**Predicting the Prices of Used Cars in India**

-Overview:

Our main question focuses on predicting prices in India’s used car market. We found the dataset from Kaggle, which contains relevant information and variables such as the car’s engine model and power. We performed some data cleaning to aid our analysis, such as medianImpute. We used five different methods to train our dataset: random forest, boosting, lasso regression, ridge regression and linear regression. Ultimately, we found that boosting and random forest performed the best. While working on this project, we learned that sophisticated models tend to perform better.

-The big picture:

We find our main question interesting and worth studying because we did a similar project (predicting house prices) in class, and we want to apply the skills we mastered in a fresh setting. Besides that, there already exists some used car price prediction systems online like Cars.com and Kelly Blue Book. Although these websites offer this service, their prediction method may not be the best. Therefore, we are curious about the logic and methodology behind these car valuation systems and what features are playing important roles in car residuals values as new models come out. More importantly, from an economic perspective, it is important to know their actual market value while buying-and-selling-asymmetric information may lead to inefficient market allocations.

-Data

The data used in this project was downloaded from Kaggle. It was uploaded by Kaggle.com user- Avwe Kasliwal. It contains relevant information including location, owner type, engine, power and 10 other variables. The cleaning process took us quite a while, but it seems worth it after a few trials with tidier training datasets. One of the drawbacks of this dataset is for the variable called “Names,” it contains too many details about the types of cars in each observation such as “Mahindra XUV500 W8 2WD.” Thus, we used a function called substr (substring) to select first four letters as the car brands’ names (ex. Merc stands for Mercedes-Benz), so it avoids creating more than two thousand dummy variables, with the cost of including all different series car in one general brand. Another tricky part is Mileage column which contains different fuel efficiency (kmpkg and kmpL) for traditional fossil fuel (petrol and diesel) and new energy fuel (LPG and CNG). To fix this problem, we used ifelse function to classify this variable into two different variables. We converted the variables that must be numeric by applying the as.numeric function. There’s only one feature - “seats” has missing value, and we apply to medianImpute to fill in the missing value, then center,scale afterwards to standardize remaining numerical values. The last step is to apply dummy variables function to deal with character variables.

-Methods:

We chose five methods to train our datasets: random forest, boosting, lasso regression, ridge regression and linear regression. For random forest, the tuned parameters are the number of predictors at each split and the size of node. The method for tuning that we choose is out-of-bag error. The spit rule is “extratrees” which is suitable for regression setting. For boosting, there are three tuning parameters: the number of trees to prevent the model overfitting, the interaction depth-determine tree’s complexity, the shrinkage parameter- learning rate and fixed minimum leaf size which we set for 10. We choose 5-fold cross validation for tuning parameters. For lasso and ridge regression, the tuned parameter lambda is shrinkage penalty for adding variables, and we cross validate with cv.glmnet() to pick the λ with minimum cross validated RMSE value. For linear regression, there is no tuned parameter, and we use cross validation with fold of 10 to train the model and make prediction.

-Results and conclusion:

We use MAE, RMSE, and R squared to measure performance. It turns out Boosting and Random Forest are the top two models among the five models with the lowest MAE in both test dataset and cross validation (train dataset), lower RMSE, and higher R square value. These results make sense, given that ensemble methods tend to have lower variance, and thus lead us to a better prediction. The way we dealt with names of the cars in data preprocessing may limit our model performance, but it is reasonable since the prediction behavior is better when we use dummy variables to dummy every single series of each car brand. Therefore, we stick with this plan. One of the biggest things we have learned in this process is the realization that the more sophisticated model has the better performance. Learning from previous models and ensemble multiple models are much better than simply penalizing additional variables in prediction performance.