- ACM476 TERM PROJECT PHASE 2

Melbourne Housing

Kaan Akkök, Hasan Emir Çilesiz

21.05.2025

Content

1 Regression

2 Classification

Regression:

After conducting Exploratory Data Analysis (EDA), we determined that some features in the dataset introduced noise and were unrelated to price values. Therefore, we removed “Address”, “SellerG”, “Date”, “Propertycount”, and “Postcode”.

Target variable:

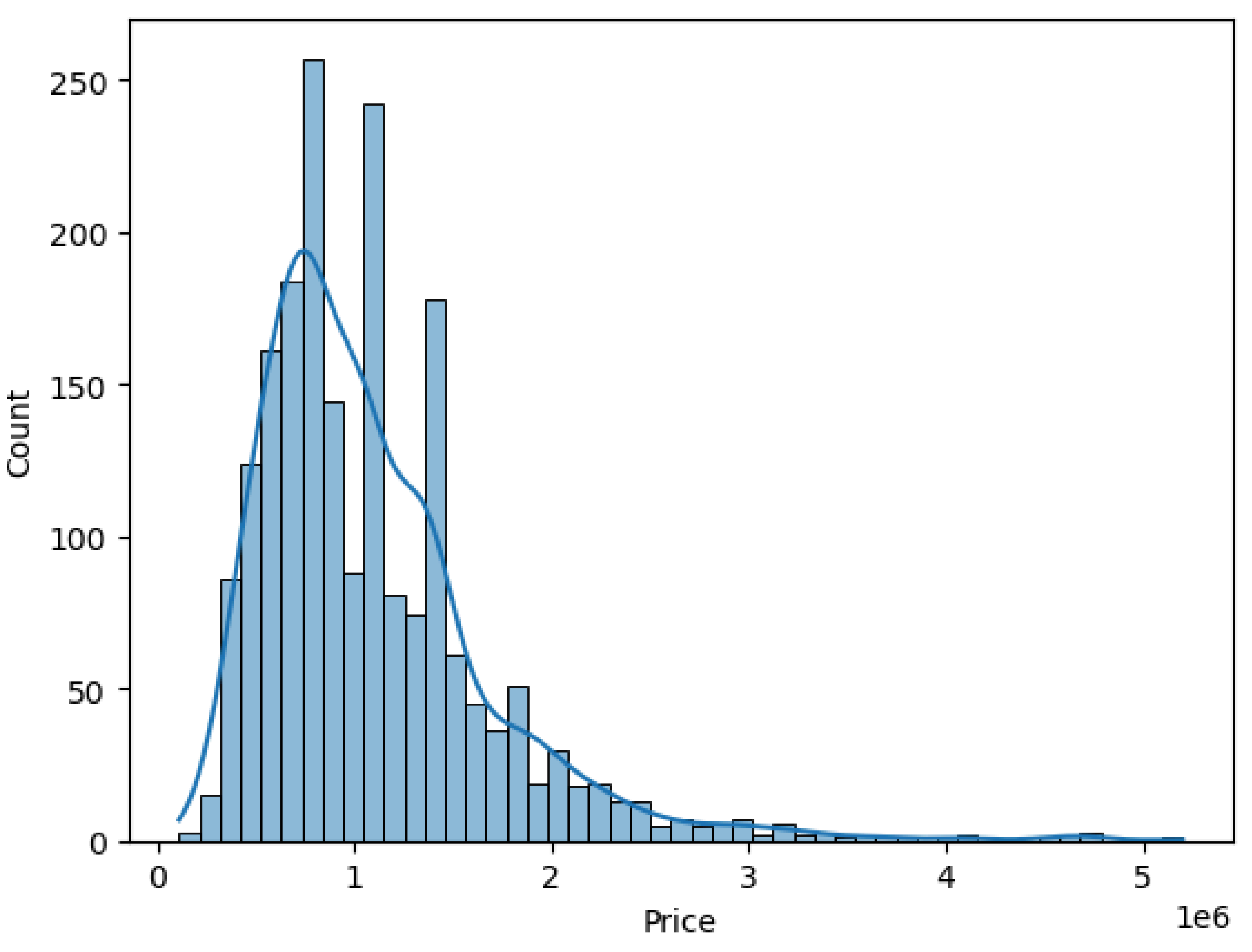
- Price

Independent variables:

