Report Mini-project: Regression Model

Loris Constantin
Rachel Monney

December 2020

1 Introduction

The aim was to find the best regression model to predict the perceived pleasantness of an odor.

2 Results

2.1 Exploration

We were given a set of 4780 predictors for 708 observations. To have a better overview of the pleasantness, we plotted a histogram which seems to respect a Gaussian distribution. However, there is an unexpected proportion of people at the low grades-end of the distribution. Afterwards, we deleted the variables with zero variance and plotted a sample of the data to observe the eventual linear dependencies between the predictors.

2.2 Linear models

We started with a simple regression model using the $lm()$ method as a baseline. As expected, the MSE under those conditions was really bad (1.8 mio) and the predictions far exceeded the allowed range (i.e. 0 to 100). We then tried a sub-model selection, but the linear dependencies in the predictors prevented us from setting appropriately the parameters for an optimal research. We quickly turned to Lasso (L1) regression, which happened to be the most appropriate regularization method for the linear model, which gave us a reasonable MSE of 476 when using the best λ found by cross-validation. We experimented further with Lasso to see how the data behaved, but found no better linear model than that one.

2.3 Non-linear models

We observed that linear models did not fit our data well. Therefore, we hoped that non-linear models would show better results. We tried boosting and neuronal networks.

Boosting was interesting for us because it uses only three hyper-parameters and is applicable to large data sets (we have more than 3000 predictors for about 700 observations), it does not depend on initial condition and selects feature importance.

Neural Networks had a chance to weight out the predictors in a more complex manners than linear regression could, and is widely applicable to large data sets.

2.3.1 Tree, boosting

First, we tried to fit a simple tree because the method is easy to implement and to interpret. We obtained a tree with 24 nodes and a mean squared error (MSE) of 560. This method selects automatically some predictors and it works better than linear regression without selection. This was the baseline. To improve it, we did cross-validation using the $cv.tree()$ function, and selected the size of the tree with the smallest deviation. The size was 2, and we plotted the pruned tree with this previous size. The MSE was better, but by plotting the prediction in comparison with the data, we observed an underfit.

We decided to analyse the data with boosting, which should give more accurate results as long as we avoid under- or overfitting by tuning hyper-parameters properly. At first, we computed different λ from $10^{-3.5}$ to $10^{-0.2}$, and we fixed the number of trees at 1000 and the maximal tree size at 3. With this procedure, we found the best

model with an optimal shrinkage $\lambda \approx 0.003162$ and a $MSE \approx 404$. The function $xgb.importance()$ shows the most important predictors: the first one is $Mor24m$, but is not much more significant than the following ones. To evaluate more hyper-parameters, we computed another test, by making a hyper- grid containing different values of λ, $maximalsized$, and we performed cross-validation using $xgb.cv()$ with up to $5000 nrounds$. This code has unfortunately computational cost and takes a lot of time. The better model was boosting with $d = 1$, $\lambda = 0.01$, and $nrouds = 2939$ with the $MSE \approx 423$. The most important predictor was also $mor24m$.

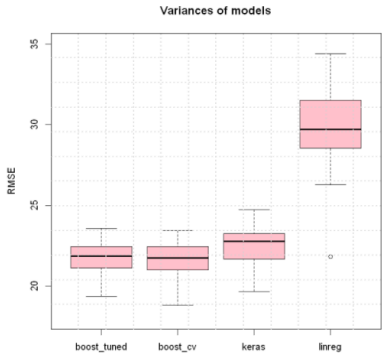
Finally, we also compared to bagging and random forest. The results were in the same range.

2.3.2 Neural Networks

We tried to fit a Neuronal Network to the scaled data (0,1), which improves the weights' convergence. After a few trials, we found that overfitting occured rather quickly and used from there on validation sets combined with early stopping not to overfit the data. We found two main sources of variability of the model around a unique parameter combination: most of it came from the split between training and test sets, and in a lesser measure from the randomness in the gradient descent process. To tune the hyper-parameters, we averaged the MSE given multiple predictions based on the same training set (to flatten gradient descent variability), for multiple combinations of 1-3 hidden layers, with 100-2000 neurons. We cross-validated this approach with different training sets and could found an optimal tuning: 2 hidden layers with shape 600 and 100 respectively, $relu$ activation, and $linear$ activation for the output layer. That tuning stands for all different training sets, despite the variability of the MSE. Then, we experimented a bit more on the last most stable conformation to pick the very best configuration, which gave $MSE = 442$. Following this approach, we could generate a model that predicts generally more spread responses than linear regression, better MSE for some splits between training and test sets, but also worse for others. We could not find a way to reduce the variability due to the choice of training and test sets to such an extent that the model would always be better than linear.

2.4 Variance of models

After tuning all our models to their best performance, we ran them on fifty different splits of the data set and plotted boxplots to compare their variance. As expected, non-linear models are better, boosting in particular having a slightly lower mean MSE the neural network, as well as less variance.In addition, our Kaggle-submissions ranked in the same order as these observations would predict.



3 Conclusion

Using non-linear models, we could most of the time predict better responses than for linear regression, with comparable performances of Neural Networks and Boosting (boosting being a bit ahead). However, the error drop is not very significant, and we could never get under a solid threshold of $MSE = 400$, with still highly variable results given the splits between training and test sets. In addition, the predictions stay pretty close to the mean, and the values at the extremes, under 15 and above 80, are never predicted within a reasonable range whatever the method and parameters. We believe that important predictors are missing or at the very least these predictors are not significant enough, making the predictions gather around the most probable answers of the training set rather than relying on heavy evidence given by the parameters.