Predicting Car Price With Trees and Lasso Regression Methods

Summer Wang, Yawen Ye, Matt Staton, Wenduo Wang

July 25, 2016

1. Our Approach and Discussion

The `Cars.csv` data contain 15 predictors for `price`, which are of categorical and numerical types. To understand the correlations between these predictors and `price`, the Generalized Linear Regression method is chosen as an intuitive baseline to explore the model.

> *For benchmark purposes, 20% of the raw data are seperated for testing, while the other 80% is used for training. "In sample" and "out-of-sample" below are defined as on the two portions of data*

1.1 Lasso Regression

For exploration purpose, Lasso method is selected for its regularization mechanism to balance the *bias-variance* trade-off in traditional approaches. This is usually expressed in the form of

$$arg\ min_\beta \sum_{i=1}^{n}(Y_i - \beta_0 - \sum_{j=1}^{p}\beta_j X_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j| \ ^{[1]}$$

The presence of λ limits the complexity of the model by penalizing the scale of β, and thus makes the method a good candidate in this study to eliminate the effect of predictors noise. `cv.glmnet` function from `glmnet` library is located for this purpose. However, it does not automatically detect interactions between predictors, which mandates manually selection. This process is assisted with IBM Watson™ [2], which listed the contribution of predictors toward price. Based upon this information, 23 interaction terms were selected. Secondly, this method is not tailored for mixed data types, so it is necessary to transform the original data into binary dummy variables, which in our study consists of 164 columns in total, compared with 17 in raw data. The details are included in the Supporting Information *Predict car price with lasso regression*. After transforming the raw data and scaling `mileage` and `featureCount`, the Lasso method returned RMSE over actual price of ~7500 (in sample), and ~8100 (out-of-sample).

1.2 Trees methods

Given the inherent limitation of `glmnet` functions, it is logical to look for alternative tools to enhance prediction accuracy. Among non-parametric techniques, trees methods are vastly popular in analytics, which are data type tolerant and produce accurate results, partly because their algorithms by default infer predictor interactions. Here we have chosen Random Forest, Boosting and its improved version XGBoost to predict car prices.

1.2.1 Random Forest and Boosting

Random Forest and Boosting are usually applied in parallel and perform similarly, and easy to implement in R given the `randomForest`, `mboost` and `gbm` libraries. As a significant advantage over `glmnet`, these functions tolerate a mixture of data types, and thus extra data transforming is obviated. Thereby with calling `randomForest` and `gbm` functions, the Random Forest and Boosting methods built a model of trees with cross-validation, and returned RMSE of 6900(in)/ and 7100(in)/7100(out), respectively.

1.2.2 XGBoost

XGBoost stands for "Extreme Gradient Boosting"[3], which is a computationally improved implementation of Boosting, which "used a more regularized model formalization to control over-fitting, which gives it better performance"[4]. Practically, the XGBoost differs from Boosting/Random Forest that by design it only take numeric data, and therefore it is necessary to convert the raw data using `sparse.model.matrix`. The output RMSE is though close to `randomForest`. The lack of improvement may indicate an inherent quality issue of the raw data.

1.2.3 Fine-tuning Function Parameters

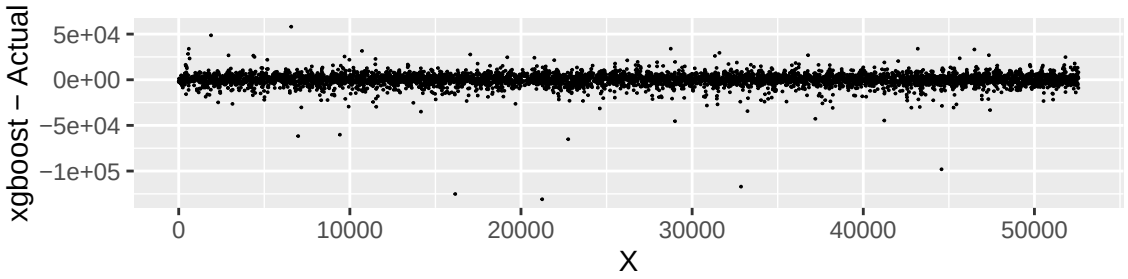
The relative ease to use non-parametric methods ironically comes with a price in terms of tuning the function parameters. Generally within trees methods there are several parameters, e.g. `n.tree`, `interaction.depth`, `shrinkage`, requiring fine-tuning to produce better fits. This process could be facilitated with the `train` function within `caret` package, which generalize the parameter optimization problem for various models. Details are available in Supporting Information *Issues with trees methods*

1.2.4 Feature Engineering and Dimension Reduction

Feature engineering and dimension reduction are essential for building robust models, particularly for Lasso regression which requires comparable predictor values. In addition, different methods have been applied to remove predictors with low contribution to the model capability. They include checking `nearZeroVar` in XGBoost and plotting *relative information* chart in Boosting. `impute` function in library `Hmisc` was employed to reduce missing values, with potential benefit dealing with sparse information.

1.3 Residuals Plot

The goal of modeling in supervised learning is reducing the residual error to a small random number. To inspect the residual error, Lasso regression, Boosting and XGBoost are selected for comparison. Since they are all similar, only XGBoost is plotted. The residuals are mostly on the negative side without an meaningful pattern, indicating many cars were sold more expensive than predicted. Reasons could be other factors including sales skills, which are not considered in the current dataset.



2. Conclusion

Through the study into the car price data set, two families of prediction models are built. In comparison, trees methods, including Random Forest and Boosting, show lower out-of-sample RMSE than Lasso method, with an advantage of ~6900 versus ~8100. Meanwhile, the models reveal relatively more significant contribution of `mileage`, `year`, `trim` and `displacement` toward `price`. Yet to further improve the model capability, extra predictors may be necessary, as seen by the lack of improvement between different algorithms.

3. Reference

[1] Carlos M. Carvalho, *class materials for Regression and Model Selection*

[2] IBM, www.ibm.com/watson

[3] Online material, http://xgboost.readthedocs.io/en/latest/model.html

[4] T. Chen, *Quora answer to what is the difference between the R gbm gradient boosting...*

4. Supporting Information

R Markdown code of this paper, *Predict car price with lasso regression R Markdown code*, *Issues with trees methods* are all available upon request.