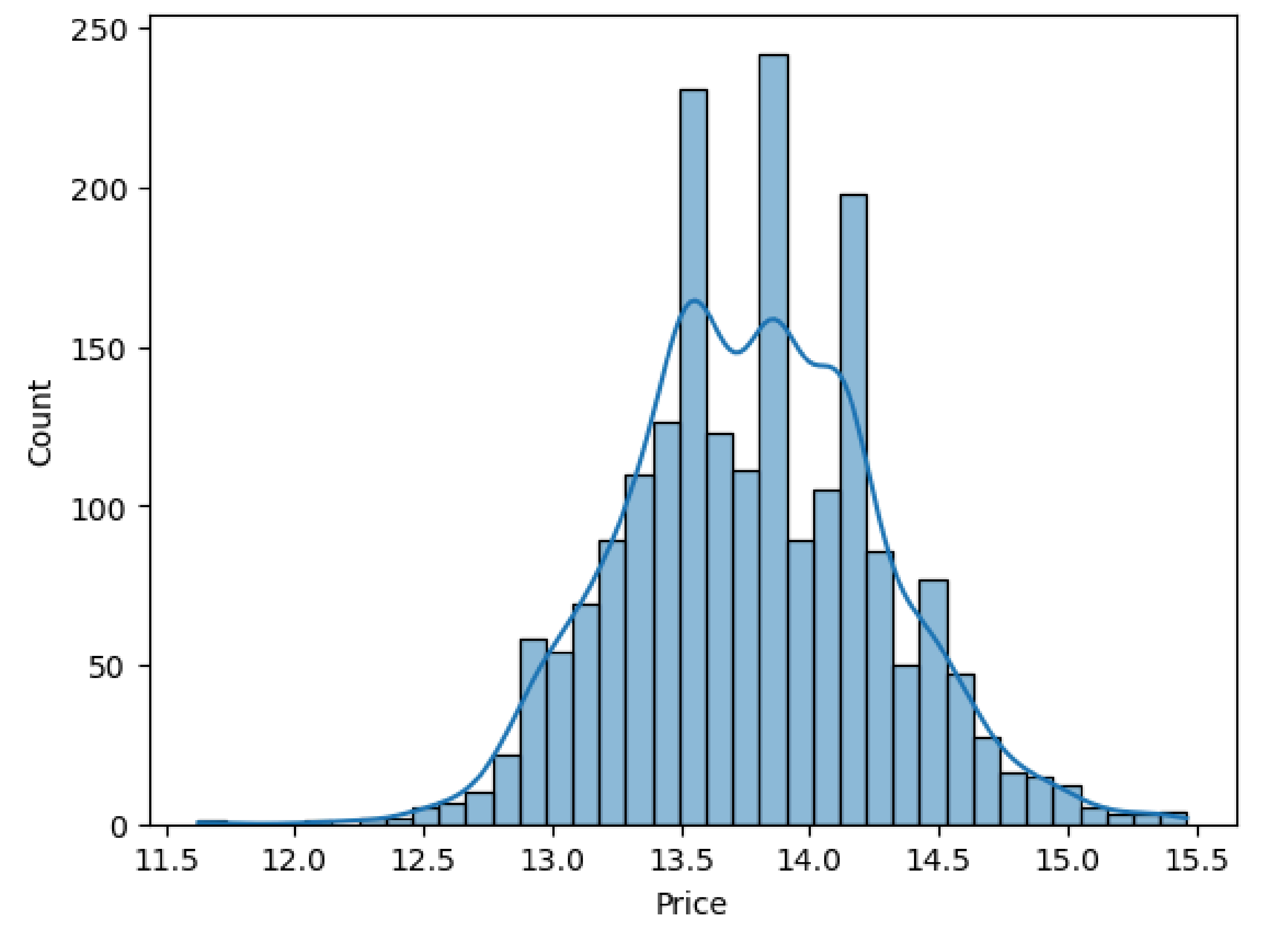
- Suburb
- Rooms
- Type
- Method
- Distance
- Bedroom2
- Bathroom
- Car
- Landsize
- CouncilArea
- Lattitude
- Longtitude
- Regionname

Regression:

To analyze the Price target variable, we examined its distribution table and found that it is right-skewed.

