CSE 422 Lab Project Report

Topic: Flat Price Prediction

Team members

Md. Hossain Anas	22299217
Rifah Tasnim Labonno	22201040

Group No. - 11

Section - 19

Table of Contents

1. Introduction..3

2. Dataset description..3

3. Dataset preprocessing..6

4. Feature scaling...7

5. Dataset splitting..8

6. Model training and testing..8

7. Model selection/Comparison analysis..........................8

8. Conclusion...16

1. Introduction

This project aims to analyze a dataset of flat properties to predict their Price_Category (e.g., Low, Medium, High) based on various features such as location, size, number of bedrooms/bathrooms, building age, and amenities.

The project addresses the challenge of classifying flat prices into distinct categories. This can be valuable for: 1) Buyers/Renters: To understand if a flat's listed price category aligns with its features and market trends. 2) Sellers/Agents: To strategically price properties and understand key drivers of price perception. 3)Investors: To identify potentially undervalued or overvalued property segments.

The motivation behind this project is to leverage machine learning techniques to extract meaningful insights from real estate data. By building predictive models, we can understand the complex interplay of factors determining a flat's price category, ultimately aiding in more informed decision-making in the property market. The increasing availability of property data makes such analytical approaches both feasible and highly relevant.

2. Dataset Description

Dataset Description:

The dataset comprises 1200 data points and 12 features, which include an equal number of 6 numerical and 6 categorical features. The numerical features are : ‘Size_sqft', 'Num_Bedrooms', 'Num_Bathrooms', 'Floor_Number', 'Building_Age_Years', 'Distance_to_CityCenter_km' and the categorical features are : 'Location', 'Has_Balcony', 'Parking_Available', 'Nearby_Schools', 'Security_Level', 'Price_Category' .

Classification or Regression Problem:

This is a multi-class classification problem since the target variable Price_Category consists of discrete ordinal categories (Low, Medium, High). The primary goal is to predict the correct price category based on the input features, which aligns with the objectives of a classification model.

Number of Data Points:

The dataset contains 1200 rows and 12 columns. Each row represents a unique observation of the Price_Category of a flat.

Types of Features:

- Quantitative/Numerical Features:

    Size_sqft: Continuous (area in square feet).

    Num_Bedrooms: Discrete (count of bedrooms).

    Num_Bathrooms: Discrete (count of bathrooms).

    Floor_Number: Discrete (floor level of the property).

    Building_Age_Years: Discrete (age of the building in years).

    Distance_to_CityCenter_km: Continuous (distance in kilometers).

- Categorical Features:

    Nominal:

        Location: Countryside/City Center/Suburbs (no inherent order).

        Has_Balcony: Binary (Yes/No).

        Parking_Available: Binary (Yes/No).

        Nearby_Schools: Ordinal (Few/Many, but could be treated as nominal).

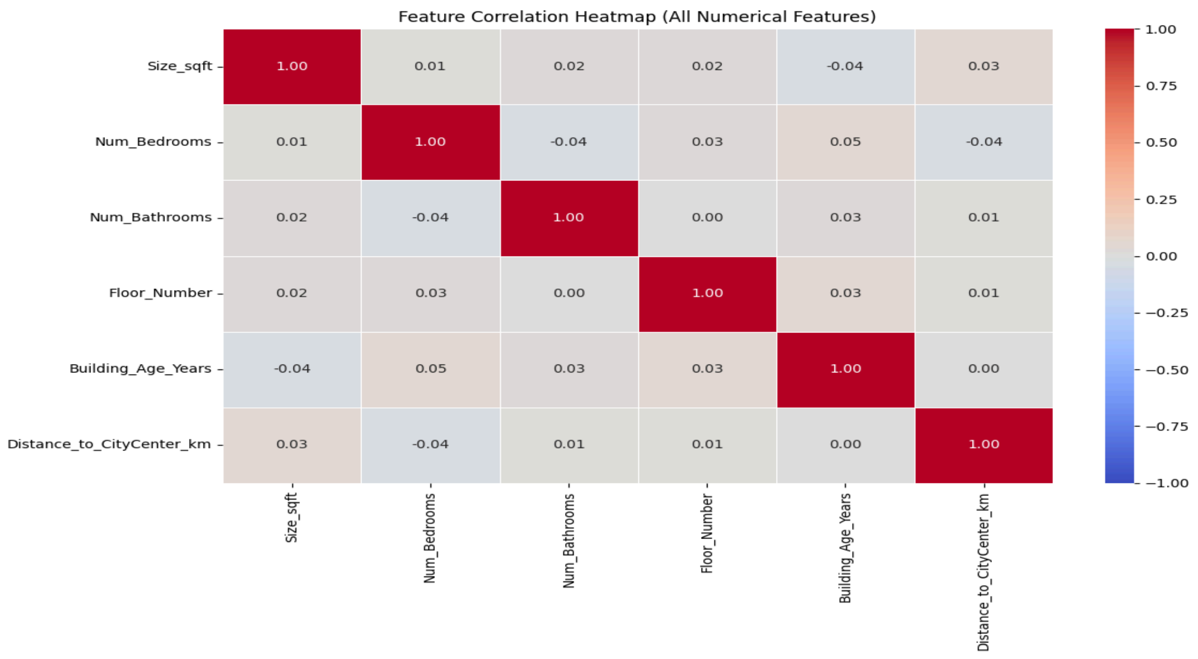
    Ordinal:

