Inspiring Excellence

BRAC University

Department of Computer Science and Engineering

Artificial Intelligence

Course Project Report

Student Information:

Name: Abdullah Al Mazid Zomader
ID: 24241189
Section: 20

Name: Avisheek Pal Joy
ID: 23101115
Section: 20

Course Information:

Course Code: CSE-422
Course Title: Artificial Intelligence

Date: 05 January, 2026

Contents

Chapter 1

Introduction

Housing prices are influenced by multiple factors such as location, size, number of rooms, and available amenities. Accurately estimating property value is important for buyers, sellers, and real estate analysts, as it helps in making informed financial decisions. However, manually analyzing these factors can be time-consuming and subjective, often leading to inconsistent pricing judgments.

This project aims to address this issue by developing a machine learning–based classification system that predicts the price category of a flat (such as low, medium, or high) based on its key attributes. Instead of predicting an exact price, the model classifies properties into meaningful categories, making the results easier to interpret and more practical for decision-making.

The motivation behind this project is to explore how data-driven approaches can improve transparency and efficiency in real estate pricing. By leveraging historical housing data and applying classification algorithms, the system seeks to uncover patterns between property features and price ranges. This can assist potential buyers in budgeting, help sellers set competitive prices, and provide a foundation for further predictive analytics in the real estate domain.

Chapter 2

Dataset description

The dataset used in this project contains information related to residential properties. It consists of 100 data points, where each data point represents a single housing unit. The dataset includes 12 features in total, out of which 11 are input features and 1 is the output feature. The output feature is `Price_Category`, which represents the price range of the property and takes three possible values: Low, Medium, and High.

2.1 Class Distribution and Imbalance Analysis

To examine whether the dataset is balanced, the distribution of the target variable `Price_Category` was analyzed. The dataset contains a total of 1200 instances after preprocessing, distributed as follows:

- Medium: 413 instances
- Low: 402 instances
- High: 385 instances

The class distribution indicates that the dataset is **approximately balanced**, as the difference in the number of instances across the three classes is relatively small. Therefore, severe class imbalance is not present, and no aggressive resampling techniques were required. However, stratified data splitting was used to preserve class proportions during training and evaluation.

2.2 Categorical Feature Encoding

Prior to analysis and model training, categorical features were converted into numerical representations. Binary categorical variables (`Has_Balcony`, `Parking_Available`, and `Nearby_Schools`) were encoded using label encoding. The feature `Location` was ordinally encoded as Countryside (0), Suburbs (1), and City Center (2). The ordinal feature `Security_Level` was encoded as Low (0), Medium (1), and High (2). The target variable `Price_Category` was encoded as Low (0), Medium (1), and High (2). Ordinal encoding was applied to features with an inherent ordering. This encoding was implemented using explicit numerical mapping in order to preserve the relative order among categories.

2.3 Missing Value Imputation

Missing values in the dataset were handled using different imputation strategies based on feature characteristics. The feature `Size_sqft` was imputed using the mean value. Numerical features such as `Num_Bedrooms`, `Num_Bathrooms`, `Floor_Number`, `Building_Age_Years`,

and `Distance_to_CityCenter_km` were imputed using the median to reduce the impact of outliers. Categorical features including `Has_Balcony`, `Parking_Available`, `Nearby_Schools`, `Security_Level`, and `Location` were imputed using the most frequent value.

2.4 Correlation Analysis

A correlation matrix was computed using the Pearson correlation coefficient after encoding and imputing all features. The correlation matrix was visualized using a heatmap to analyze linear relationships between input features and the output feature.

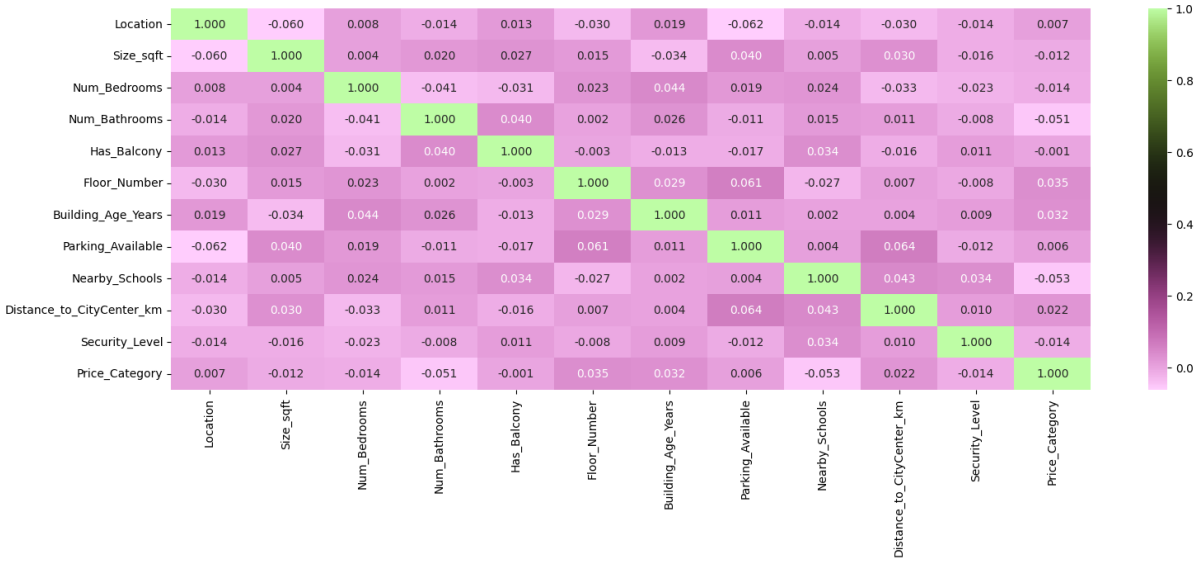

Figure 2.1: Confusion matrix for Price Category classification

2.5 Interpretation of Correlation Results

The correlation analysis reveals that most input features exhibit **very weak linear correlation** with the target variable `Price_Category`. The highest absolute correlation observed between any input feature and the target is below 0.06, indicating the absence of strong linear dependencies. This suggests that the relationship between features and the target variable is likely non-linear or influenced by interactions among multiple features rather than individual attributes.

