Sahil Rangwala
MSDS 422

Week 5 Write-up

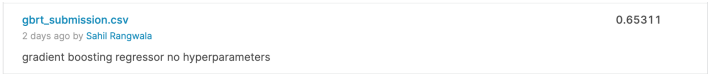
This week we explored 3 different models such as random forest regressor, gradient boosted trees, and extra trees. The generic code to implement it is from Chapter 7 Ensemble Learning and Random Forest from the textbook. It said that the extra trees and the random forest methodologies were very similar and it was derived from the same library. Surprisingly, extra trees fared better than random forest in the final run with the hyperparameters but by very little.

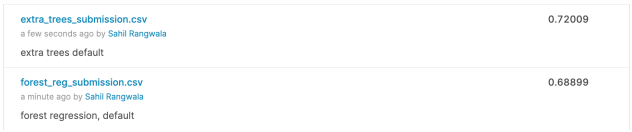
Initially, I used the default model parameters when building the model. I did this for all three different models to use as a base result. I added a few of the scores below. Next, I knew modifying the n_estimators would help create a more refined score, thus I modified that within my code and saw the cross-validation score increase slightly. I wanted to make sure this offered genuine improvement thus, I ran it through the Kaggle hidden scores. This version fared the best so far, thus I hoped that modifying the criterion on the cross-validation would matter as well.

Surprisingly, the extra trees did not fare well enough in comparison to the random forest regressor. The main difference between both models is that the random forest particularly chooses the cut points while extra trees randomly choose it. Additionally, the random trees have a much quicker computational speed as well. Additionally, Random forest uses bootstrap replicas, that is to say, it subsamples the input data with replacement, whereas Extra Trees use the whole original sample. I expected to have very small differences in the scoring of both models, but gradient boosting is where I believe it performs the best.

Gradient boosting would perform the best because of the preliminary cross-validation scores as well as the Kaggle hidden test scores. Gradient boosting is a greedy algorithm and can overfit a training dataset quickly. However, it can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting. Due to this dataset being a smaller data set I was worried there would be overfitting accidentally introduced, however, I knew this model would perform the best as it is the most optimized.

Default scores:





Actual Scores: