COE Project 1
Aidan Liu
Professor: Joe Stubbs

Which techniques did you use to train the models? (1 point)
I used the 4 base techniques to train the models: K Nearest Neighbors, Decision Tree Classifier, Random Forest Classifier, and AdaBoost Classifier. I also did take other models into consideration such as Gradient Boosting, but in this case, Random Forest and AdaBoost would handle non linear relationships better and have better reliability against outliers. Ultimately, using other models like Gradient Boosting might perform a fraction better than these 4 base techniques and worth testing out, but the results are not significant enough to consider it as the most optimal performance. To dive a little deeper into each of the classifiers we can start off with the K Nearest Neighbors (KNN) model first. Ths model is simple and effective when the data is scaled properly and standardized. I used a pipeline that combines a StandardScaler with the KNN classifier to make sure that the features were scaled on a comparable basis. Secondly, the Decision Tree Classifier uses lots of if and else statements to classify data. These could be interpreted easily and made easy to understand but these trees are usually prone to overfitting. Thirdly, we have the Random Forest Classifier, which basically builds multiple decision trees and aggregates their predictions. This way reduces overfitting and can generally improve the standardization of the data. Finally, we have the AdaBoost Classifier. This technique has a decision tree stump to act as the base estimator and builds a strong classifier from weak learners.

Explain any techniques used to optimize model performance? (1 point)
To optimize the model performance, I employed the GridSearchCV techniques with cross validation. This technique specifically reduces overfitting for the models that are used later. There were also key hyperparameters that were changed. For KNN, I searched over a range of values for the best number of neighbors (n_neightbors) while using a pipeline that had standardization. This made sure that the calculations were not biased by the scaling of the data. For the Decision Tree, I adjusted the parameters such as the max_depth to limit how far a tree can grow and also had the minimum number of samples required to split into an internal node. Changing these parameters helped control the overall overfitting because the tree wanted to overfit with perfect training scores. For the Random Forest, I used n_estimators which is the number of trees in the forest, the max_depth of the forest, min_samples_leaf, and class_weight to handle the standardization of the data. I ran into some troubles with the sheer sizing of this model while testing, namely the number of CV folds and search space size. To combat this issue I just reduced the number of CV folds from 5 to 3 and the search space size. For

AdaBoost, I just changed learning_rate and the n_estimators while using decision tree stump as the base estimator.

Compare the performance of all models to predict the dependent variable? (1 point)
A general comparison for all the models is that the top performer was the Random Forest which had a 89% test accuracy which was closely followed by AdaBoost with 88% test accuracy. KNN and Decision Tree both had around 84% after they were tuned, which was better than the baseline performance. Ultimately the two ensemble methods of Random Forest and AdaBoost outperformed the single model approaches of KNN and Decision Tree. They had higher accuracy and stronger precision/f1 scores.

KNN: Baseline KNN Performance: Test Accuracy: 83% Train Accuracy: 91% Optimized KNN Performance: Test Accuracy: 84% Train Accuracy: 86%	**Decision Tree:** Baseline Decision Tree Performance: Test Accuracy: 83% Train Accuracy: 100% Optimized Decision Tree Performance: Test Accuracy: 84% Train Accuracy: 92%
Random Forest: Optimized Random Forest Performance: Test Accuracy: 89% Train Accuracy: 100%	**AdaBoost:** Optimized AdaBoost Performance: Test Accuracy: 88% Train Accuracy: 89%

Which model would you recommend to be used for this dataset (1 point)
For this dataset, I would recommend the Random Forest Classifier because of its high performance at 89% which is the highest out of every classifier. Even though it does achieve 100% on the training set which could mean overfitting issues, the strong performance on the test set would suggest that its generalization is pretty good for this problem. Random Forest is also good because of its robustness against overfitting and its ability to handle large datasets. By averaging the predictions of multiple decision trees, the random forest would be less prone to overfitting as well. This would provide a peace of mind for the training data results, because it just means that it performed exceptionally well for the training data.

For this dataset, which metric is more important, why? (1 point)
For this dataset, I would say the F1-score has the most balanced measure. Since the classification for this dataset is for mostly the middle of things such as “above median price”, the existence of false positives and false negatives are both equally consequential without a specific one being overwhelmingly favored. The F1-score does that by offering a balance between precision and recall.