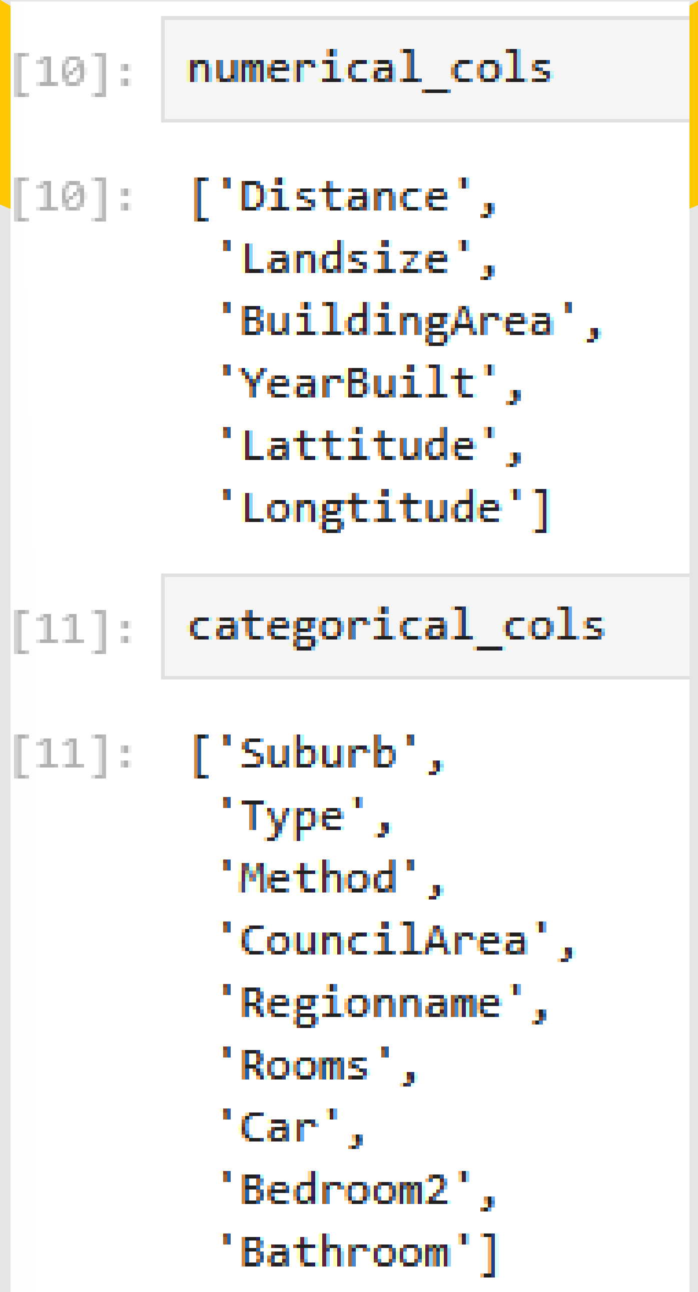


We applied Log-Transformation to the price values for normalization with np.log1p.



Regression:

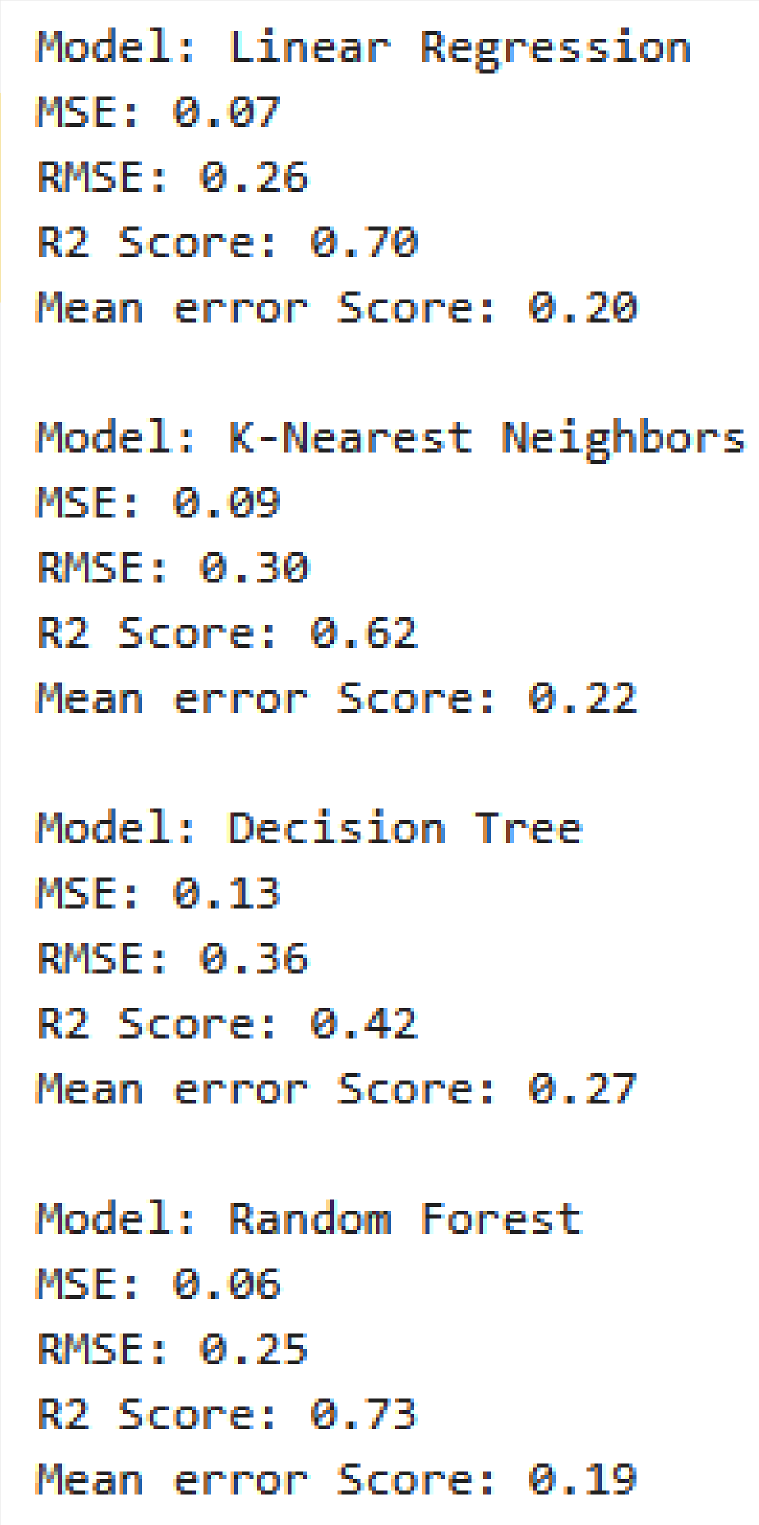
- We identified the numerical and categorical features to apply standard scaling and one-hot encoding.
- Rooms, Car, Bedroom, and Bathroom are numerical features but behave like categorical variables. Therefore, we reclassified them from numerical to categorical.
- Then, we applied standard scaling to numerical features and one-hot encoding to categorical features.



```
[10]: numerical_cols

[10]: ['Distance',
       'Landsize',
       'BuildingArea',
       'YearBuilt',
       'Lattitude',
       'Longtitude']

[11]: categorical_cols

[11]: ['Suburb',
       'Type',
       'Method',
       'CouncilArea',
       'Regionname',
       'Rooms',
       'Car',
       'Bedroom2',
       'Bathroom']
```

