Part 1 Comments:

- The variables did not need data type conversion. And there were no duplicate rows.
- The max values for MedInc, AveRooms, AveBedrms, Population, and AveOccup are oddly high. Therefore, I ran the same models on the original dataset and then again on the dataset but with the outliers (anything outside of Q1 and Q3) removed. I decided to remove the outliers because the characteristics, such as the number of rooms and bedrooms, are not realistic since a building of that size would be considered for commercial use. I found that despite the model's metrics performing worse on the surface (the numbers were lower), there were fewer false positives and false negatives on the dataset with the outliers removed.
- Visualizing the dataset, I found that the dataset is very skewed to the right. In addition, I noticed that the houses with prices above the median are symmetrical; this could possibly explain why the model metrics are almost the same for each model (i.e., precision, recall, and f1-score are close in number for each model). However, after removing the outliers, the dataset is generally a normal distribution.

Note: I ran the model twice, ones using the original cleaned dataset and then once removing the outliers. I do this to compare the scores with and without.

Which techniques did you use to train the models?

I used four supervised learning models: K-Nearest Neighbors, Decision Tree, Random Forest, and AdaBoost. Before I trained the model, I cleaned the dataset by checking and removing the duplicates and graphing the data to ensure it was correct. In order to train the model, I first split the dataset into training and testing sets using a stratified split to maintain the proportion of the target variable. In order to improve the model, I then standardized features using a StandardScaler. I also removed outliers to try to improve the model even more. Finally, I fit each model to the training data using their respective fit() methods.

Explain any techniques used to optimize model performance?

To optimize performance, I standardized features using StandardScaler so that distance-based algorithms like KNN perform optimally. In addition, I tested different test/train splits to find the best split to ensure that the class distribution of the target variable is preserved in both the training and testing sets, reducing sampling bias. Finally, I removed outliers to try to optimize the model performance even further.

Compare the performance of all models to predict the dependent variable?

KNN: This model generally performed well when features were standardized, but its performance can be sensitive to the choice of neighbors and local data variations. In addition, I

noticed that because I tested so many neighbors (1-21), this model took a noticeably larger amount of time to run.
Decision Tree: This model provided an easily interpretable model. However, it can be prone to overfitting if not properly pruned or tuned. I noticed that despite the high performance, this model had the most false positives and false negatives.
Random Forest: This model performed the best overall. Random forest had the best performance by averaging multiple trees, which reduced overfitting and was able to deliver better performance.
AdaBoost: This model also performed pretty well by sequentially correcting misclassifications, though it can be sensitive to noisy data. I think this performed well on this dataset because this dataset was pretty clean.

Which model would you recommend to be used for this dataset?

Based on the results, I would recommend using the Random Forest classifier for this dataset. Random Forest demonstrated the most balanced performance, with high accuracy and strong generalization capabilities. It effectively handles non-linear relationships and avoids overfitting better than a single Decision Tree. Additionally, this model had the lowest number of false positives and false negatives.

For this dataset, which metric is more important, why?

For this dataset, I think recall is the most important metric because it ensures that we correctly identify as many high-value houses as possible. In real estate pricing, missing a high-value house (false negatives) could result in undervaluing properties, which might lead to financial loss. However, if both false positives and false negatives need to be balanced, the F1-score would also be a suitable metric since it considers both precision and recall.

Reference:
Outlier removal: https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/
Hyperparameter: ChatGPT debug; copying from the class notes did not work correctly because the param grids are not quite correct for other models. Helped adjust the param grids for hyperparameters. In addition, helped make running the model faster because I was trying to iterate through every possible combination of params, and it would run for a very long time.