Additionally, correlations among input features are also generally low, implying minimal multicollinearity in the dataset. These observations justify the use of non-linear machine learning models, which are better suited to capture complex relationships that are not evident through linear correlation analysis.

2.6 Discussion

The weak linear correlations indicate that simple linear models may struggle to capture the underlying patterns in the data. Consequently, models capable of learning non-linear

decision boundaries are expected to perform better on this task. The approximately balanced class distribution further ensures that classification models can be trained without significant bias toward any particular class.

Chapter 3

Dataset pre-processing

Real-world datasets often contain imperfections that can negatively impact machine learning model performance. In this study, several preprocessing challenges were identified and addressed systematically. Each issue is discussed below along with the corresponding solution applied.

3.1 Null and Missing Values

Problem: The dataset contained null and missing values across multiple numerical and categorical features. Missing values can lead to biased results or errors during model training if not handled appropriately.

Solution and Justification: No rows or columns were removed from the dataset because the dataset is relatively small and limited in size. Removing data could have resulted in a significant loss of information. Instead, imputation techniques were applied. The feature `Size_sqft` was imputed using the mean value, while numerical features such as `Num_Bedrooms`, `Num_Bathrooms`, `Floor_Number`, `Building_Age_Years`, and `Distance_to_CityCenter_km` were imputed using the median to reduce sensitivity to outliers. Categorical features were imputed using the most frequent value to preserve existing category distributions.

3.2 Categorical Variables

Problem: Several features in the dataset were categorical in nature. Machine learning algorithms require numerical inputs and cannot directly process categorical values.

Solution and Justification: Categorical encoding was performed based on the nature of each feature. Binary categorical features were encoded using label encoding. Ordinal features were encoded using explicit numerical mapping to preserve the inherent hierarchical relationships within the data. This approach ensured that category ordering was retained where applicable while maintaining numerical compatibility with machine learning algorithms.

3.3 Feature Scaling

Problem: The numerical features in the dataset were measured on different scales. Models such as Logistic Regression, K-Nearest Neighbors, Neural Networks, and Naive Bayes are sensitive to feature magnitude, which can bias model learning if scaling is not applied.

Solution and Justification: Standardization was applied to numerical features including `Size_sqft`, `Num_Bedrooms`, `Num_Bathrooms`, `Floor_Number`, `Building_Age_Years`,

and `Distance_to_CityCenter_km` using the `StandardScaler`. The scaler was fitted only on the training data and subsequently applied to the test data to prevent data leakage. Feature scaling ensured that all numerical attributes contributed equally during model training.

3.4 Class Imbalance Handling

Problem: Although the dataset was approximately balanced, minor class distribution differences existed in the target variable `Price_Category`. Such imbalance can affect the performance of certain classification models.

Solution and Justification: The Synthetic Minority Over-sampling Technique (SMOTE) was applied to the training dataset after feature scaling. SMOTE generates synthetic samples for minority classes, improving class representation while preserving original test data distribution. This approach was applied exclusively to the training data to avoid information leakage and to enhance model generalization.

3.5 Summary

The preprocessing steps applied in this study addressed missing values, categorical data handling, feature scaling, and class imbalance. These steps ensured that the dataset was suitable for training a diverse set of machine learning models, including Logistic Regression, Decision Trees, Random Forests, K-Nearest Neighbors, Naive Bayes, Neural Networks, and K-Means clustering.

Chapter 4

Dataset splitting

Proper dataset splitting is essential to ensure unbiased model evaluation and reliable performance estimation. In this study, the dataset splitting strategy was carefully designed to preserve class distributions and prevent data leakage.

4.1 Data Splitting Strategy

Problem: Random splitting of classification datasets may lead to unequal class distributions between training and testing sets, especially when multiple classes are present. Such imbalance can result in biased model evaluation.

Solution and Justification: A stratified splitting strategy was employed to divide the dataset while preserving the original class proportions of the target variable `Price_Category`. Stratified splitting ensures that each class is proportionally represented in both the training and testing sets, which is particularly important for multi-class classification problems.

4.2 Train-Test Split Ratio

Problem: Selecting an inappropriate split ratio may lead to insufficient training data or unreliable testing performance.

Solution and Justification: The dataset was divided into a training set and a testing set using a 75%–25% split. This ratio provides a sufficient number of samples for model training while reserving an adequate portion of the data for unbiased evaluation.

4.3 Handling of Validation Data

Problem: Model evaluation may be compromised if test data is used during model selection or parameter tuning.

Solution and Justification: No separate validation set was used in this study. Model evaluation was performed exclusively on the test set

Chapter 5

Model training & testing

This section presents the supervised and unsupervised learning models applied in this study. Each supervised model is briefly introduced, followed by a table reporting its evaluation metrics. Finally, a separate subsection explains why the overall model scores are low.

5.1 Supervised Learning Models

All supervised models were trained on the training dataset and evaluated on an unseen test dataset. Performance was measured using accuracy, precision, recall, and F1-score.

5.2 Decision Tree Classifier

A Decision Tree classifier was applied due to its interpretability and its ability to model non-linear relationships through decision rules.

Table 5.1: Performance Metrics for Decision Tree Classifier

Metric	**Value (%)**
Accuracy	35.00
Precision	34.97
Recall	35.00
F1-score	34.93

5.3 Random Forest Classifier

Random Forest is an ensemble model that aggregates multiple decision trees to improve stability and reduce variance.

Table 5.2: Performance Metrics for Random Forest Classifier

Metric	**Value (%)**
Accuracy	35.33
Precision (macro)	35.21
Recall (macro)	35.27
F1-score (macro)	35.23

5.4 Logistic Regression

Logistic Regression was used as a baseline linear classifier for multi-class classification.

Table 5.3: Performance Metrics for Logistic Regression

Metric	**Value (%)**
Accuracy	33.00
Precision (macro)	33.58
Recall (macro)	33.00
F1-score (macro)	33.14

5.5 Naive Bayes Classifier

Naive Bayes is a probabilistic classifier based on Bayes' theorem. It is computationally efficient and commonly used as a baseline model.

