CYO-Cap

Executive

Dataset consists of 4843 entries of sale prices for vehicles given many feature options, vehicle type, milage, etc. My goal was to find the best algorithm to guess the sale price. I tried knn and rpart methods, but before anything I had decided which predictors to keep and I also had to normalize the data in order to avoid large RMSEs. I kept all 8 features that were listed, though it did not specify what feature they are, type of fuel that the car uses, paint color of the car, engine power, and type of car as predictors.

Methods

I decided to get rid of variables which had a lot of variation or that I deemed unhelpful such as mileage,sell date, and master key. Not shown in the script is me trying training with these variables, something that yielded high RMSEs. Thanks to the size of the dataset, I was able to use the caret package to form predictions. I first normalized the data by using the scale function, which just subtracts the mean and divides by the standard deviation. I ended up using the 90-10 split for sets since I wanted a good ammount of the data to train the algorithms and didn't want to risk undertraining. I also tuneGrided the algorithms to find the best parameters and for knn I made sure to check RMSEs for both sets to see if there was overfitting done. The result from all this was the rpart(random forest) algorithm performing slightly better but both performing with RMSE < .61. (In rerunning data, these numbers changed but rpart still performs best)

Results

The knn model gave a .601 RMSE while the rpart gave a .599. The difference is very small so I believe I can say the difference in methods isn't that meaningful from the two that I chose. I believe If I had to choose one to showcase it would be knn since the rpart ended up having a node for every entry that was provided and showing that would be very difficult.

Conclusion

In the end both algorithms performed extremely well. Given that I could do cross-validation and show it more easily, I would choose the knn algorithm if I had to perform a visual report with graphs. I believe that having a node for every entry makes it impossible to show the random trees and tried to limit it with maxNodes but it did not prove successful. I do believe that, if the prediction were categorical, rpart would prove superior as it would show nodes for every predictor as opposed for every prediction. I was considering using lda/qla but with more than 8 predictors I deemed that it may be too much and did not. In the future I can try making a list of all algorithms and simply feed it into the train function to see which one performs better training, though this idea just came to me as I am typing it.