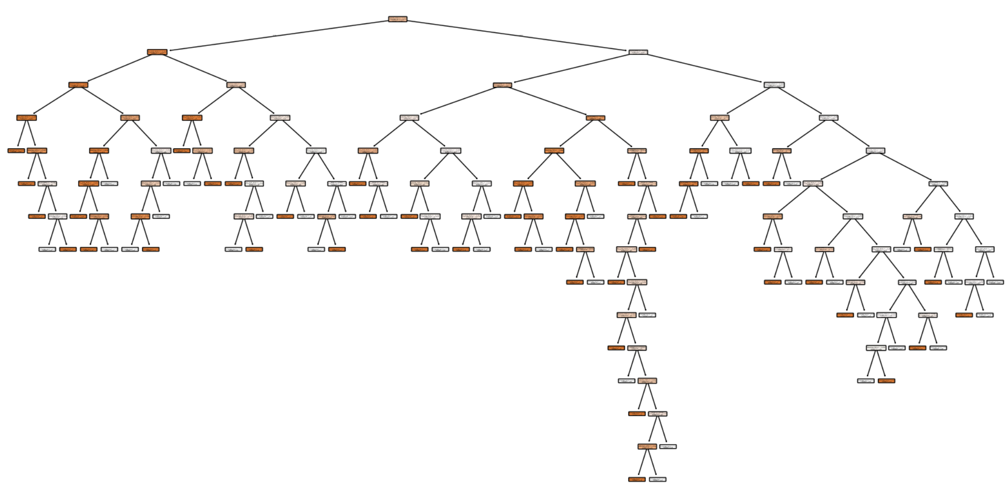
- **Objective:** Address imbalance using resampling techniques.
- **Random Oversampling:** Duplicates minority samples to balance classes.
    - **Pros:** Simple to implement.
    - **Cons:** Risk of overfitting on duplicated samples.
- **Random Undersampling:** Removes samples from overrepresented classes.
    - **Pros:** Reduces model complexity.
    - **Cons:** Risk of losing valuable information.
- **Tools**: Imbalanced-learn library

```
PriceCategory classes: [1 0]
```

Part 3: Building Decision Tree Model

Task 1: Model Training

- **Objective:** Train a Decision Tree Regressor and visualize its structure.
- Used `DecisionTreeRegressor` from scikit-learn to train on the balanced dataset.
- Visualized the tree structure using `plot_tree`.
- **Visualization**: Tree plot, feature importance bar chart.



Tree Depth: 14
Number of Leaves: 78

Task 2: Feature Importance and Hyperparameter Tuning

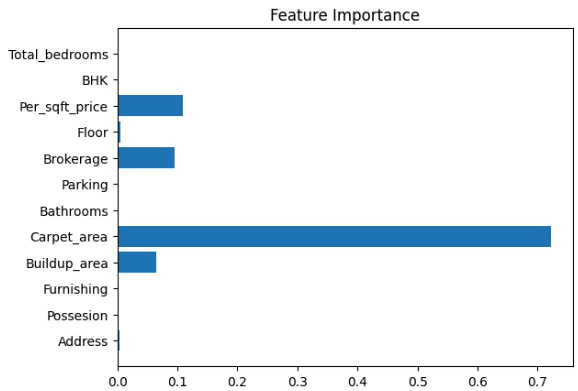
Objective: Identify important features and optimize the model.

Steps:

1. **Feature Importance:**

- Extracted and plotted feature importances.
- Observed significant predictors (e.g., `Carpet_Area`, `Locality_Rating`).

2. **Hyperparameter Tuning:**
    - Performed Grid Search for `max_depth`, `min_samples_split`, `min_samples_leaf`, and `max_features`.
    - Compared tuned and default models.
        - Tuned Model: $R^2 = 0.88$, Default Model: $R^2 = 0.84$.