Table 5.4: Performance Metrics for Naive Bayes Classifier

Metric	**Value (%)**
Accuracy	34.00
Precision (macro)	34.73
Recall (macro)	33.95
F1-score (macro)	34.16

5.6 K-Nearest Neighbors (KNN)

KNN is a distance-based classifier that predicts the class of a sample by majority voting among its nearest neighbors.

Table 5.5: Performance Metrics for K-Nearest Neighbors Classifier

Metric	**Value (%)**
Accuracy	36.33
Precision (macro)	35.92
Recall (macro)	36.18
F1-score (macro)	35.34

5.7 Neural Network

A Neural Network classifier implemented using `sklearn` with a softmax output layer was used to capture non-linear relationships.

Table 5.6: Performance Metrics for Neural Network Classifier

Metric	Value (%)
Accuracy	33.33
Precision	11.11
Recall	33.33
F1-score	16.67

5.8 Unsupervised Learning: K-Means Clustering

To treat the problem as an unsupervised learning task, K-Means clustering was applied to explore inherent structure in the dataset. The number of clusters was set to $k = 3$, corresponding to the number of price categories.

Table 5.7: K-Means Clustering Evaluation

Metric	Value
Number of Clusters (k)	3
Silhouette Score	0.088

5.9 Discussion on Low Model Performance

The relatively low performance observed across supervised models can be explained by intrinsic characteristics of the dataset. The correlation analysis showed very weak linear relationships between individual input features and the target variable, indicating that no single feature strongly predicts `Price_Category`. Furthermore, the feature distributions of the three classes overlap substantially, which creates ambiguous decision boundaries and increases misclassification.

The dataset size also limits the ability of models—especially Neural Networks—to learn stable and generalizable patterns. Although SMOTE was applied after splitting and scaling to address minor class imbalance, synthetic oversampling cannot resolve strong overlap between classes and may not improve separability. The low Silhouette Score obtained from K-Means clustering further supports this conclusion, indicating that the feature space does not naturally form well-separated clusters. As a result, both supervised classification and unsupervised clustering remain challenging for this dataset.

Chapter 6

Model Comparison Analysis

This section compares all classification models based on predictive performance. The comparison is presented in three stages: (i) analysis of accuracy, precision, recall, and F1-score, (ii) confusion matrix interpretation, and (iii) macro AUC score analysis. (ROC curve discussion is intentionally omitted here and will be included later.)

6.1 Accuracy, Precision, Recall, and F1-score Analysis

The overall performance of the supervised classifiers is low. Since the task is a three-class classification problem, a random baseline accuracy is approximately $1/3 \approx 33.33\%$. Most models achieve accuracy values close to this baseline, indicating that the classes are difficult to separate using the current feature set.

Among the tested models, K-Nearest Neighbors (KNN) achieved the highest accuracy (36.33%), followed by Random Forest (35.33%) and Decision Tree (35.00%). Naive Bayes achieved 34.00% accuracy, while Logistic Regression and the Neural Network achieved 33.00% and 33.33%, respectively. The macro-averaged precision, recall, and F1-score values follow the same trend and remain close to the accuracy scores, suggesting consistent but weak classification ability across models.

The Neural Network model reported particularly low precision (11.11%) while maintaining recall near the baseline (33.33%). This pattern indicates a high false-positive rate for at least one class, which reduces precision substantially and results in a low F1-score (16.67%). This suggests unstable class discrimination under the current training configuration.

6.2 Confusion Matrix Analysis

Confusion matrices provide a detailed view of class-wise performance by showing how frequently each true class is predicted as each output class. Across all models, the diagonal values (correct predictions) are only moderately larger than off-diagonal values (misclassifications). This indicates substantial overlap among the three classes and frequent confusion between categories.

6.2.1 Decision Tree

Figure 6.1 shows the confusion matrix for the Decision Tree classifier.

The Decision Tree shows moderate correct predictions but misclassifies all classes at similar rates, implying that the learned splits are not strongly discriminative.

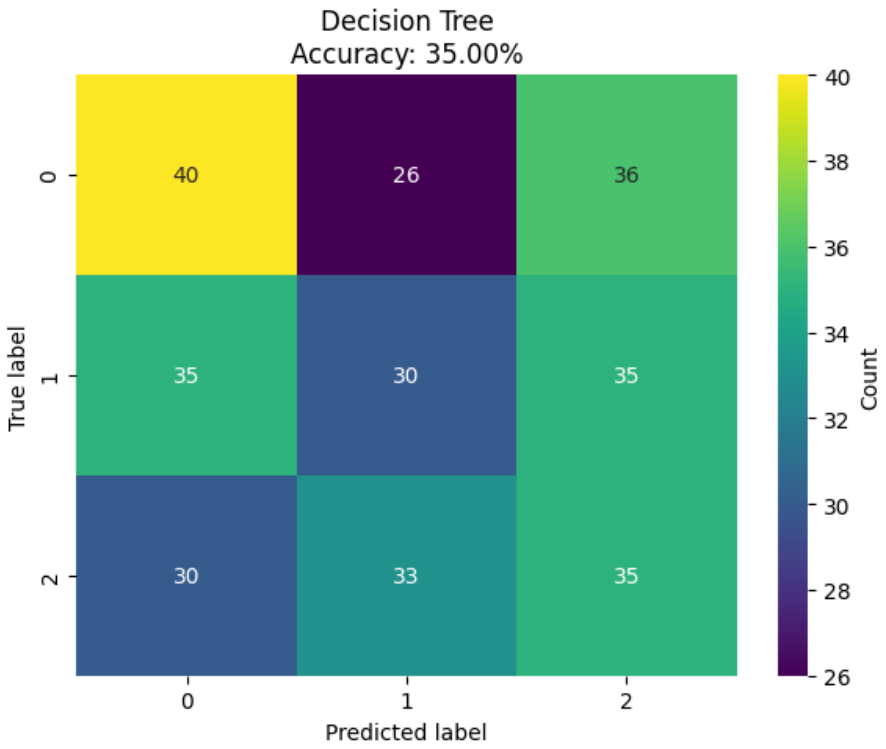

Figure 6.1: Confusion Matrix for Decision Tree Classifier

6.2.2 Random Forest

Figure 6.2 shows the confusion matrix for the Random Forest classifier.

Random Forest slightly improves correct classification for the middle class (37 correct), but still shows substantial confusion across classes, indicating limited separability even with ensemble learning.

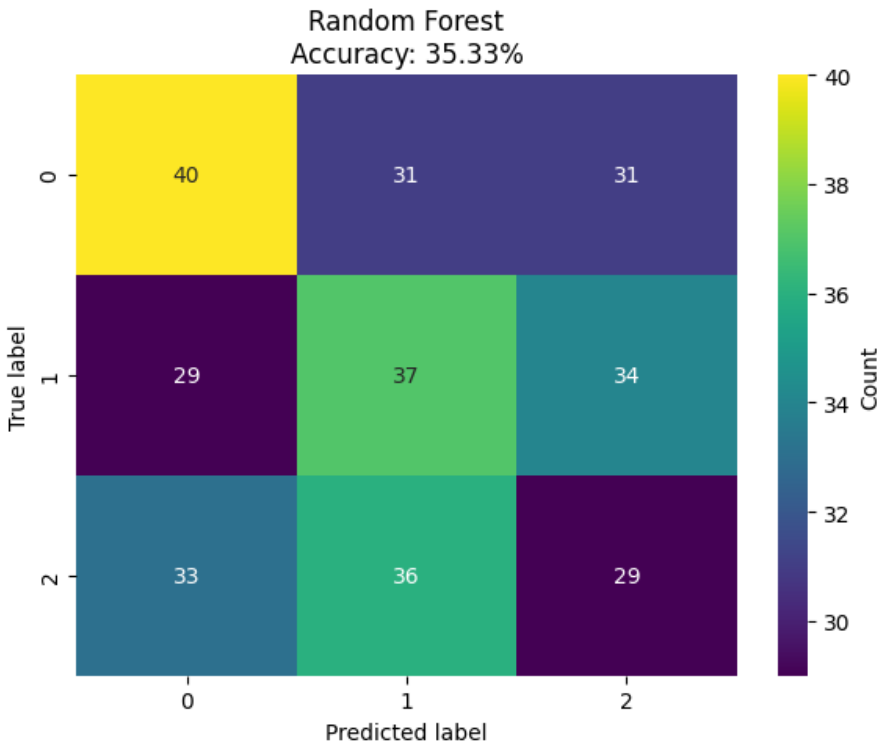

Figure 6.2: Confusion Matrix for Random Forest Classifier

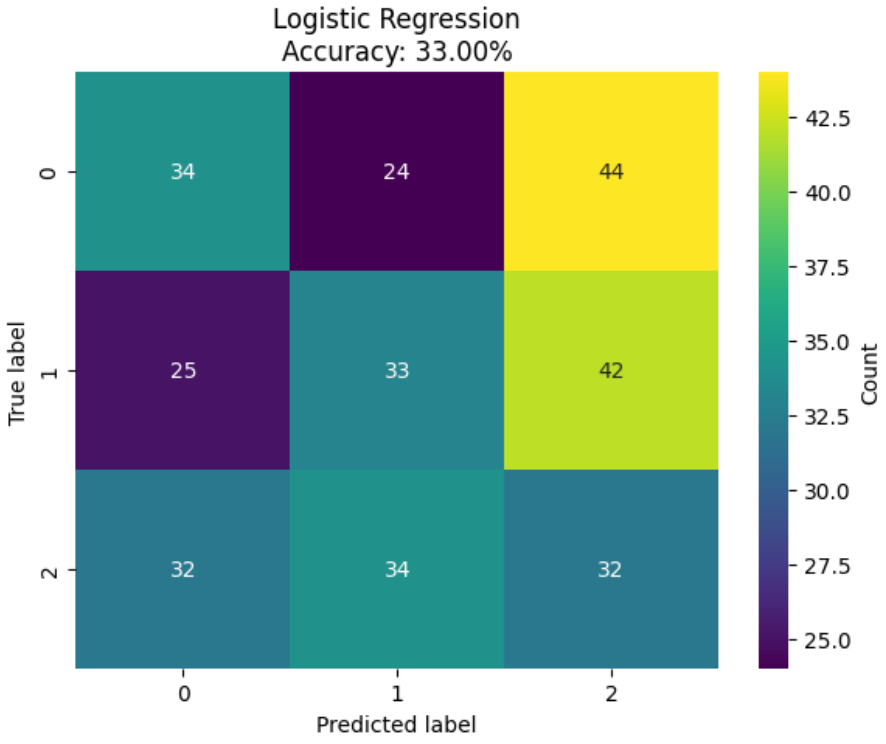

Figure 6.3: Confusion Matrix for Logistic Regression classifier

6.2.3 Logistic Regression

Figure 6.3 shows the confusion matrix for the Logistic Regression classifier.

Logistic Regression shows heavy misclassification into the third column (e.g., 44 and 42), suggesting that a linear decision boundary is insufficient and the model tends to assign samples disproportionately to one class.

6.2.4 Naive Bayes

Figure 6.4 shows the confusion matrix for the Naive Bayes classifier.