Regression:

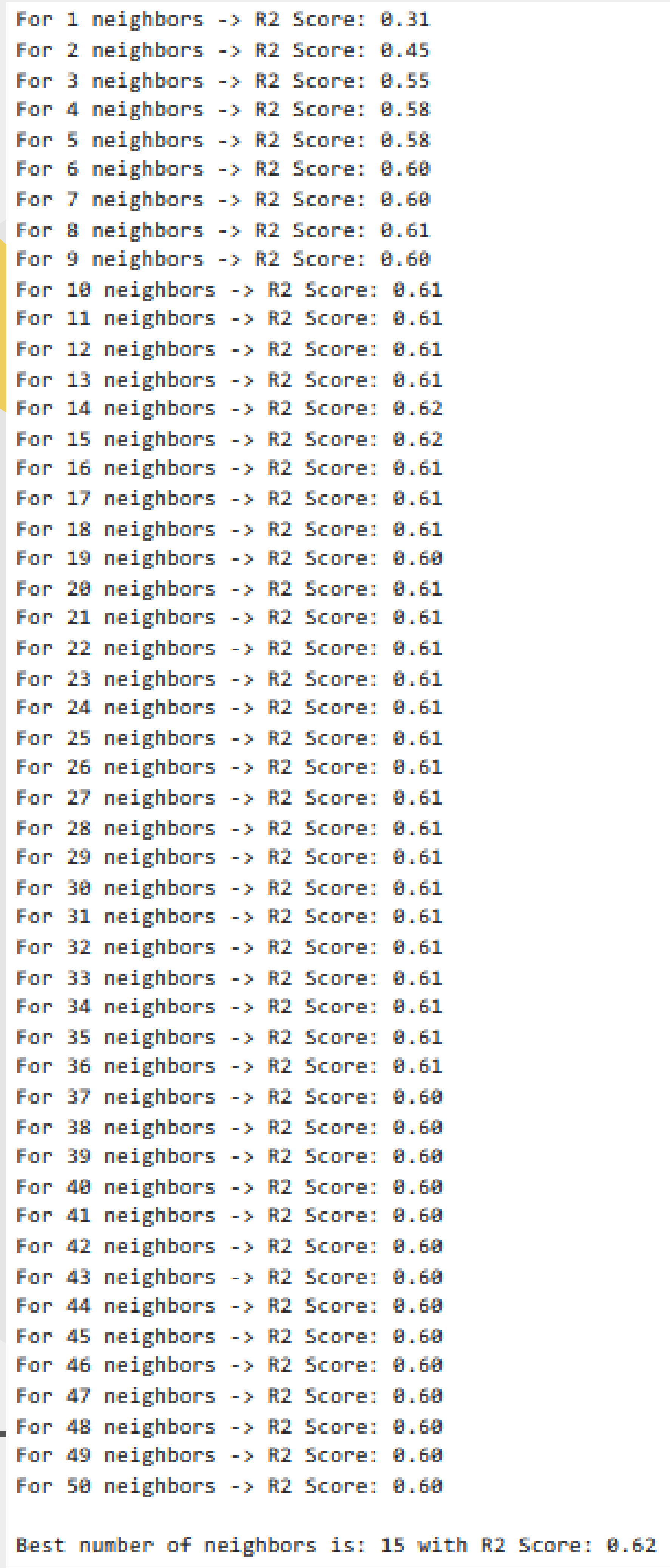
- We split the independent variables into 80% training and 20% testing sets.
- Then, we initialized the following regression models:
    - Linear Regression
    - K-Nearest Neighbors (n_neighbors=15)
    - Decision Tree (random_state=42)
    - Random Forest (random_state=42)
- Finally, we evaluated the performance of these models using the following metrics:
    - Mean Square Error
    - Root Mean Square Error
    - R2 Score
    - Mean Error Score
- The best-performing model, according to the R^2 score, is Random Forest.



```
Model: Linear Regression
MSE: 0.07
RMSE: 0.26
R2 Score: 0.70
Mean error Score: 0.20

Model: K-Nearest Neighbors
MSE: 0.09
RMSE: 0.30
R2 Score: 0.62
Mean error Score: 0.22

Model: Decision Tree
MSE: 0.13
RMSE: 0.36
R2 Score: 0.42
Mean error Score: 0.27

Model: Random Forest
MSE: 0.06
RMSE: 0.25
R2 Score: 0.73
Mean error Score: 0.19
```

Regression:

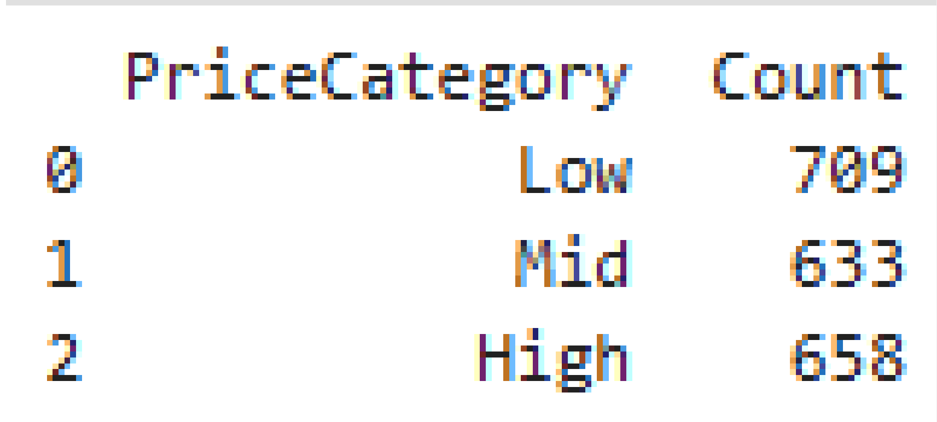
To determine the optimal number of neighbors for the KNN model, we implemented a for loop that iterates 50 times, checking the R² score to identify the best parameter.



```
For 1 neighbors -> R2 Score: 0.31
For 2 neighbors -> R2 Score: 0.45
For 3 neighbors -> R2 Score: 0.55
For 4 neighbors -> R2 Score: 0.58
For 5 neighbors -> R2 Score: 0.58
For 6 neighbors -> R2 Score: 0.60
For 7 neighbors -> R2 Score: 0.60
For 8 neighbors -> R2 Score: 0.61
For 9 neighbors -> R2 Score: 0.60
For 10 neighbors -> R2 Score: 0.61
For 11 neighbors -> R2 Score: 0.61
For 12 neighbors -> R2 Score: 0.61
For 13 neighbors -> R2 Score: 0.61
For 14 neighbors -> R2 Score: 0.62
For 15 neighbors -> R2 Score: 0.62
For 16 neighbors -> R2 Score: 0.61
For 17 neighbors -> R2 Score: 0.61
For 18 neighbors -> R2 Score: 0.61
For 19 neighbors -> R2 Score: 0.60
For 20 neighbors -> R2 Score: 0.61
For 21 neighbors -> R2 Score: 0.61
For 22 neighbors -> R2 Score: 0.61
For 23 neighbors -> R2 Score: 0.61
For 24 neighbors -> R2 Score: 0.61
For 25 neighbors -> R2 Score: 0.61
For 26 neighbors -> R2 Score: 0.61
For 27 neighbors -> R2 Score: 0.61
For 28 neighbors -> R2 Score: 0.61
For 29 neighbors -> R2 Score: 0.61
For 30 neighbors -> R2 Score: 0.61
For 31 neighbors -> R2 Score: 0.61
For 32 neighbors -> R2 Score: 0.61
For 33 neighbors -> R2 Score: 0.61
For 34 neighbors -> R2 Score: 0.61
For 35 neighbors -> R2 Score: 0.61
For 36 neighbors -> R2 Score: 0.61
For 37 neighbors -> R2 Score: 0.60
For 38 neighbors -> R2 Score: 0.60
For 39 neighbors -> R2 Score: 0.60
For 40 neighbors -> R2 Score: 0.60
For 41 neighbors -> R2 Score: 0.60
For 42 neighbors -> R2 Score: 0.60
For 43 neighbors -> R2 Score: 0.60
For 44 neighbors -> R2 Score: 0.60
For 45 neighbors -> R2 Score: 0.60
For 46 neighbors -> R2 Score: 0.60
For 47 neighbors -> R2 Score: 0.60
For 48 neighbors -> R2 Score: 0.60
For 49 neighbors -> R2 Score: 0.60
For 50 neighbors -> R2 Score: 0.60

Best number of neighbors is: 15 with R2 Score: 0.62
```

Classification:

In the classification phase, we divided house prices into three categories: Low, Mid, and High.

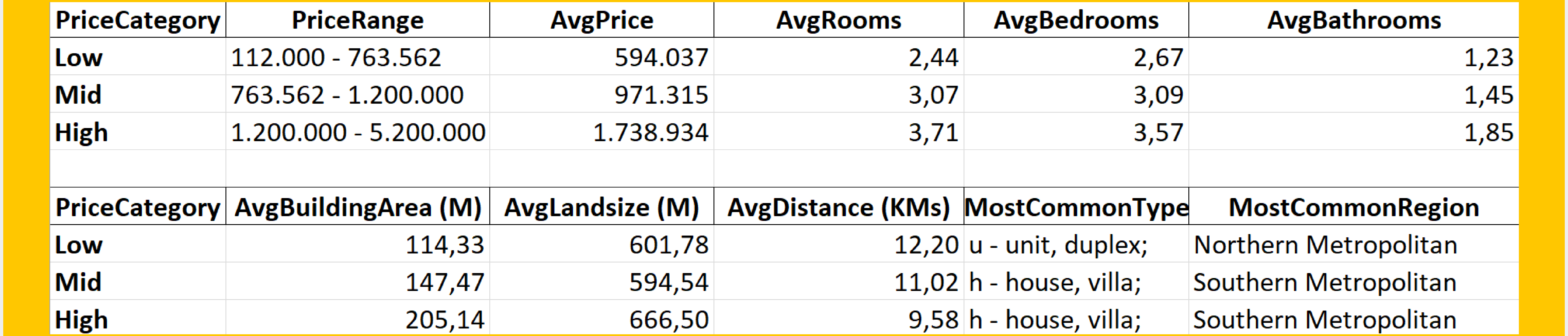


```
  PriceCategory  Count
0           Low    709
1           Mid    633
2          High    658
```

After applying the log(x + 1) transformation, the resulting class distribution was relatively balanced in terms of sample size.