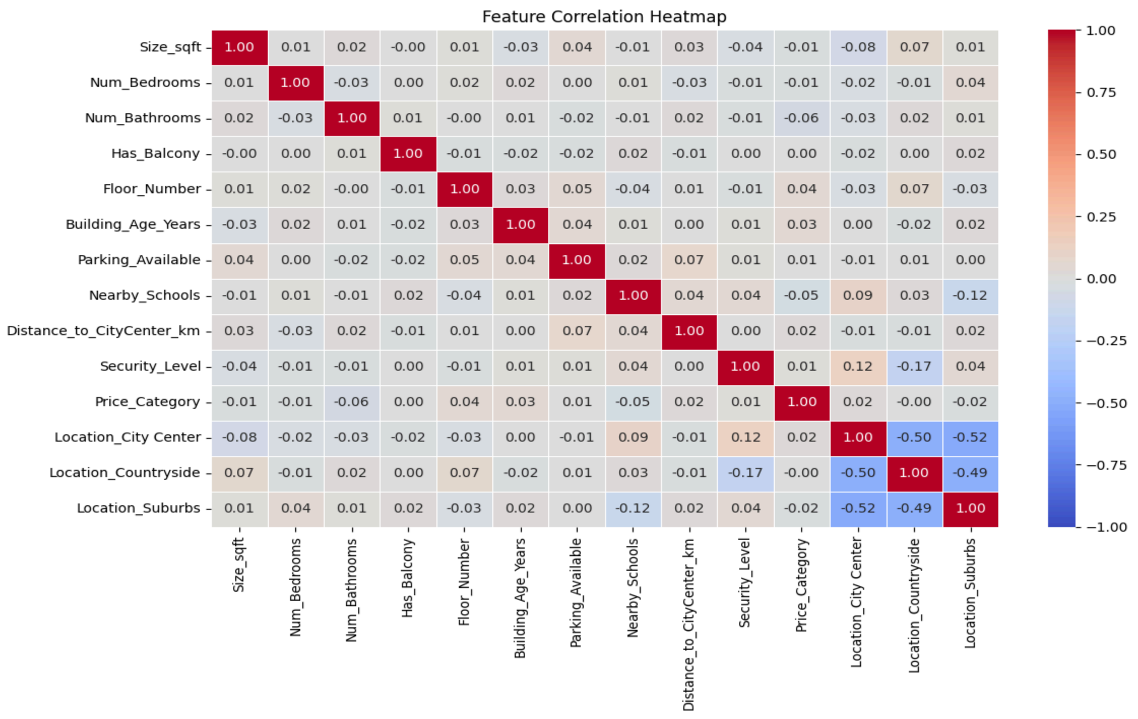
        Security_Level: Ordered categories (Low/Medium/High).

        Price_Category: Ordinal categorical (Low < Medium < High).

Heatmap:

A heatmap was generated to visualize the correlation of all features.

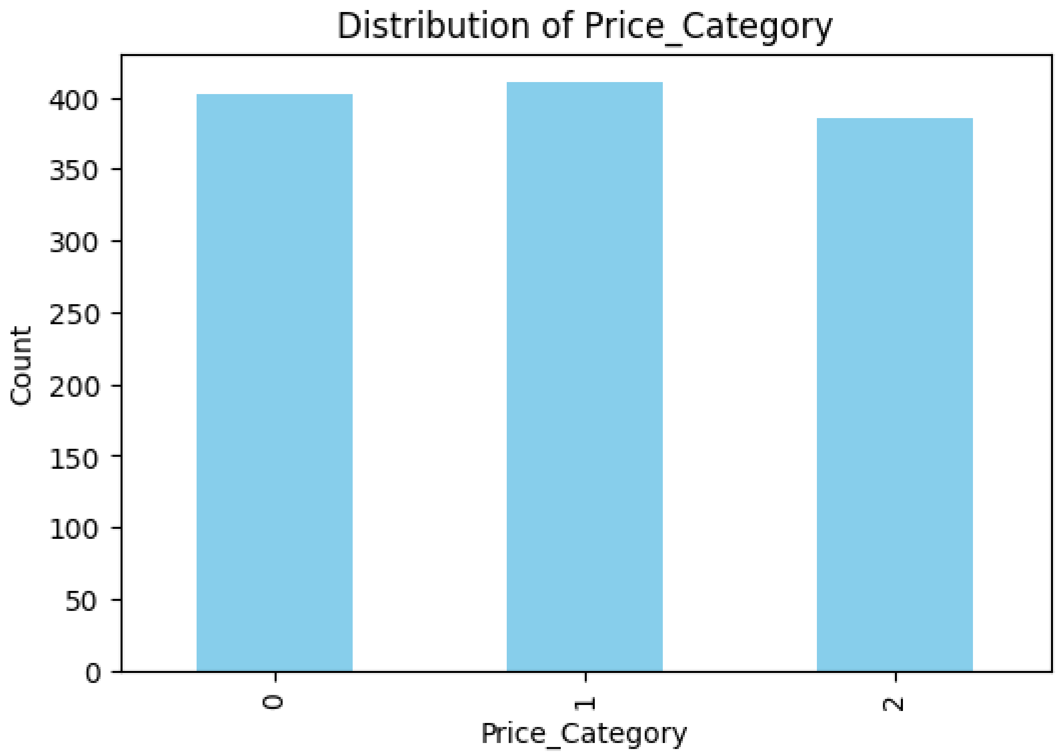




The heatmap shows a very poor correlation between the features.

Imbalanced Dataset:

The dataset is somewhat balanced for the target variable 'Price_Category but has very poor correlation between the features.



3. Dataset Preprocessing

Faults:

- Many Missing Values(NaN) in a very small dataset

- Nominal features (Location, Has_Balcony, etc.) need one-hot encoding.

- Ordinal features (Security_Level, Price_Category) need label/ordinal encoding.

Solutions:

Since our dataset is relatively small and doesn't contain any obviously irrelevant columns, we had to handle preprocessing thoroughly. We couldn't afford to drop rows or columns arbitrarily, as each data point was valuable. Also in most of the cases the distribution of the data was balanced.

1. Handling Missing Values:

Before replacing any NULL values, we thoroughly explored the data by visualizing each column. This helped us understand the relationships between variables, identify skewness, spot outliers and assess overall data distribution. For categorical features, we analyzed their distribution and, in many cases, used proportion-based imputation-splitting the missing values based on existing values frequencies (such as 60/40, 30/70 etc). For numerical features, instead of blindly using mean, median or mode, we relied on visual analysis and distribution patterns to impute values in a way that preserved the balance and integrity of the dataset. For some columns we even checked if we can predict the missing values by running a model.