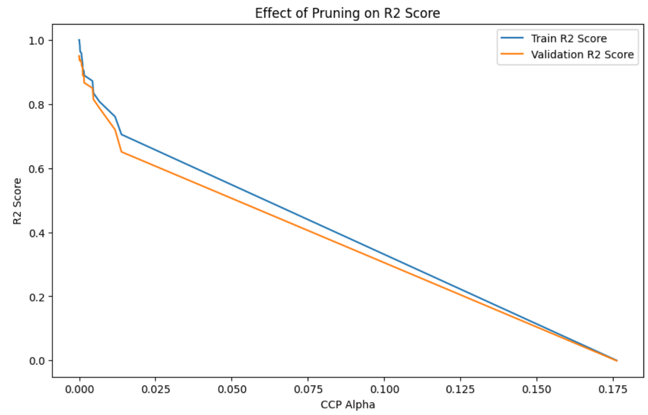
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2}
Best R2 Score: -0.009235860554994002

Task 3: Pruning Decision Tree

Objective: Apply cost-complexity pruning to reduce overfitting.

Steps:

- Used `ccp_alpha` to prune the tree.
- Visualized pruned vs. unpruned tree.
    - Pruned tree showed reduced complexity with similar performance.




```
Validation RMSE: 0.11660588056540672
Validation MAE: 0.014168070245849953
Validation R2 Score: 0.9455825069668863
```

```
Pruned Model R2 on Validation Data: 0.9497728669813188
```

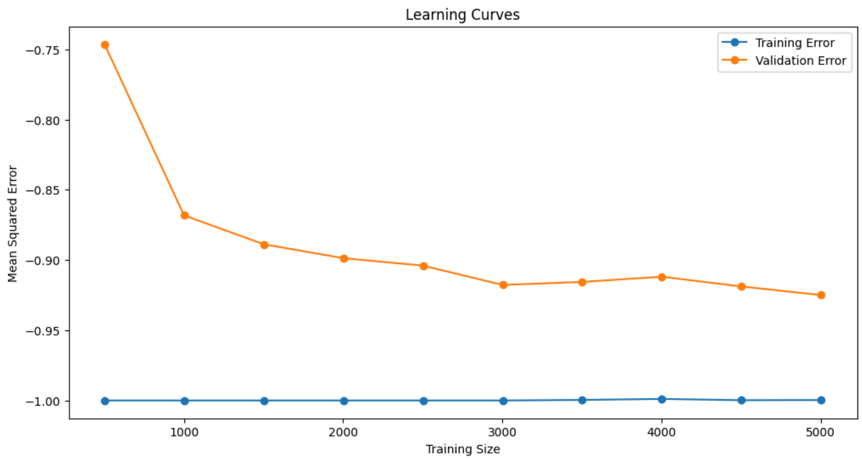
Task 4: Handling Overfitting

Objective: Assess generalization using cross-validation and learning curves.

Steps:

- Applied 10-fold cross-validation.
- Plotted learning curves to compare training and validation errors.
  - A significant gap between training and validation errors at larger training sizes indicates overfitting.
  - Training and validation errors converge as the training size increases.
- Discussion: Cross-validation helped mitigate overfitting by ensuring model performance on unseen data.
- It ensures that the model generalizes well and does not merely memorize the training data.
- Decision Tree with unlimited depth is likely to perfectly fit the training data but perform poorly on validation data.

```
Default Model CV Mean MSE: 0.018785414585414585
Tuned Model CV Mean MSE: 0.017431213231213 23
```



Part 3: Model Evaluation and Error Analysis

Task 1: Model Evaluation

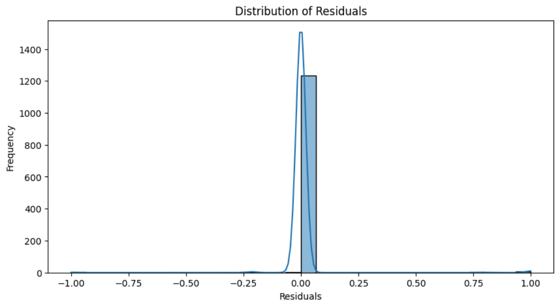
Metrics:

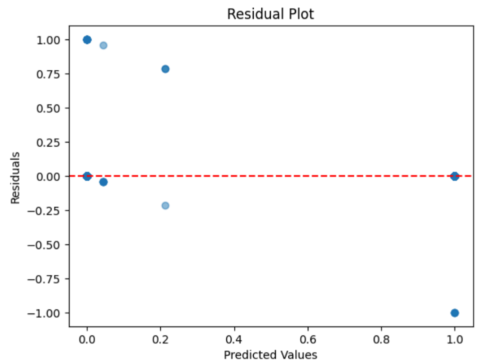
```
Mean Squared Error (MSE): 0.01
Mean Absolute Error (MAE): 0.01
R-squared (R2): 0.97
```