However, if there had been a significant imbalance, it could have negatively impacted the model's performance by making it biased toward the majority class.

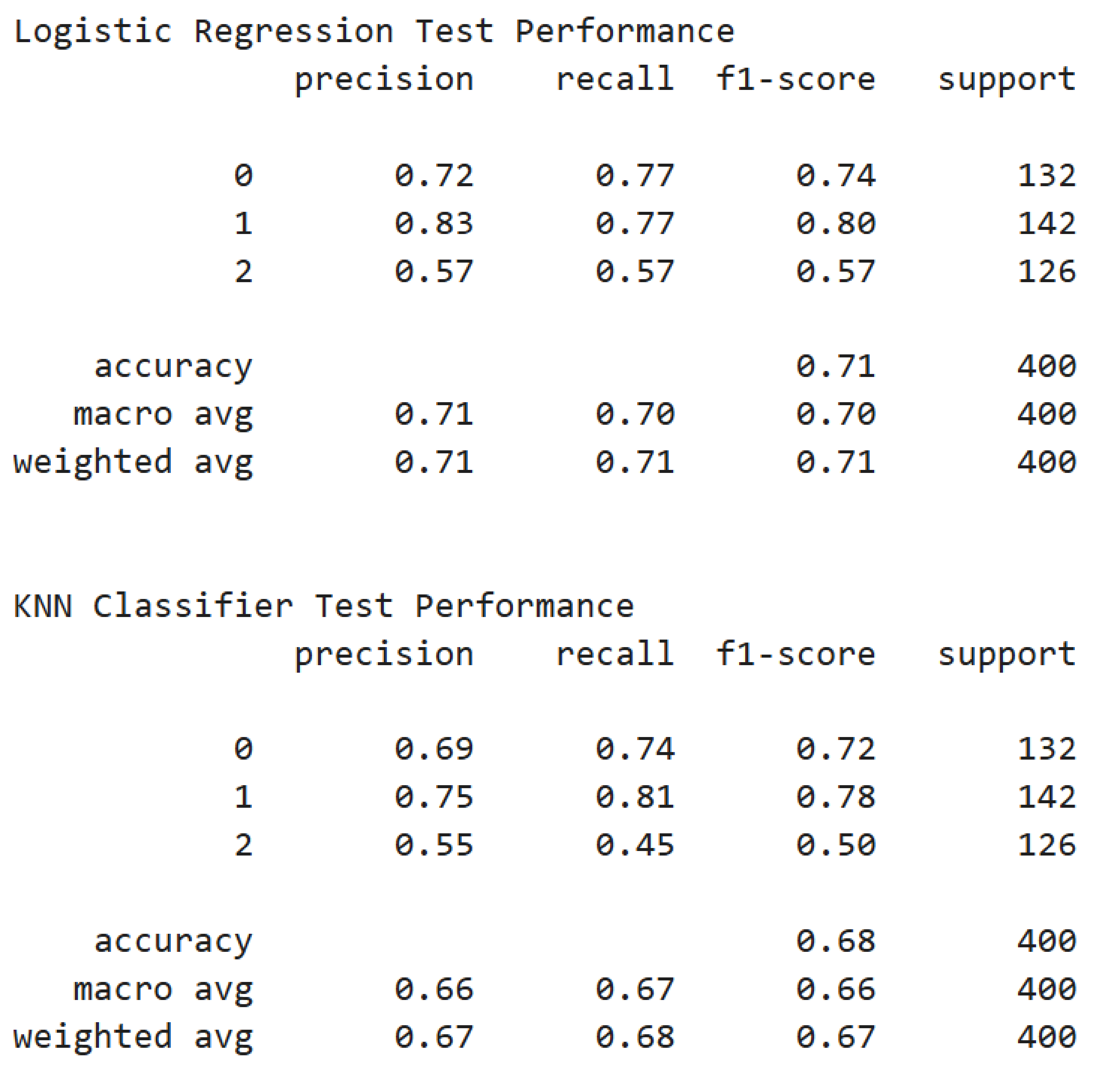
Price Category Summary:



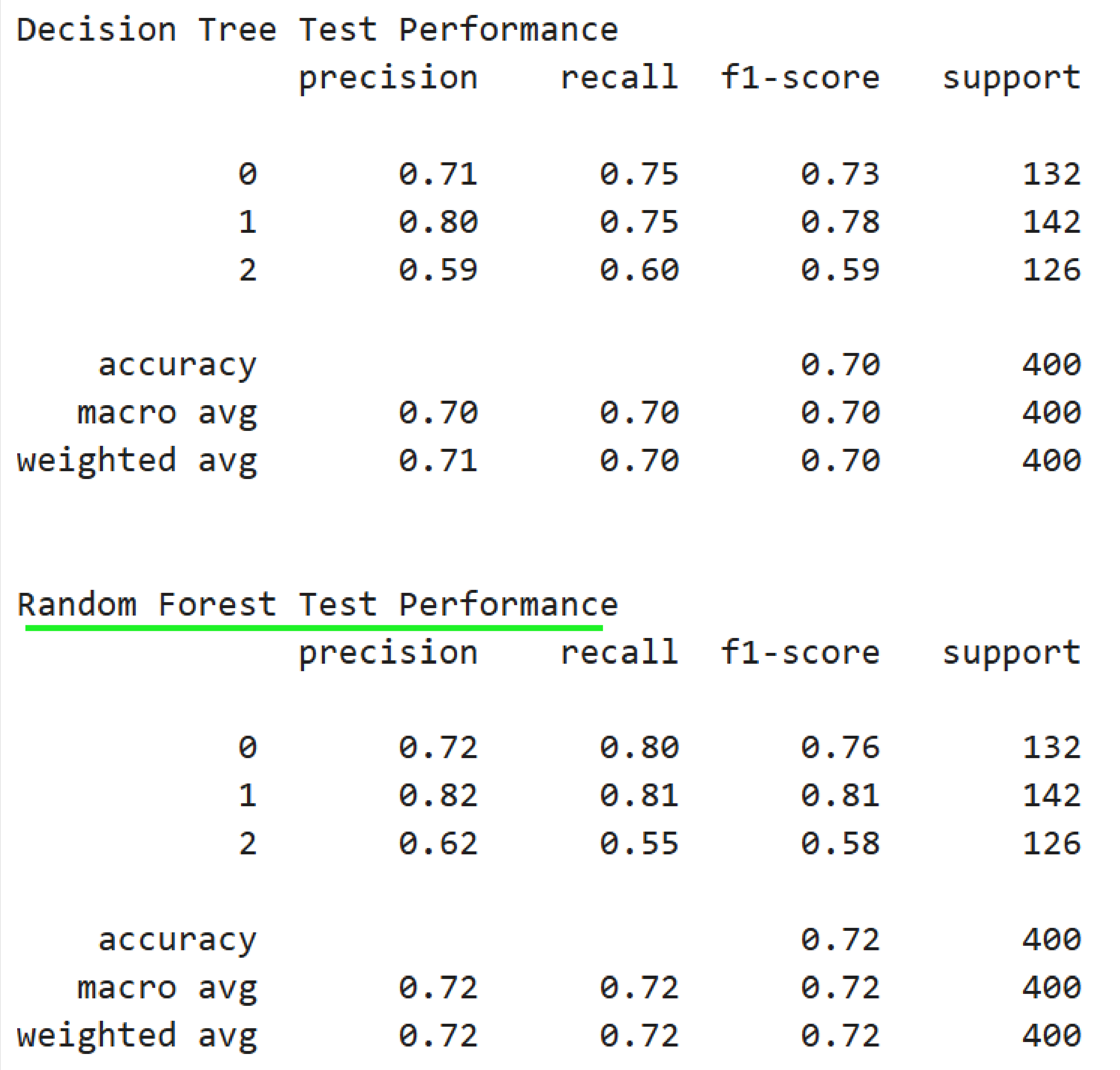
PriceCategory	PriceRange	AvgPrice	AvgRooms	AvgBedrooms	AvgBathrooms
Low	112.000 - 763.562	594.037	2,44	2,67	1,23
Mid	763.562 - 1.200.000	971.315	3,07	3,09	1,45
High	1.200.000 - 5.200.000	1.738.934	3,71	3,57	1,85

PriceCategory	AvgBuildingArea (M)	AvgLandsize (M)	AvgDistance (KMs)	MostCommonType	MostCommonRegion
Low	114,33	601,78	12,20	u - unit, duplex;	Northern Metropolitan
Mid	147,47	594,54	11,02	h - house, villa;	Southern Metropolitan
High	205,14	666,50	9,58	h - house, villa;	Southern Metropolitan

