Arabel Rachman

Project 2 Report

Model Training Techniques

In order to predict whether or not a house is above or below the median price, I tried using five different models: KNN, Decision Trees, Random Forrest, AdaBoost, and Gradient Boosting. In order to train these models and find their most optimal hyperparameters, I tried using both grid search and random search cross validation techniques. I did this by giving the grid search algorithm ranges for the hyperparameters of each model and allowed the search to go through the range/distribution and find the most optimal one. Examples of hyperparameters searched for the models include the n_neighbors for the KNN and the n_estimators for the Random Forrest and AdaBoost algorithms. In the end, I found that using grid search was better than using random search by testing the two on the most optimal algorithm that I found.

ML Model Optimization

As a way to make the models work most optimally, I employed a combination of data tuning and model optimization techniques. For data tuning, I applied basic data preprocessing techniques like checking for duplicates and missing values. Furthermore, upon inspection I noticed that some data columns had a relatively high amount of outliers that were a lot larger than the mean value. In order to treat these values, I attempted to “cap” the data, setting the data below the 0.01 percentile and those above the 0.99 percentile to a max cap value. I also used a robust scaler to standardize the data, as all columns were of different range. I found that there was no need for datatype conversion. While the dependent variable, the price_above_median is set as an integer value of only 0s and 1s, I intentionally kept it as an integer to allow the model to make decimal predictions in the middle and work from that. In order to optimize the ML models, I tried using an early stopping technique, in order to reduce potential overfitting. Lastly, I tried to optimize my final models by using an ensemble technique, combining the two best performing models into a final model and I evaluated this combined model.

Arabel Rachman

Model Performance Evaluation

I will evaluate the models starting from the worst performing to the best performing model. The worst performing model was the KNN model with an accuracy of 0.83. This made sense, as the way the algorithm works would struggle with both high dimensional data and data with large noise/outliers. The Decision Tree model also tied as the worst performing, with an accuracy of 0.83. This model struggled heavily with overfitting, as its accuracy on the training data is a whopping 1.00, showing a large discrepancy between the test accuracy and training accuracy. The AdaBoost came third, with an accuracy of 0.88. The model did not suffer due to overfitting, as its accuracy on the training data was only 0.01 higher. However, its inability to see better patterns within the training data to get a higher train accuracy may have inhibited its performance. The Random Forrest came second with an accuracy of 0.89 (different from the one shown in the Notebook, as finding this best one had a very long runtime but was done in a previous iteration). Averaging many different decision trees allowed the Random Forrest to reduce the overfitting that occurred within each singular decision tree. Lastly, the Gradient Boost algorithm was the best performing, with an accuracy of 0.91. An in depth analysis of why this might be the case is found in the next part. Another important thing to note is that out of the different techniques used to optimize these models, I found that the only one that had an observable impact would be the basic data processing and the data scaling. While things like capping the data and using an ensemble model seemed hopeful in improving the accuracy, I found that they did not make any change to the highest performing 0.91 accuracy.

Best Performing Model

In the end, I found that the Gradient Boost model was the highest performing model, with an accuracy of 0.91, indicating that this model was the best at learning the underlying pattern within the training data for houses above the median price, allowing it to be the best in predicting this variable on unseen data. This ability to fit intricate relationships is because the Gradient Boost works by combining many weak learners, which allows each tree to correct small mistakes, creating a robust final combined model. Building each final model gradually, sequentially correcting errors, allowed this model to struggle less from the noise and outliers in the data and

from struggling with its own bias. The use of a learning rate also keeps the variance during model training in check.

Most Important Metric

Lastly, it is important to note that all model training was done with accuracy set as the metric being maximized. For this problem, I believe that accuracy is the most important metric to assess a model's effectiveness. The main reason for this, is that by definition, the dependent variable of such a dataset will always be evenly split, as it's based on the median of all houses within the data. Measuring accuracy allows for the evaluation of the model that is not dominated by any one class, allowing for an unbiased view of how well the model can predict the classification. Furthermore, for this problem, there is not an unequal cost for any one type of error, like a false positive or a false negative. This means that a balanced metric like accuracy would be better than one that wants to maximize either one of this type of error, like precision or recall. Lastly, simply a convenience of using accuracy is it being an intuitive metric. Making it easier to explain these models for any potential external stakeholders, such as real estate agents that would like to know how these houses were predicted, or simply hopeful homeowners.