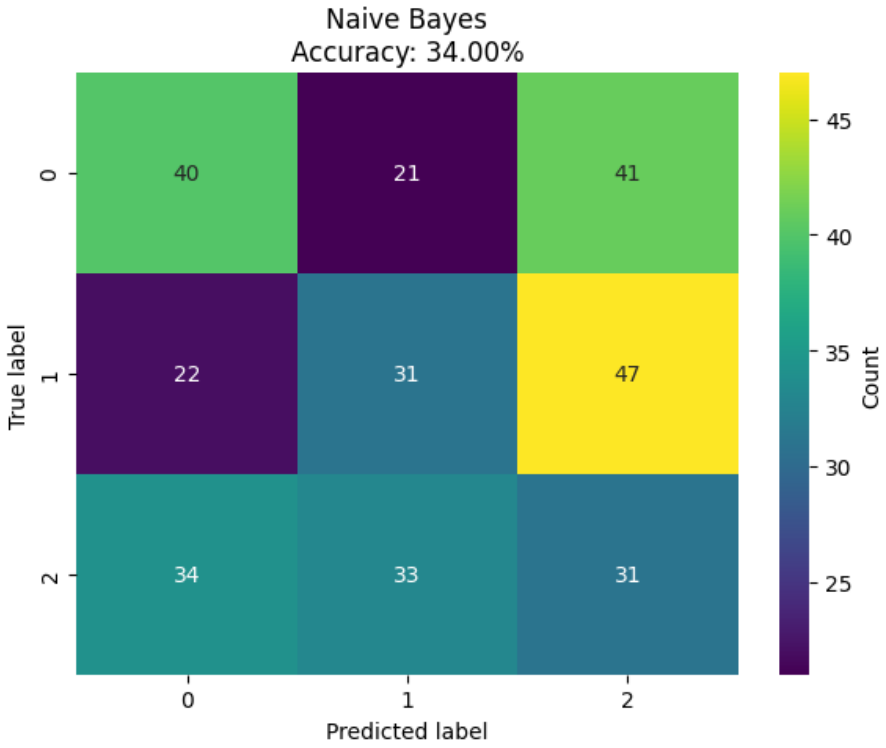

Figure 6.4: Confusion Matrix for Naive Bayes Classifier

Naive Bayes frequently predicts the third class (notably 47), reflecting that its conditional independence assumption does not align well with the feature relationships in the dataset.

6.2.5 K-Nearest Neighbors (KNN)

KNN achieves the best performance for the first class (54 correct), but still suffers from considerable confusion in the remaining classes, which is consistent with the limited overall accuracy.

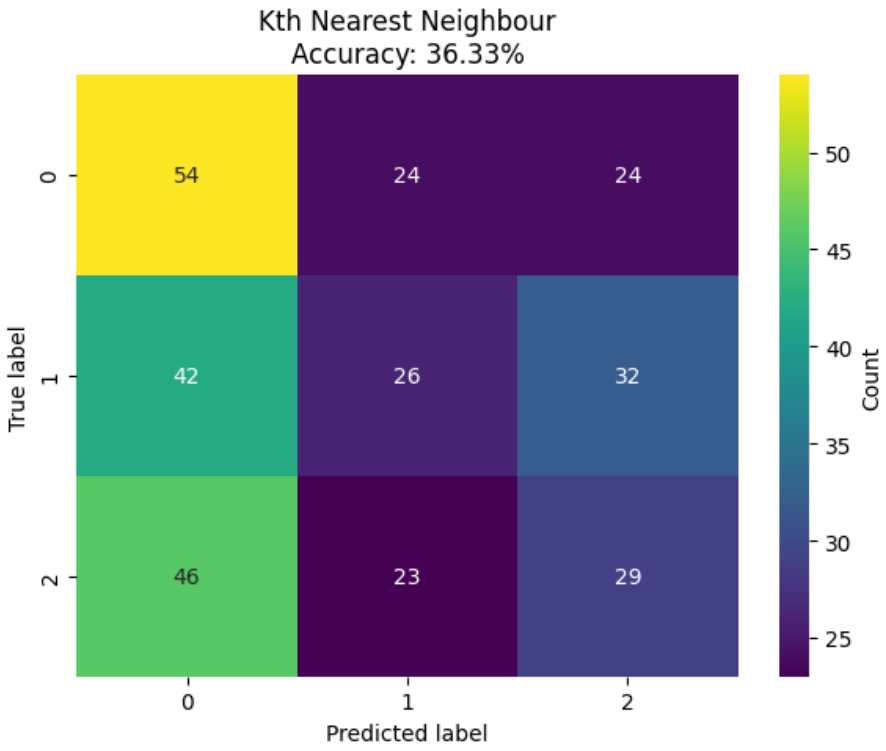

Figure 6.5: Confusion Matrix for K-Nearest Neighbors Classifier

6.2.6 Neural Network

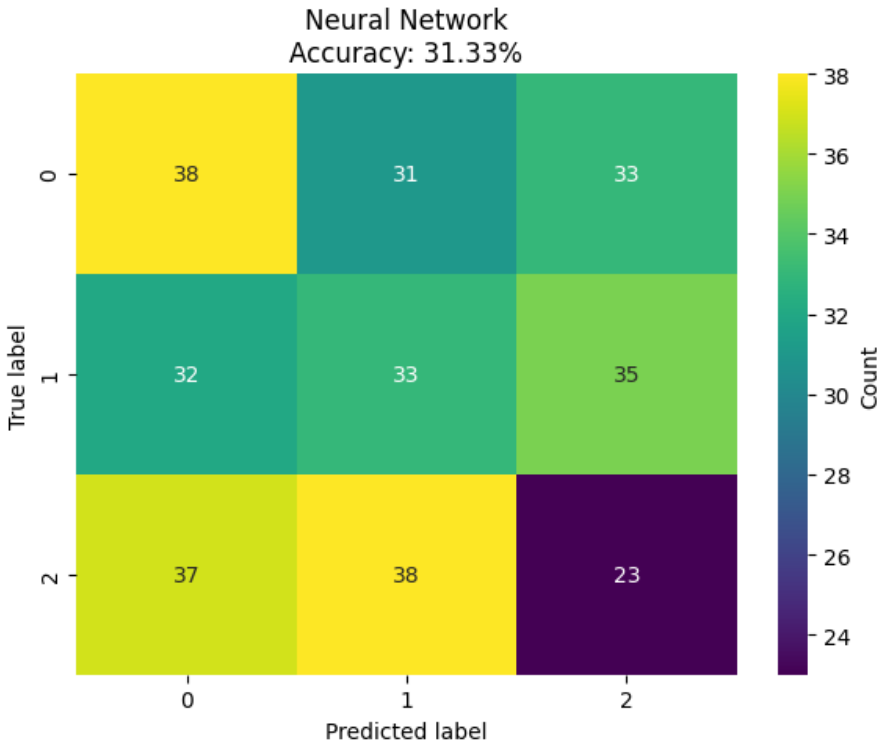

Figure 6.6: Confusion Matrix for Neural Network Classifier

The Neural Network exhibits weak diagonal dominance, especially for the third class (23 correct). The spread of off-diagonal values indicates poor class discrimination, consistent with its low precision and F1-score.

6.3 AUC Score Analysis (ROC Curve Omitted Here)

To further evaluate classification performance, macro-averaged AUC scores were computed using the One-vs-Rest (OvR) strategy. AUC values near 0.50 indicate weak class separability and performance close to random ranking.

6.3.1 Macro AUC (Scaled Features)

The macro AUC (OvR) values for scaled features are:

- Decision Tree: 0.5124
- Random Forest: 0.5204
- Logistic Regression: 0.4813
- Naive Bayes: 0.4860
- KNN: 0.5259
- Neural Network: 0.4987

KNN achieved the highest scaled AUC (0.5259), followed by Random Forest (0.5204). However, all values remain close to 0.50, confirming weak separability among classes.

6.3.2 Macro AUC (Unscaled Features)

The macro AUC (OvR) values for unscaled features are:

- Decision Tree: 0.4914
- Random Forest: 0.5193
- Logistic Regression: 0.4688
- Naive Bayes: 0.5000
- KNN: 0.4825
- Neural Network: 0.5078

A noticeable drop is observed for KNN when using unscaled features (0.4825), which is expected because KNN relies on distance computations that are highly sensitive to feature scaling.

6.4 ROC Curves (Multiclass One-vs-Rest)

Figure **??** presents the ROC curves for all six classifiers using the One-vs-Rest (OvR) strategy. Each image contains two panels: *scaled features* (left) and *unscaled features* (right). Overall, most curves lie close to the random baseline (diagonal), consistent with AUC values near 0.50–0.53, indicating modest class separability.

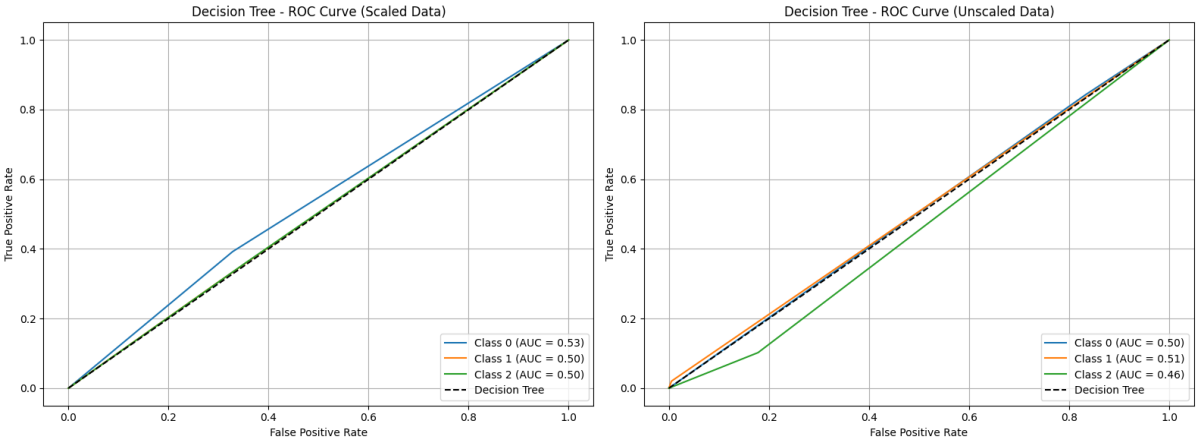

Figure 6.7: Decision Tree: ROC curves (scaled vs unscaled).

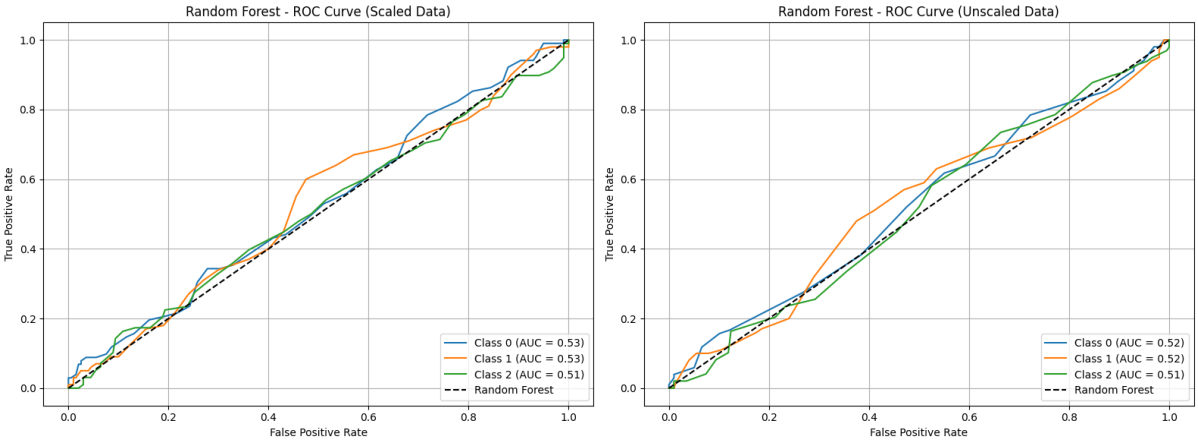

Figure 6.8: Random Forest: ROC curves (scaled vs unscaled).