Models' Test Performance

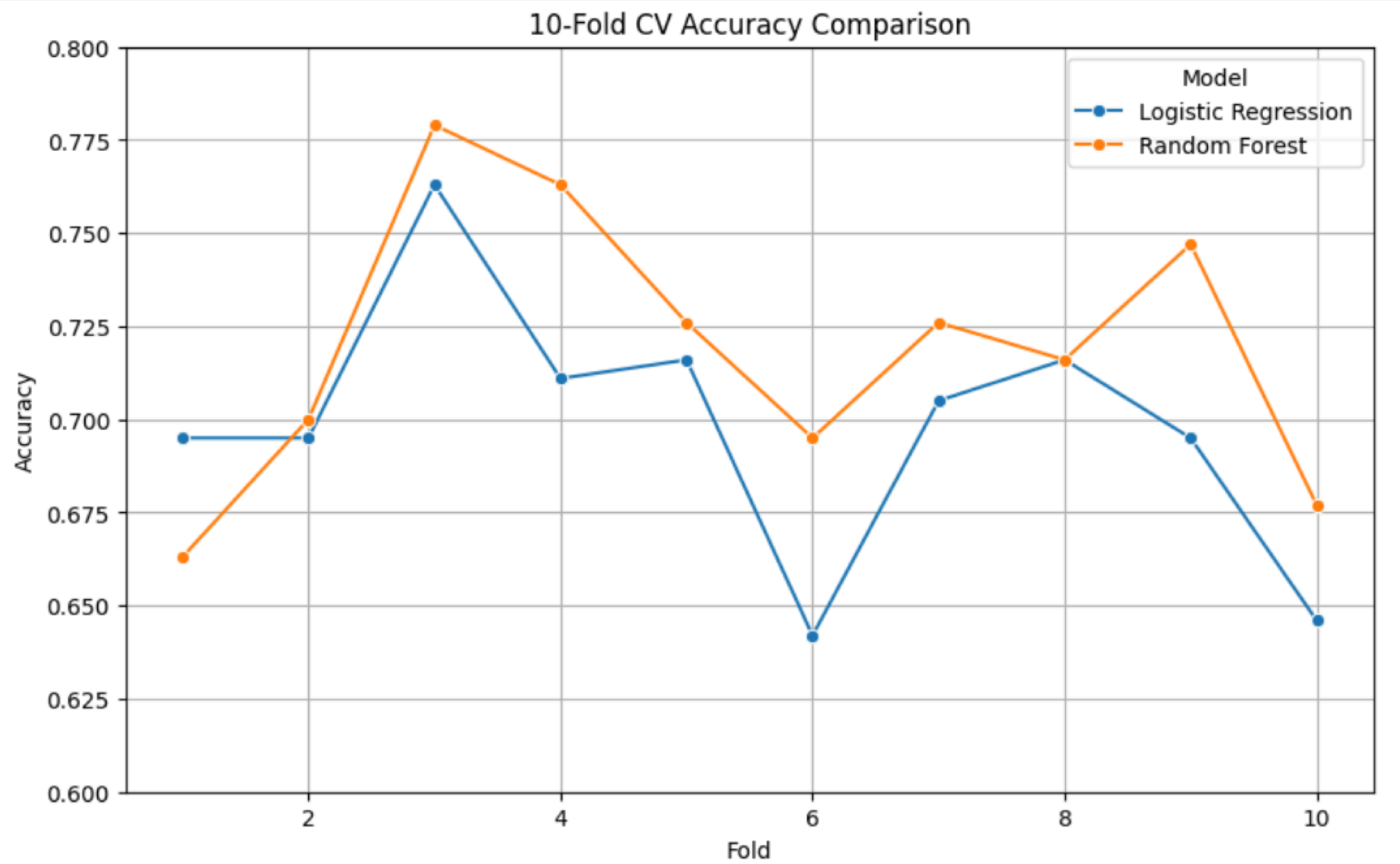


```
Logistic Regression Test Performance
              precision    recall  f1-score   support

           0       0.72      0.77      0.74       132
           1       0.83      0.77      0.80       142
           2       0.57      0.57      0.57       126

    accuracy                           0.71       400
   macro avg       0.71      0.70      0.70       400
weighted avg       0.71      0.71      0.71       400


KNN Classifier Test Performance
              precision    recall  f1-score   support

           0       0.69      0.74      0.72       132
           1       0.75      0.81      0.78       142
           2       0.55      0.45      0.50       126

    accuracy                           0.68       400
   macro avg       0.66      0.67      0.66       400
weighted avg       0.67      0.68      0.67       400
```



```
Decision Tree Test Performance
              precision    recall  f1-score   support

           0       0.71      0.75      0.73       132
           1       0.80      0.75      0.78       142
           2       0.59      0.60      0.59       126

    accuracy                           0.70       400
   macro avg       0.70      0.70      0.70       400
weighted avg       0.71      0.70      0.70       400


Random Forest Test Performance
              precision    recall  f1-score   support

           0       0.72      0.80      0.76       132
           1       0.82      0.81      0.81       142
           2       0.62      0.55      0.58       126

    accuracy                           0.72       400
   macro avg       0.72      0.72      0.72       400
weighted avg       0.72      0.72      0.72       400
```

10-Fold Cross-Validation

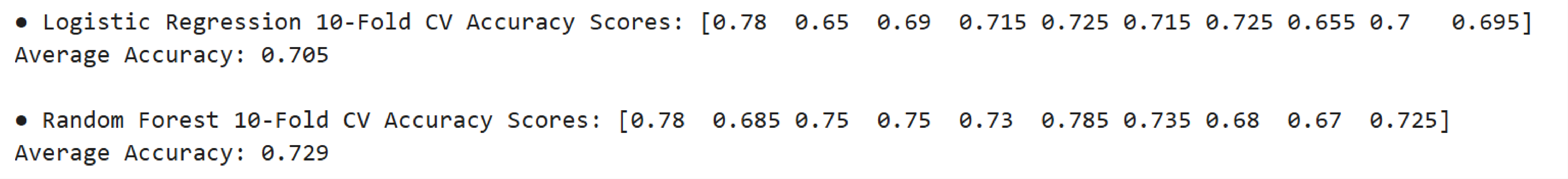


✅ **Random Forest:**

- Accuracy (Test): **0.72**
- Cross-validation accuracy ranged between **67%** and **78%**, which is consistent with test performance.

✅**Logistic Regression:**

- Accuracy (Test): **0.70**
- Cross-validation accuracy ranged between 64% and 76%, showing noticeable fluctuations.



```
• Logistic Regression 10-Fold CV Accuracy Scores: [0.78  0.65  0.69  0.715 0.725 0.715 0.725 0.655 0.7   0.695]
Average Accuracy: 0.705

• Random Forest 10-Fold CV Accuracy Scores: [0.78  0.685 0.75  0.75  0.73  0.785 0.735 0.68  0.67  0.725]
Average Accuracy: 0.729
```