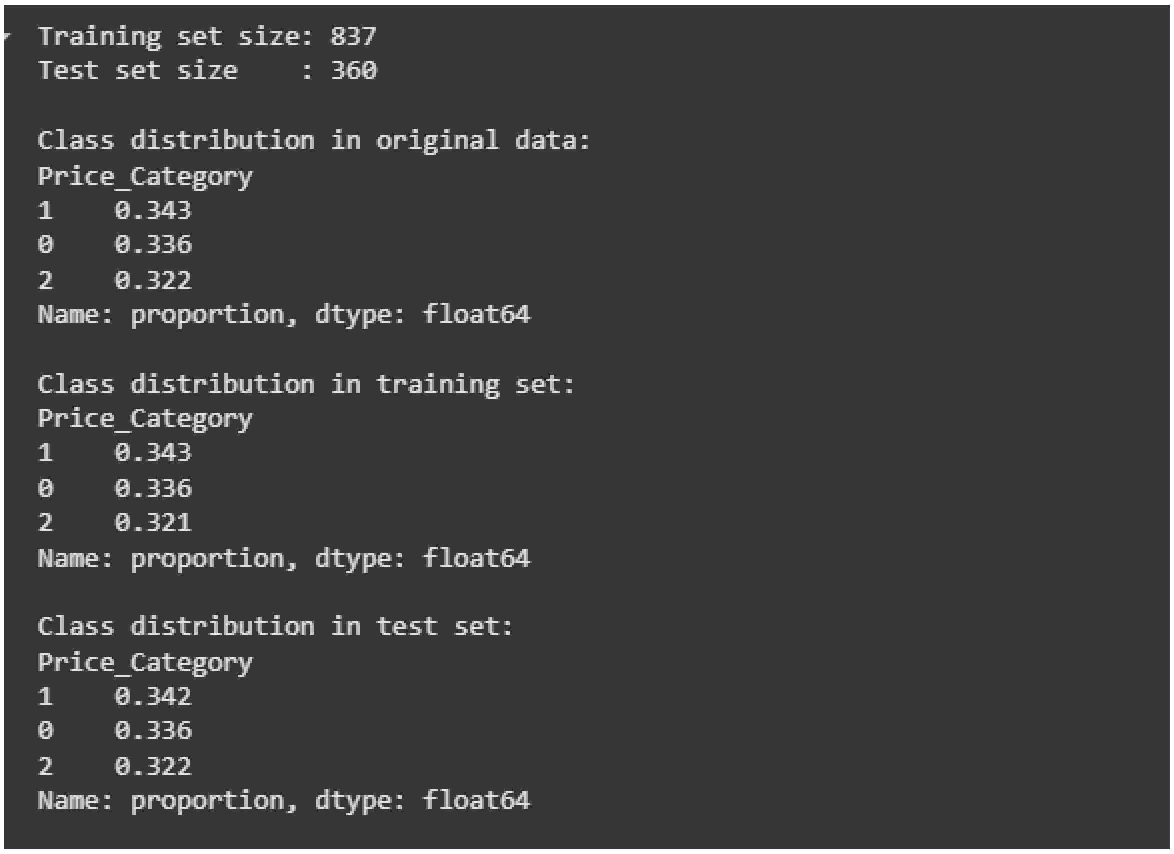
2. Encoding:

To prepare the dataset for modeling, several categorical features had to be encoded into numerical format. For binary columns containing values like 'yes'/'no' or 'high/low' we made it 0/1 . This allowed us to retain the binary nature of the data in a format suitable for running ML models. For ordinal features such as price_category we assigned 0,1,2 in-place of low, mid and high.

4. Feature Scaling:
In our feature scaling approach, we applied mixed scaling using ColumnTransformer to handle different feature types appropriately. Specifically, we used StandardScaler for continuous numerical features such as Size_sqft, Num_Bedrooms, and Distance_to_CityCenter_km, ensuring they have zero mean and unit variance. For binary and ordinal features like Has_Balcony, Parking_Available, Nearby_Schools, and Security_Level, we applied MinMaxScaler to scale them into the [0, 1] range, preserving their non-negativity and relative order. All other columns, such as one-hot encoded location variables, were left unchanged using the remainder='passthrough' setting. This method ensures each type of feature is scaled in a way that aligns with its meaning and use in downstream models.

5. Data Splitting

We split the dataset in a 70:30 ratio as instructed. Seventy percent of the data was used for training, and 30% for testing. The data split was stratified to ensure that the class distribution in our training and testing datasets is similar to the original.



```
Training set size: 837
Test set size    : 360

Class distribution in original data:
Price_Category
1    0.343
0    0.336
2    0.322
Name: proportion, dtype: float64

Class distribution in training set:
Price_Category
1    0.343
0    0.336
2    0.321
Name: proportion, dtype: float64

Class distribution in test set:
Price_Category
1    0.342
0    0.336
2    0.322
Name: proportion, dtype: float64
```

6. Model Training and Testing

We trained five models (including Neural Network) using this dataset. The models used are:

1. Neural Network

2. KNN

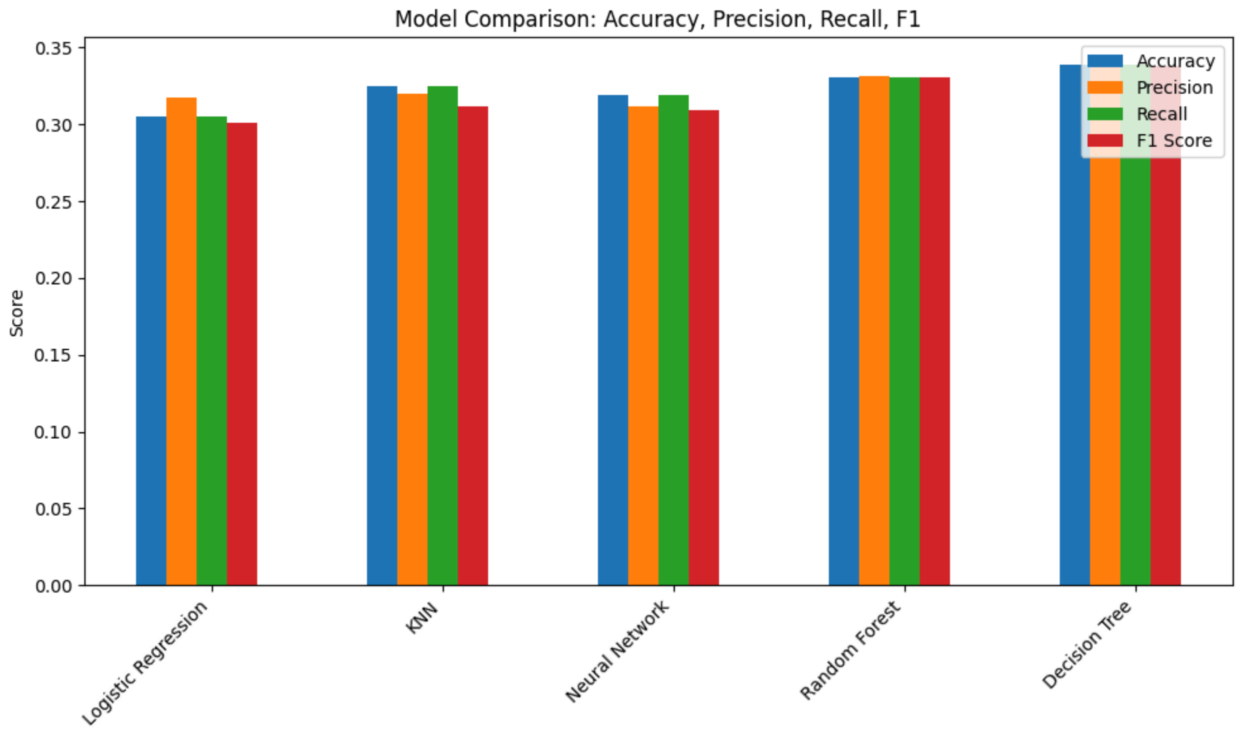
3. Logistic Regression

4.Random Forest

5. Decision Tree

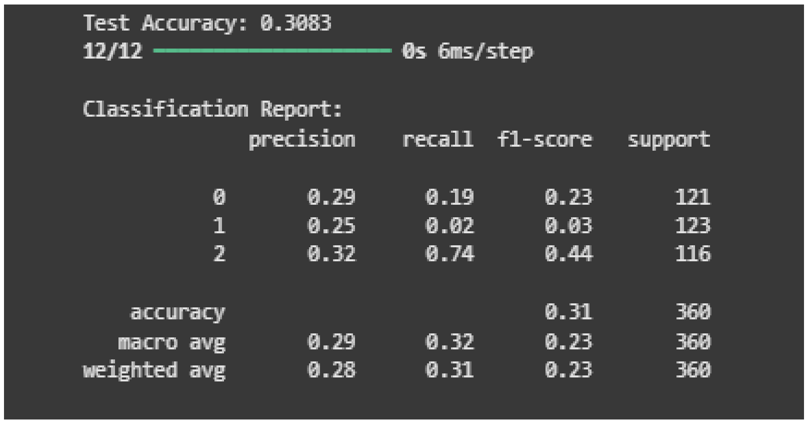
7. Model selection/Comparison analysis

1. **Bar chart showcasing the prediction accuracy of all models:**



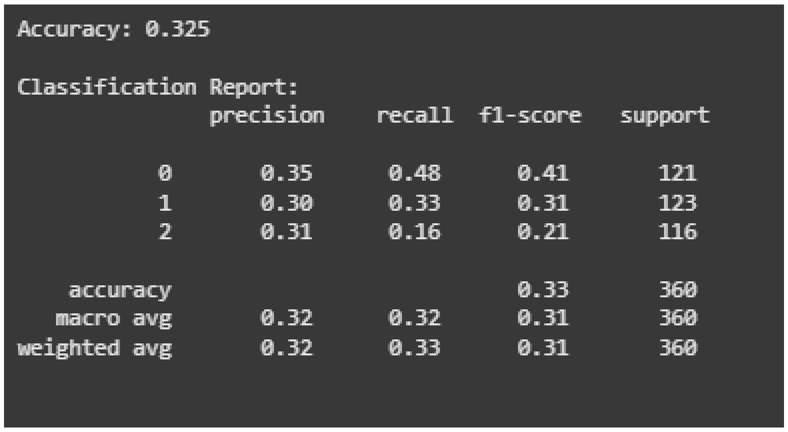
2. Precision, recall comparison of each model:

- **Neural Network :**



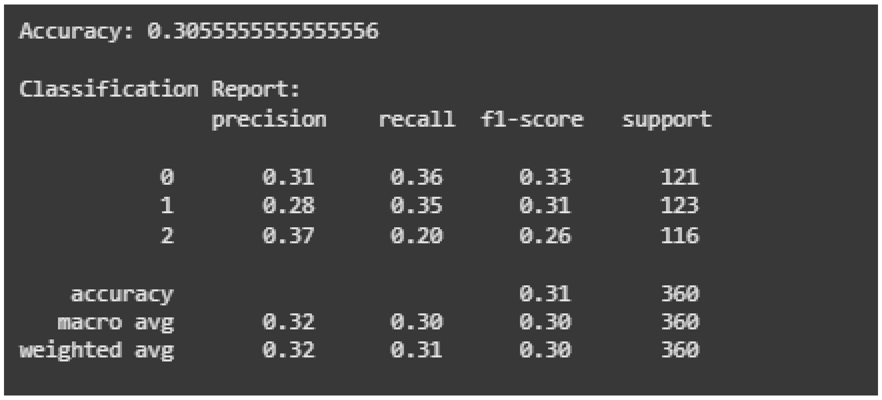
```
Test Accuracy: 0.3083
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step

Classification Report:
              precision    recall  f1-score   support

           0       0.29      0.19      0.23       121
           1       0.25      0.02      0.03       123
           2       0.32      0.74      0.44       116

    accuracy                           0.31       360
   macro avg       0.29      0.32      0.23       360
weighted avg       0.28      0.31      0.23       360
```