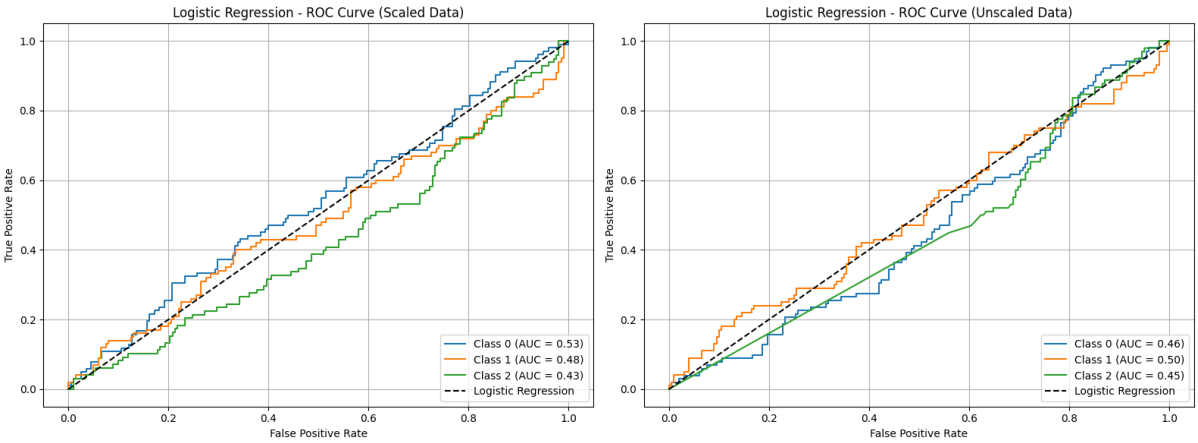

Figure 6.9: Logistic Regression: ROC curves (scaled vs unscaled).

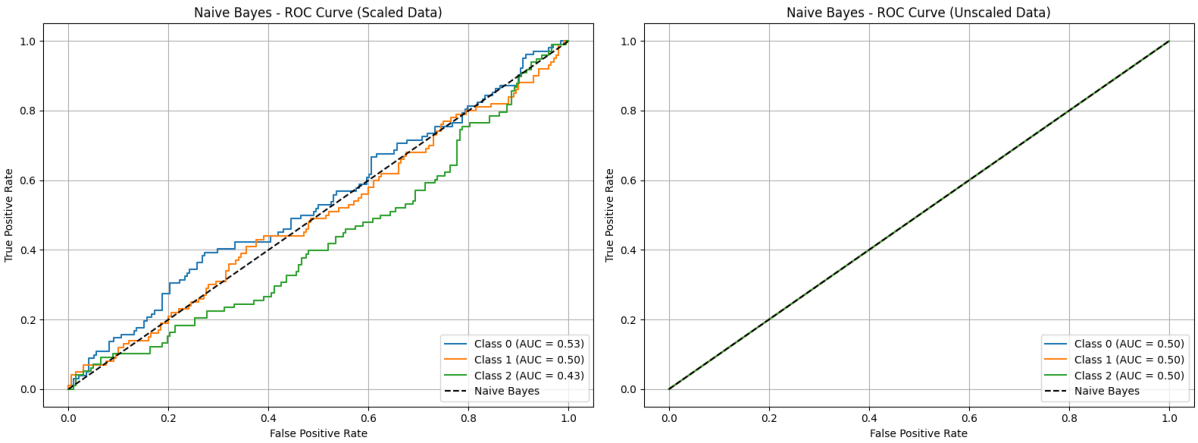

Figure 6.10: Naive Bayes: ROC curves (scaled vs unscaled).

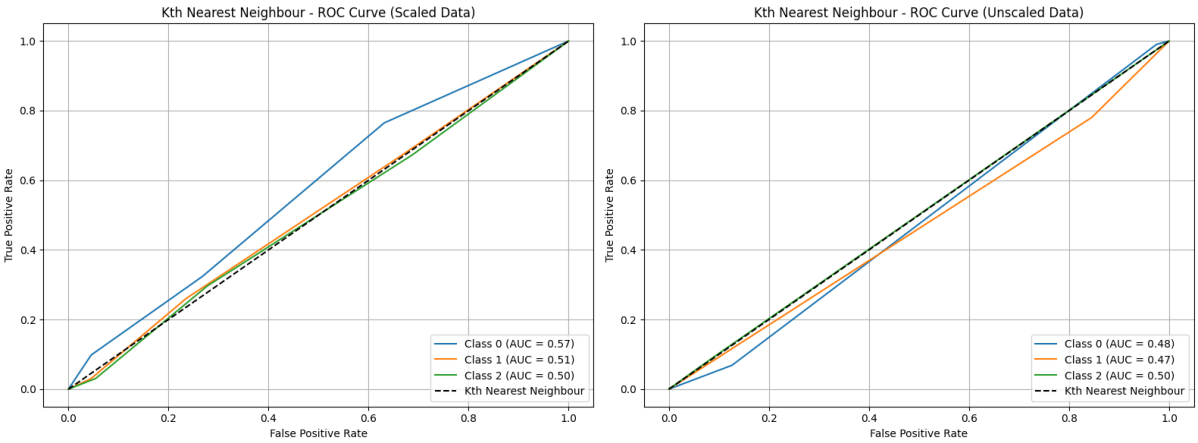

Figure 6.11: KNN: ROC curves (scaled vs unscaled).

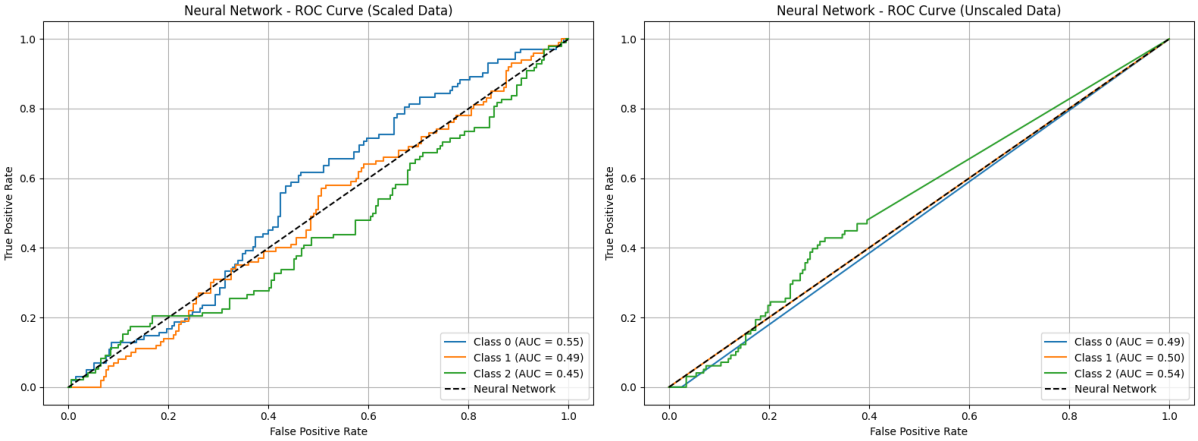

Figure 6.12: Neural Network (MLP): ROC curves (scaled vs unscaled).

