COE379L Project 2 Report

Ayushi Sapru

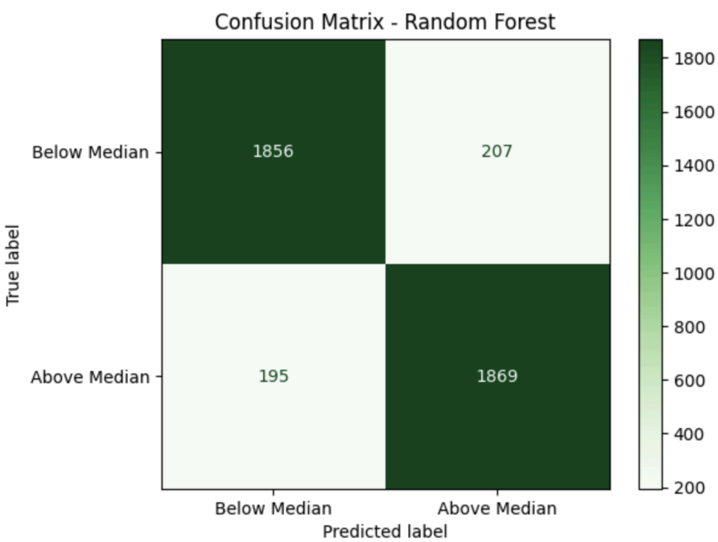
as98489

In this project, I implemented 4 supervised learning algorithms to classify whether a housing price is above or below the median value. The models trained were K-Nearest Neighbors, Decision Tree, Random Forest, and AdaBoost. Each model was trained using the train-test split method, ensuring the training and test sets kept the class distribution of the target variable.

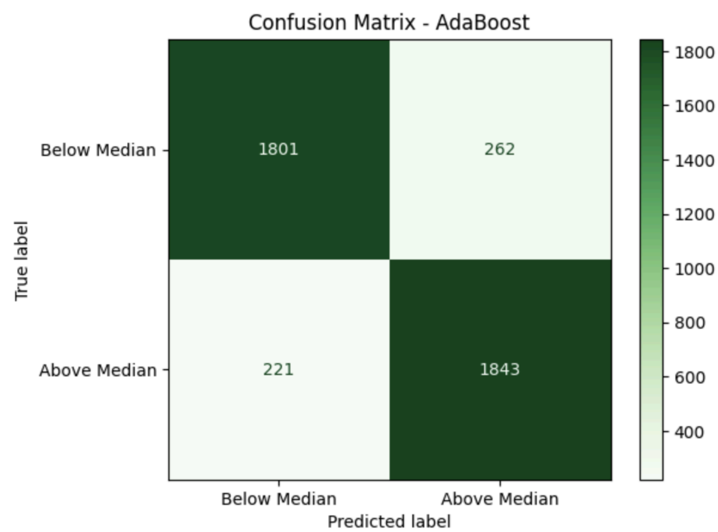
To enhance model performance, I used 2 key optimization techniques: data standardization and hyperparameter tuning. Data standardization was applied to KNN only, since it relied on distance-based calculations, while tree-based models handled raw data efficiently without needing to be scaled. Additionally, I used hyperparameter tuning with GridSearchCV to find the optimal model configurations. For KNN, I optimized n_neighbors to determine the best number of nearest neighbors. In the Decision Tree model, I tuned max_depth to prevent overfitting. For Random Forest, both n_estimators and max_depth were adjusted to balance generalization and performance. Finally, in AdaBoost, I optimized n_estimators and learning_rate to fine-tune the contribution of weak learners. These optimizations guaranteed the models performed at their best while maintaining strong generalization to unseen data.

Each model was evaluated using accuracy, precision, recall, and F1-score on both the training and test sets. When comparing the performance of the four models, I observed significant differences in their ability to classify housing prices accurately. Random Forest achieved the highest test accuracy (90.26%) and F1-score (90.29%), making it the strongest model in terms of pure predictive power. However, it also showed signs of overfitting, as it attained 100% accuracy on the training data, meaning it memorized the dataset rather than generalizing patterns. AdaBoost performed almost as well, with an accuracy of 88.30% and an F1-score of 88.41%, but with significantly less overfitting. This makes AdaBoost a more stable model for real-world applications where unseen data needs to be handled effectively. The Decision Tree model performed slightly worse, with an accuracy of 84.88% and an F1-score of 85.07%, indicating it was more prone to overfitting than AdaBoost but still provided a structured way to classify housing prices. KNN had the lowest performance, with an accuracy of 84.18% and an F1-score of 84.34%, likely due to its sensitivity to data distribution and reliance on distance-based calculations, which made it less effective for complex housing price classifications, even after optimization. While Random Forest technically outperformed all other models, its overfitting raises concerns about its reliability when applied to new, unseen data. Because of this, AdaBoost is the best choice for this dataset since it achieves high accuracy while avoiding overfitting. Unlike Decision Tree and KNN, which struggled with classification performance, AdaBoost effectively balances precision and recall, making it a strong candidate for predicting house prices above the median without being overly sensitive to training data. Therefore, while Random Forest provides the highest accuracy, AdaBoost is the more practical and robust choice for this dataset, as it maintains a balance between performance and generalization.

```
Confusion Matrix for Random Forest:
```




```
Confusion Matrix for AdaBoost:
```



For this dataset, F1-score is the most important metric because it provides a balance between precision (false positives) and recall (false negatives). In real estate pricing, both types of misclassification can have significant consequences. Incorrectly classifying a high-value home as below the median could result in missed investment opportunities, while misclassifying a low-value home as above the median could mislead buyers and investors about property values. Since both precision and recall are crucial in this scenario, only focusing on accuracy is not enough, as it does not account for the trade-off between these two errors. Precision alone would not be sufficient, as we also want to ensure that we are not missing too many actual high-value homes, and recall would not be ideal because misclassifying too many low-value homes as expensive could distort market trends. F1-score balances both precision and recall, making it the best metric to evaluate model performance for this dataset.

```
 KNNModel Performance:
Accuracy: Train = 0.8724, Test = 0.8418
Precision: Train = 0.8706, Test = 0.8348
Recall: Train = 0.8748, Test = 0.8522
F1-Score: Train = 0.8727, Test = 0.8434

 Decision TreeModel Performance:
Accuracy: Train = 0.9179, Test = 0.8488
Precision: Train = 0.9143, Test = 0.8403
Recall: Train = 0.9222, Test = 0.8614
F1-Score: Train = 0.9183, Test = 0.8507

 Random ForestModel Performance:
Accuracy: Train = 1.0000, Test = 0.9026
Precision: Train = 1.0000, Test = 0.9003
Recall: Train = 1.0000, Test = 0.9055
F1-Score: Train = 1.0000, Test = 0.9029

 AdaBoostModel Performance:
Accuracy: Train = 0.8867, Test = 0.8830
Precision: Train = 0.8876, Test = 0.8755
Recall: Train = 0.8856, Test = 0.8929
F1-Score: Train = 0.8866, Test = 0.8841
```