- **KNN :**



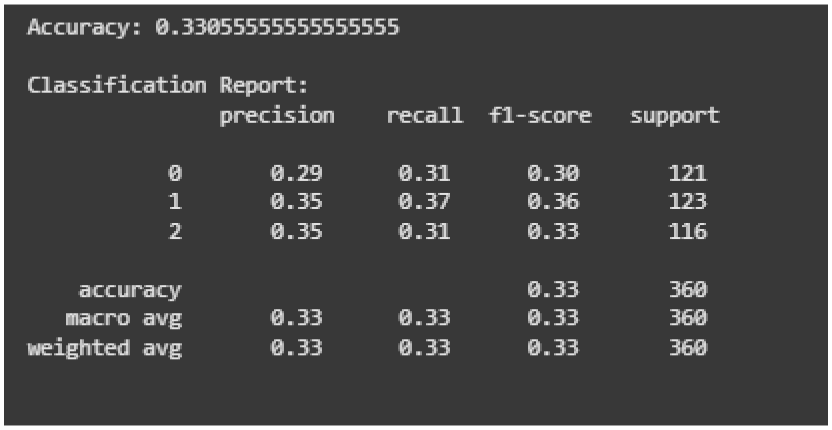
```
Accuracy: 0.325

Classification Report:
              precision    recall  f1-score   support

           0       0.35      0.48      0.41       121
           1       0.30      0.33      0.31       123
           2       0.31      0.16      0.21       116

    accuracy                           0.33       360
   macro avg       0.32      0.32      0.31       360
weighted avg       0.32      0.33      0.31       360
```

- Logistic Regression:



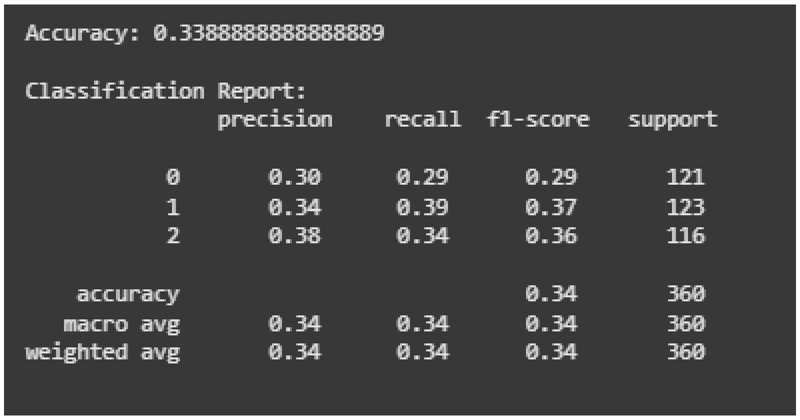
```
Accuracy: 0.30555555555555556

Classification Report:
              precision    recall  f1-score   support

           0       0.31      0.36      0.33       121
           1       0.28      0.35      0.31       123
           2       0.37      0.20      0.26       116

    accuracy                           0.31       360
   macro avg       0.32      0.30      0.30       360
weighted avg       0.32      0.31      0.30       360
```

- Random Forest:



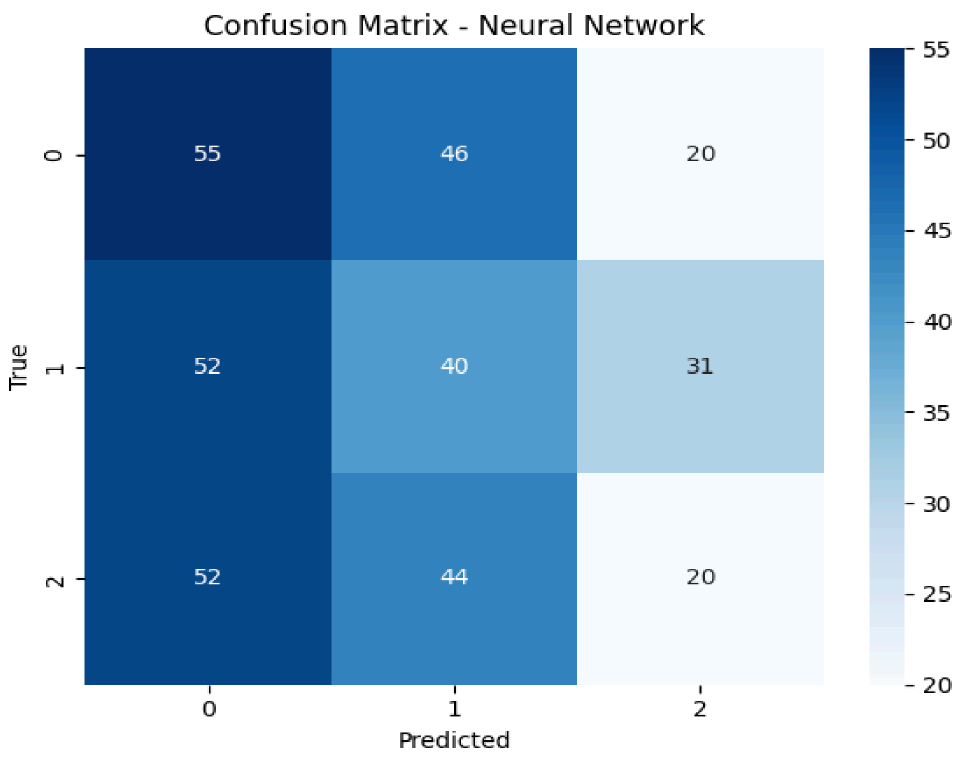
```
Accuracy: 0.33055555555555555

Classification Report:
              precision    recall  f1-score   support

           0       0.29      0.31      0.30       121
           1       0.35      0.37      0.36       123
           2       0.35      0.31      0.33       116

    accuracy                           0.33       360
   macro avg       0.33      0.33      0.33       360
weighted avg       0.33      0.33      0.33       360
```