6.4.1 Interpretation of ROC Curves

- **Overall performance near chance:** Across models, the ROC curves generally remain close to the diagonal, consistent with micro/macro AUC values around 0.48–0.53. This suggests the feature set provides limited discrimination among the three price categories.

- **Effect of scaling:** The most visible improvement with scaling is expected for **KNN**, where scaled performance improves substantially compared to unscaled. In contrast, **Random Forest** curves should appear similar in both panels, as tree-based models are typically insensitive to feature scaling.

- **Class-wise difficulty:** The ROC curve for **Class 0** is generally expected to be higher than the other classes for most models, indicating it is the easiest class to separate. **Class 2** often produces curves close to (or occasionally below) the diagonal for several models, indicating it is the most challenging class.

- **Model-specific behavior:**

- **KNN (scaled):** strongest overall separation; scaled curves should consistently dominate unscaled curves.
- **Random Forest:** stable behavior; minimal difference between scaled and unscaled panels.
- **Logistic Regression and Naive Bayes:** weaker separation, especially for Class 2; curves may hug the diagonal more closely.
- **MLP:** scaling does not consistently improve performance in this experiment; unscaled curves may be slightly better for some classes.

6.5 Summary

KNN achieved the best overall performance, yielding the highest accuracy and the strongest scaled micro/macro ROC–AUC, while Random Forest remained the most consistent model with minimal sensitivity to feature scaling. The confusion matrices reveal considerable cross-class confusion for every classifier, and the ROC–AUC values clustering near 0.50 indicate that the current feature set provides only limited separability among the three price categories. A detailed ROC-curve discussion (including scaled vs. unscaled comparisons and class-wise behavior) is presented in the following subsection.

Chapter 7

Conclusion

This project investigated the prediction of residential property price categories (`Low`, `Medium`, `High`) using multiple supervised learning models and one unsupervised clustering approach. The supervised models evaluated were Decision Tree, Random Forest, Logistic Regression, Naive Bayes, K-Nearest Neighbors, and a Neural Network, while K-Means was applied to explore whether natural clusters exist in the feature space.

7.1 Understanding from the Results

The experimental results show that all supervised models achieved relatively low predictive performance. Accuracy values ranged approximately from 33% to 36%, which is close to the random baseline for a three-class classification task ($\approx 33.33\%$). Among the supervised approaches, K-Nearest Neighbors achieved the best overall accuracy (36.33%) and also produced the highest macro AUC in the scaled setting (0.5259). Random Forest produced stable results with slightly lower accuracy (35.33%) and competitive AUC values.

The unsupervised K-Means clustering produced a low Silhouette Score (0.088), indicating weak cluster separation. This suggests that the dataset does not naturally form well-separated clusters in the given feature space, which is consistent with the difficulty observed in supervised classification.

7.2 Comments on Model Performance

Overall, the models struggled to correctly discriminate among the three price categories. The confusion matrix analysis showed substantial misclassification across classes for all models, indicating overlapping class distributions. Feature scaling improved the performance of distance-based methods such as KNN, which is expected because distance calculations are sensitive to feature magnitude. In contrast, tree-based models showed minor changes with scaling, as they are generally less dependent on feature scaling.

The Neural Network achieved accuracy close to baseline but exhibited very low precision, suggesting unstable class predictions and high false positives. This outcome indicates that the model configuration and available data were not sufficient to learn strong decision boundaries for the three classes.

7.3 Reasons for Observed Results

The low performance can be explained by intrinsic characteristics of the dataset and the classification task:

- **Weak feature–target relationships:** Correlation analysis indicated very weak linear relationships between individual features and `Price_Category`, implying that no single attribute strongly predicts the output.
- **Overlapping class distributions:** The feature values for different classes overlap substantially, creating ambiguous decision boundaries and increasing misclassification between neighboring categories.
- **Limited separability in feature space:** The low Silhouette Score from K-Means supports the conclusion that the feature space does not naturally separate into distinct groups corresponding to the price categories.
- **Dataset size and complexity:** The dataset size is limited relative to the complexity of a three-class classification task, which restricts the learning capability of more complex models such as Neural Networks.
- **Effect of oversampling:** Although SMOTE was applied after splitting and scaling to address minor imbalance, oversampling cannot resolve class overlap and may not significantly improve separability when the feature patterns are not strongly discriminative.

7.4 Challenges Faced

Several challenges were encountered during the project:

- **Handling missing values:** The dataset contained missing values in both numerical and categorical features, requiring careful imputation strategies (mean/median/mode) to avoid information loss.
- **Encoding categorical features:** Multiple categorical features required encoding using a combination of label encoding and ordinal mapping while preserving valid hierarchy where applicable.
- **Preventing data leakage:** Preprocessing steps such as feature scaling and SMOTE needed to be applied strictly after the train-test split and only on the training data to maintain unbiased evaluation.
- **Low feature separability:** The limited separability between classes made it difficult for all models to achieve strong performance, requiring careful interpretation beyond accuracy alone.
- **Model comparison across diverse algorithms:** Ensuring a consistent evaluation framework across multiple models (metrics, scaling, and probability outputs for AUC) was necessary for fair comparison.

7.5 Future Improvements

To improve predictive performance, future work may focus on incorporating additional informative features, applying feature engineering, exploring advanced models (e.g., gradient boosting), and performing systematic hyperparameter tuning with cross-validation.