Salomé BAUP
Camille DELGRANGE

Machine Learning Project in R : Predicting the perceived pleasantness of an odour based on some molecular features in a regression task

Introduction:

In this project, we were given access to a database of responses from a subject who had been asked to rate the pleasantness of the smell of a large number of molecules, and a comprehensive list of the physical and chemical features of the molecules smelled. The ultimate goal of our algorithm is to predict the correspondence between the pleasantness of each smell and a given molecule. The data set and the task are derived from the paper : https://science.sciencemag.org/content/355/6327/820.
You can find the complete project, documentation and dataset on our GitHub page: https://github.com/CamilleDelgrange/ML-Project

1. Exploration:

The predictors of the data set are presenting as follow:
Id - molecule identity, **Intensity** - intensity of the odor, **VALENCE.PLEASANTNESS** - how pleasant the odor is and **other columns** - physicochemical features of each of the molecules smelled by the subjects (atom types, functional groups, topological and geometrical properties,...etc).
We have a full data set composed of a training data of 708 rows and 4872 predictors, including our response data, VALENCE.PLEASANTNESS. Our test set given on which we could try our models was composed of 68 rows and 4871 predictors. For the purpose of the project the dataset has been preprocessed as follows: As "VALENCE.PLEASANTNESS" and "SWEETORSOUR" were not present in the test set: we decided to remove them from our data set but we kept the response -VALENCE.PLEASANTNESS- in another variable. Then, we transferred dummy variables into numeric form. The variable Intensity was a factor variable with 2 levels “low” and “high”, hence we coerced it into a numeric predictor. We also removed missing or NA values from the data set.

From the results of our exploration analysis, we removed strongly correlated predictors that would have predicted the response in a similar manner and predictors that followed a constant line with respect to the response variable, with zero variance, because they didn't add any information. This became our reduced data. In addition, we could infer from the histogram that the ‘VALENCE.PLEASANTNESS’ variable seemed to be normally distributed but contained several outliers.

We then split our raw data into a training and a test set, before and after reducing the predictors, to run each method two times, with unreduced and reduced data, and spot the differences. We set a specific seed for reproducibility. We tried to scale the data but it worsened our predictions for our methods (except Neural networks) so we finally removed that step and used specific scaling and unscaling methods for Neural Networks. For this project, we used the coefficient of determination, R^2, and the mean squared error (MSE) or root mean squared error (rmse) metrics to quantify the model’s performance.

2. Linear Methods

We did not obtain great results with linear regression (lm fit) because there were too many features, even with reduced predictors. We did not either obtain great results for selection of predictors with cross validation, because we only took a few predictors to do the cross validation (50 predictors) and it was not relevant enough to predict our response variable. Indeed, the best R^2 that we obtained was 0.11 for 9 predictors and the best MSE was 550.3 for 7 predictors with 10-fold cross-validation. Furthermore, with some trials of feature engineering on 2 parameters that had the highest correlation coefficient with the response variable, we found extremely low R^2 even if it was a little bit better than the simple unengineered predictors. Hence, we concluded that we could not rely on keeping only a few predictors to describe our response variable, but rather rely on keeping many predictors used in well-tuned linear or non-linear methods. Finally, forward selection with cross validation gave weird results because the lowest test MSE corresponded only to 2 valuable predictors over 60.

For Lasso regularization, we did 10-cross validation on the parameter lambda and first trained it with our train set not reduced as a control baseline for the other trials. It appeared that there was a huge overfitting in the results because the test MSE was much bigger than the training MSE. Hence, we did the same training on our train set reduced and we found our best model which produced a test MSE of 407.8 and a R^2 of 0.13 on the splitted test set. Furthermore, both test and train MSE were closer, meaning that we reduced overfitting. The train MSE was 403.1 and the R^2 0.28. We finally made our predictions on the full reduced data and our predictions are available in the github repository on the file FinalLasso.csv file in the folder “predictions”.

3. Non Linear Methods

First, we tried to fit a Neural Network. We began with simple layers as a control baseline. Then, we tried to put L1 regularization in layers in order to prevent overfitting. Finally, we added early stopping instead of regularization, not allowing the validation loss to increase. The best Neural Network was with regularization. It happened that MSEs were not that good and we were curious to process trees. So we then focused on Boosting and Random Forests. We first fitted simple trees, one with raw data and one with reduced data. Test MSE for trees on reduced data was better than on raw data, but overall, simple trees gave high MSEs. Then we tried Boosting: we first tuned lambda and tree depth manually with a for loop, and then we used cross validation with xgb.cv. The best test MSE was 428.9 obtained after manual tuning. You can see predictions on the provided testset in the file BoostedManual.csv (also submitted on Kaggle for final score). The predictions after cross validations are on BoostedXgbCv.csv. The last model tested was with Random Forests. We began by tuning manually mtry (number of predictors as split candidates) and ntree (number of trees). Then we used the tuneRF method: the best Out Of Bag (OOB) error gave us the best model we selected. We then fitted the method with the selected parameters on the training set and made predictions on the splitted test set. The best test MSE obtained was 436.9. After fitting this model on the whole reduced data you can see the predictions made on the provided test set in the file RFfinal1.csv (also submitted on Kaggle for final score). You can find another file RF2final.csv which are predictions from another RandomForest.

The best Non-Linear model based on Mean Squared Error is the Boosted one with parameters : max depth=1 and lambda=1e-2.6, followed closely by best Random Forest model with number of trees=540 and mtry = 1218. The test MSEs for both are equivalent (428.9 and 436.9, which gives rmse of 20.7 and 20.9, which is very close). For both Boosting and Random Forest we fitted the best model on all reduced data and made predictions for the provided test set, you will find them on BoostedManual.csv and RFfinal1.csv.

Conclusion

Throughout this report, we obtained insights about the whole data set provided, and from that, we tried several methods to obtain the best predictions we could. We discovered with the data set that the response variable is explained by a lot of predictors, as they are not strongly correlated with the response as a single predictor. With respect to the performance metric, the best model obtained overall was with Lasso regularization, and the second best with boosting followed closely by randomForest. We are surprised by the fact that Lasso regularization makes better than non-linear methods, so we express reservations on it. It is maybe due to the fact that tuning our hyper-parameters was done in the limit of time-consuming methods in the duration of the project, and the fact that we couldn’t tune every parameter at the same time with large ranges of values, limited by our running time. We had to make a balance between performance and time.