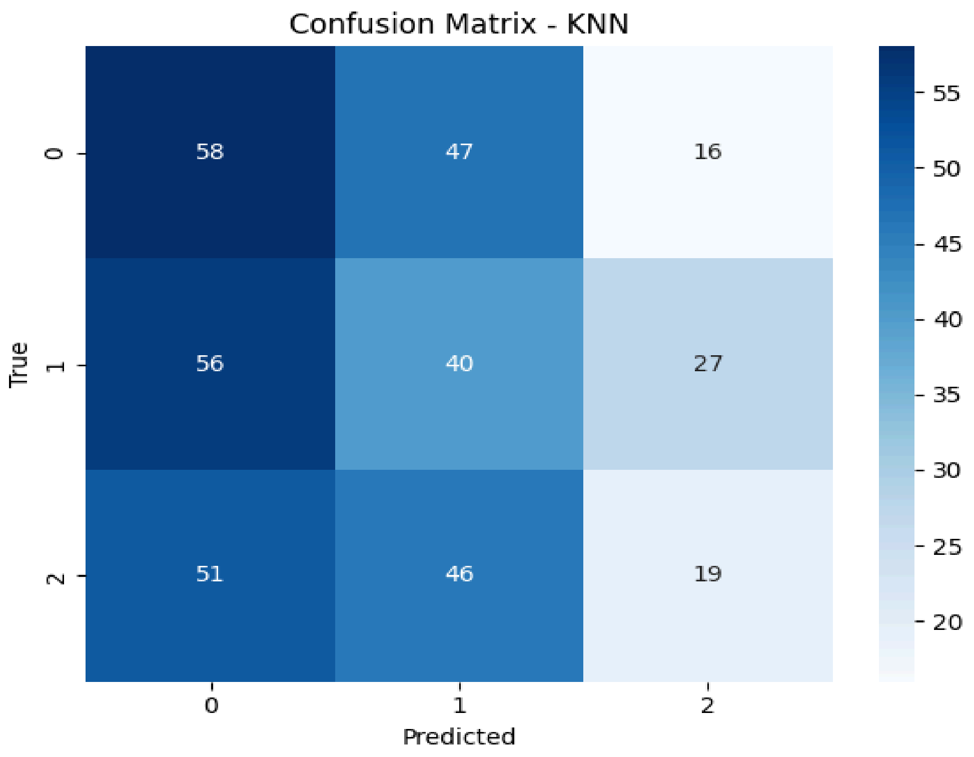
- Decision Tree:

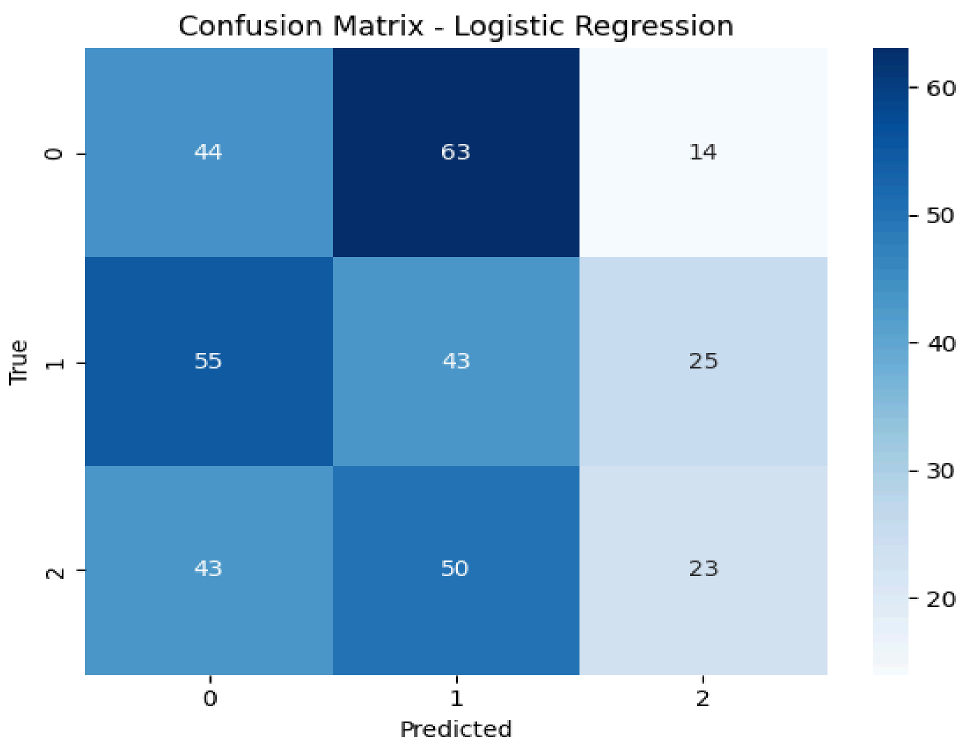


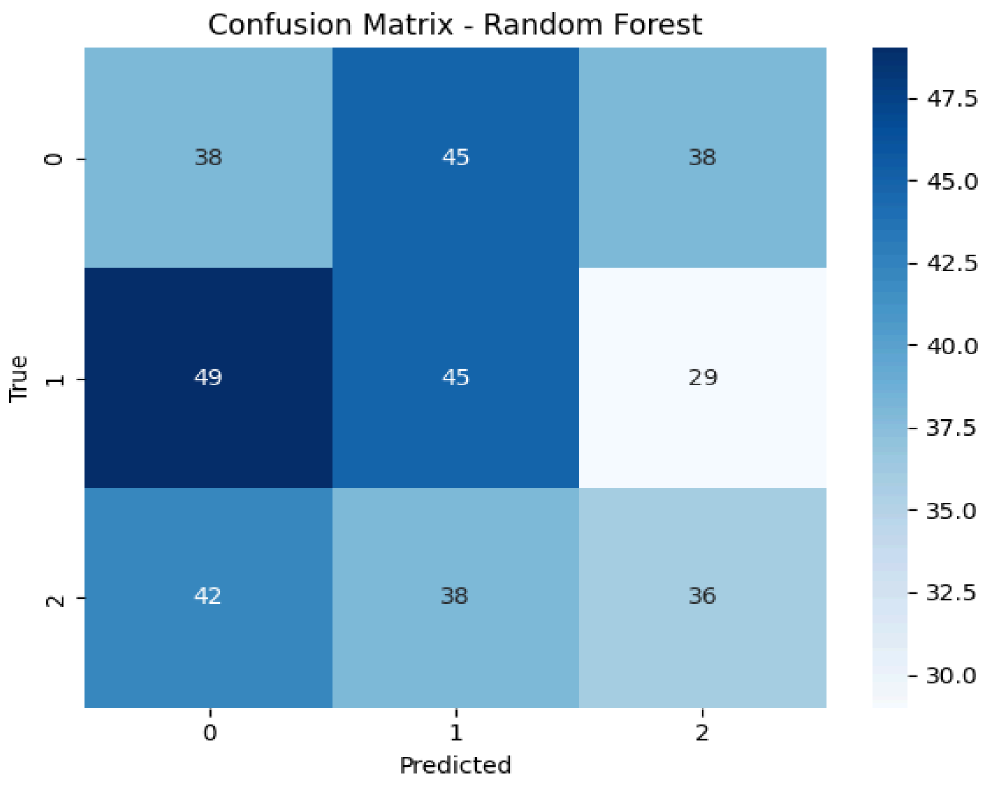
```
Accuracy: 0.3388888888888889

Classification Report:
              precision    recall  f1-score   support

           0       0.30      0.29      0.29       121
           1       0.34      0.39      0.37       123
           2       0.38      0.34      0.36       116

    accuracy                           0.34       360
   macro avg       0.34      0.34      0.34       360
weighted avg       0.34      0.34      0.34       360
```

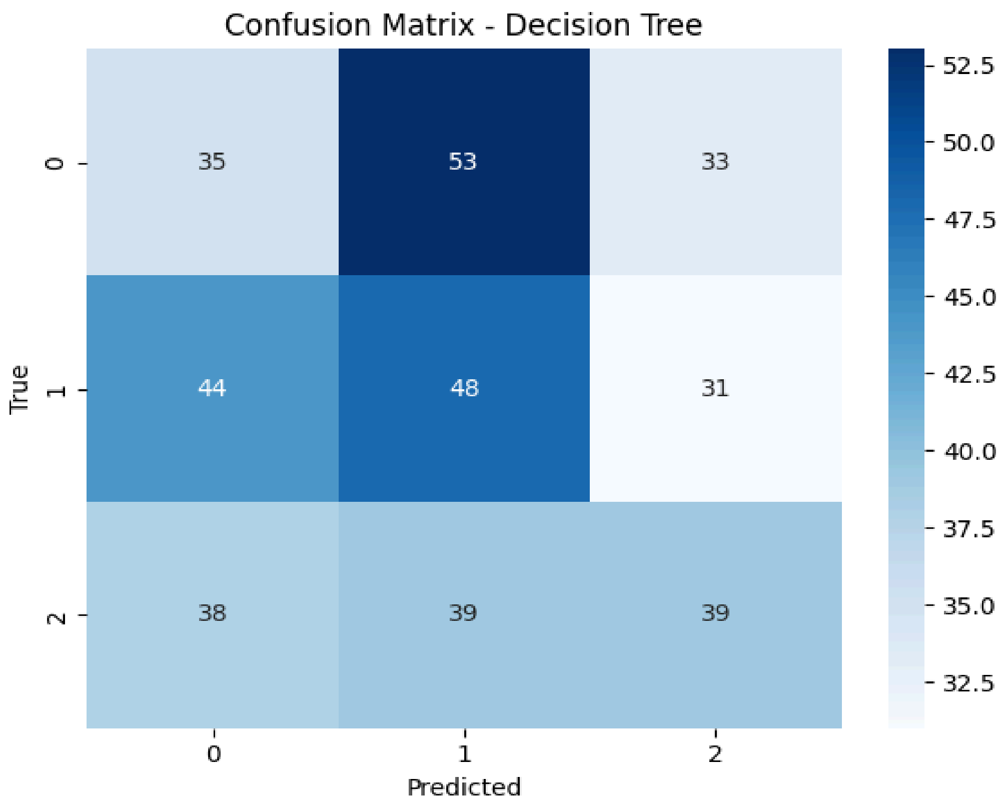
3. Confusion Matrix :