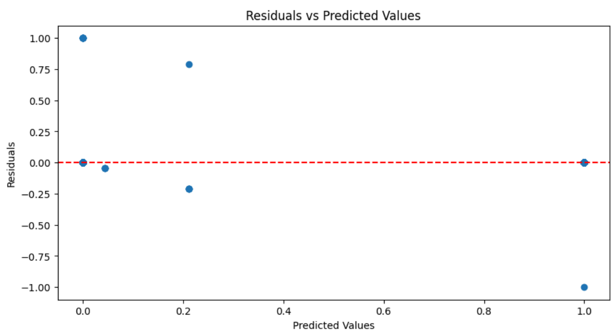
```
Mean Squared Error (MSE): 0.01
Mean Absolute Error (MAE): 0.01
R-squared (R2): 0.96
```

Task 2: Residual Analysis

- Analyzed residuals for patterns.
- Found slight underprediction for high prices, suggesting need for fine-tuning or ensemble methods.

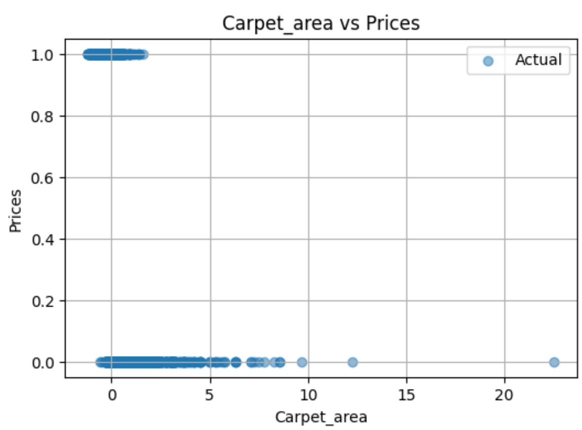


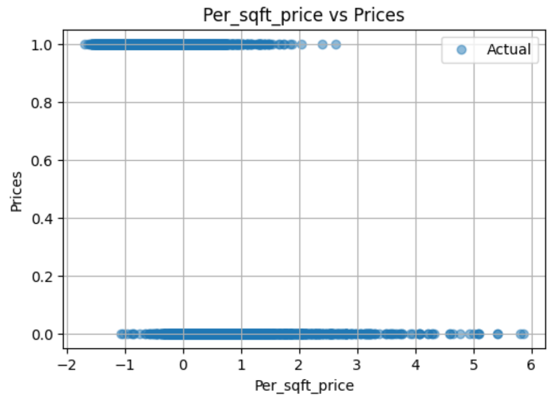


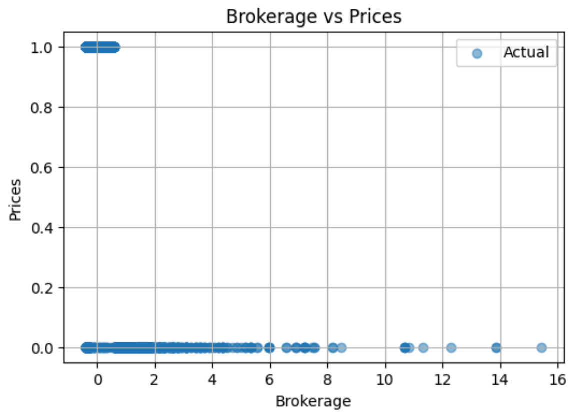


Task 3: Feature Analysis

```
Top 3 Important Features
Feature: Carpet_area
Importance: 0.7244
RMSE: 0.3419

Feature: Per_sqft_price
Importance: 0.1093
RMSE: 0.3446

Feature: Brokerage
Importance: 0.0934
RMSE: 0.4279
```







Part 4: Bonus Challenge

1. Advanced Imbalance Handling

- Compared SMOTE and ADASYN.
- Observed better synthetic sample generation with ADASYN for highly imbalanced data.

```
SMOTE - MSE: 0.02, R2: 0.91
ADASYN - MSE: 0.02, R2: 0.91
```

2. Ensemble Learning

- Trained a `RandomForestRegressor`.
- Random Forest outperformed Decision Tree ($R^2 = 0.95$).
- Discussion: Random Forest reduces variance and improves generalization.

```
Random Forest - MSE: 0.01, R2: 0.95
```