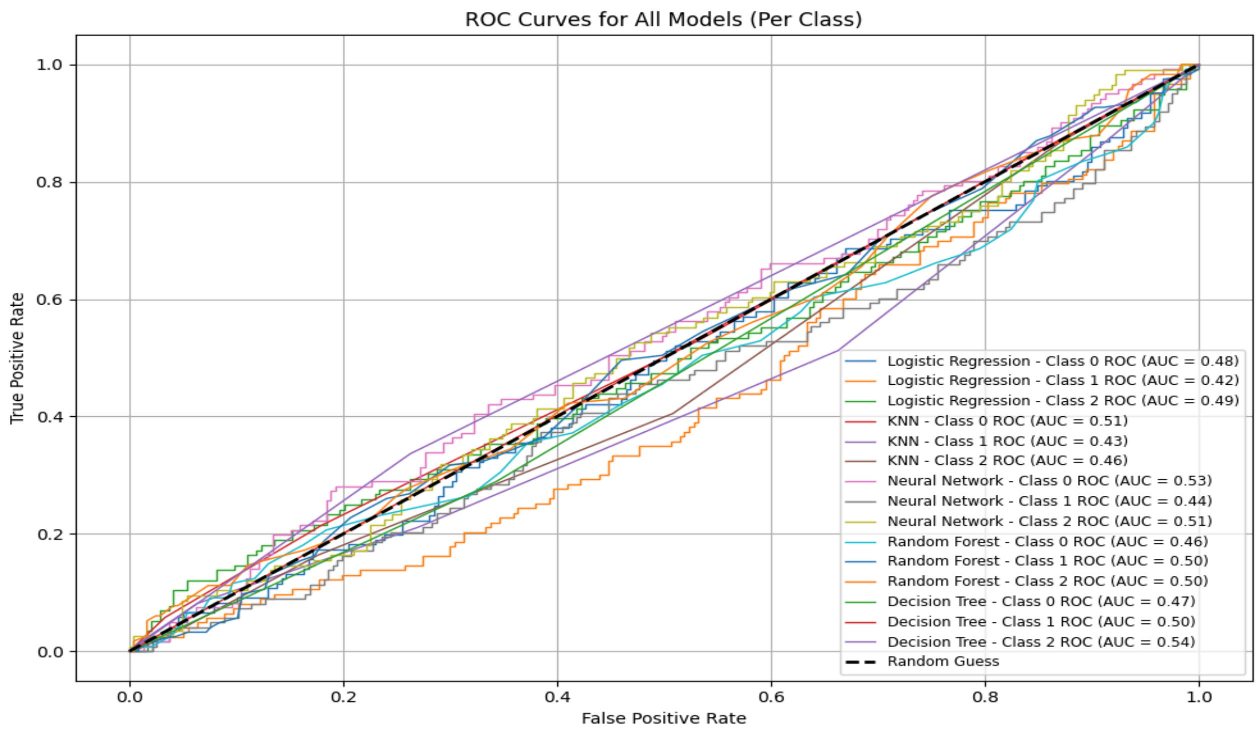


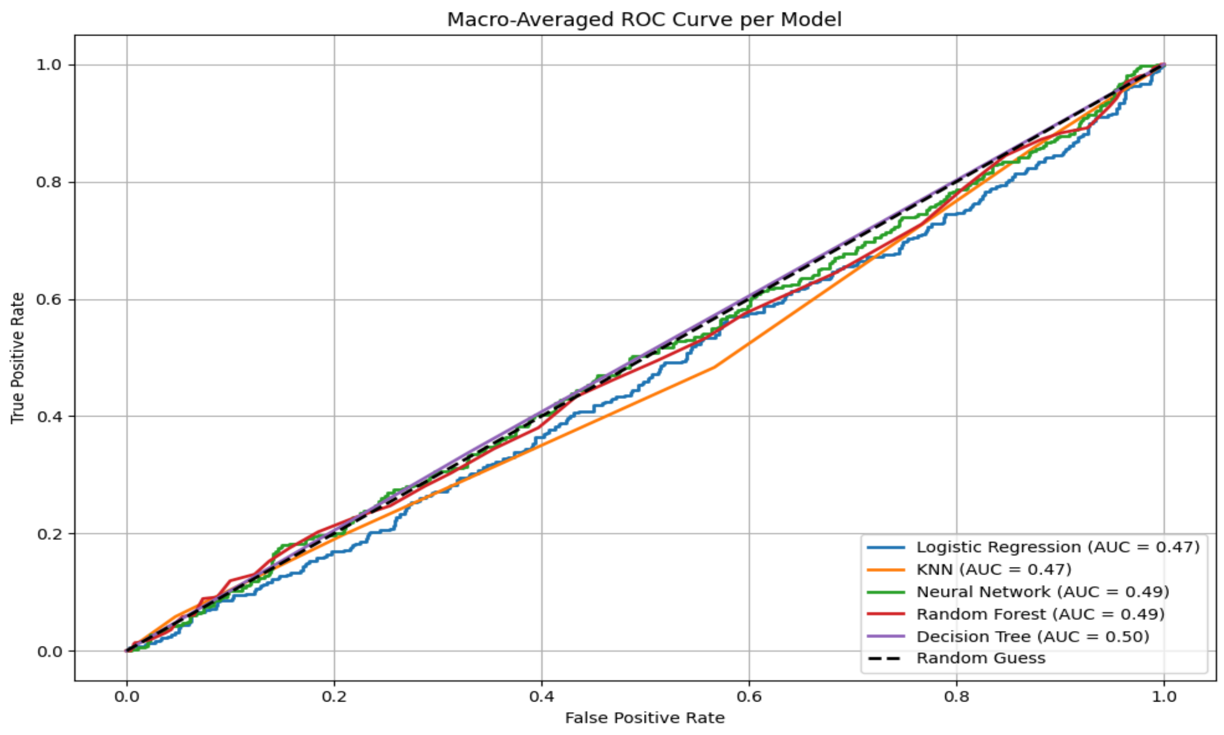

Confusion Matrix - Logistic Regression
True
Predicted
0
1
2
0
1
2
44
63
14
55
43
25
43
50
23
60
50
40
30
20


Confusion Matrix - Random Forest
True
Predicted
0
1
2
0
1
2
38
45
38
49
45
29
42
38
36
47.5
45.0
42.5
40.0
37.5
35.0
32.5
30.0


Confusion Matrix - Decision Tree
True
Predicted
0
1
2
35
53
33
44
48
31
38
39
39
52.5
50.0
47.5
45.0
42.5
40.0
37.5
35.0
32.5

4. AUC score, ROC curve :





8. Conclusion

The results of this analysis reveal that all five tested models—Neural Network, K-Nearest Neighbors, Logistic Regression, Random Forest, and Decision Tree—performed poorly in predicting flat price categories. Accuracy, precision, recall, and F1-scores hovered around 0.30–0.34, barely above random guessing for a three-class problem, while ROC curves and AUC scores showed weak discriminative power. Confusion matrices highlighted significant misclassification, especially for the 'Medium' category. The models' poor performance likely stems from low feature-target correlation, insufficient predictive signals in the dataset, and potential noise introduced by extensive imputation. Challenges faced included handling missing values, selecting appropriate imputation and encoding strategies, and implementing a mixed scaling approach. Despite thorough preprocessing and diverse algorithms, none of the models could uncover meaningful patterns, suggesting either fundamental data limitations or that key predictive features are missing. Consequently, these models currently lack practical value for real-world flat price classification.